



BSI Standards Publication

Field device tool (FDT) interface specification

Part 42: Object model integration profile – Common
Language Infrastructure (IEC TR 62453-42:2016)

National foreword

This Published Document is the UK implementation of CLC/TR IEC 62453-42:2019. It is identical to IEC TR 62453-42:2016. It supersedes PD IEC/TR 62453-42:2016, which is withdrawn.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 secretary.

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

© The British Standards Institution 2019
Published by BSI Standards Limited 2019

ISBN 978 0 539 02868 3

ICS 25.040.40; 35.100.05; 35.110

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

This Published Document was published under the authority of the Standards Policy and Strategy Committee on 31 May 2016.

Amendments/corrigenda issued since publication

Date	Text affected
31 May 2019	This corrigendum renumbers PD IEC/TR 62453-42:2016 as PD CLC/TR IEC 62453-42:2019

English Version

**Field device tool (FDT) interface specification - Part 42: Object
model integration profile - Common Language Infrastructure
(IEC TR 62453-42:2016)**

Spécification des interfaces des outils des dispositifs de
terrain (FDT) - Partie 42: Profil d'intégration des modèles
d'objets - Infrastructure commune de langage
(IEC TR 62453-42:2016)

Field Device Tool (FDT)-Schnittstellenspezifikation - Teil 42:
Profil zur Integration des Objektmodells - Common
Language Infrastructure (CLI)
(IEC TR 62453-42:2016)

This Technical Report was approved by CENELEC on 2019-03-18.

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

This document (CLC/TR IEC 62453-42:2019) consists of the text of the IEC TR 62453-42:2016 prepared by 65E: "Devices and integration in enterprise systems", of IEC technical committee 65: "Industrial process measurement, control and automation".

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.

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

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

<u>Publication</u>	<u>Year</u>	<u>Title</u>	<u>EN/HD</u>	<u>Year</u>
IEC 62453-1	2016	Field device tool (FDT) interface specification - Part 1: Overview and guidance	EN 62453-1	2017
IEC 62453-2	2016	Field device tool (FDT) interface specification - Part 2: Concepts and detailed description	EN 62453-2	2017

This page deliberately left blank

CONTENTS

FOREWORD.....	19
INTRODUCTION.....	21
1 Scope.....	23
2 Normative references.....	23
3 Terms, definitions, abbreviations and conventions.....	23
3.1 Terms and definitions.....	23
3.2 Abbreviations.....	30
3.3 Conventions.....	30
4 Implementation concept.....	31
4.1 Technological orientation.....	31
4.2 Implementation of abstract FDT object model.....	31
4.3 FDT Frame Application (FA).....	32
4.4 DTM Business Logic.....	33
4.4.1 General.....	33
4.4.2 Implementation of DTM, DTM Device Type, and Device Ident Info.....	34
4.4.3 Implementation of DTM device parameter access.....	35
4.4.4 Process Data Info.....	35
4.4.5 Diagnostic Data Info.....	36
4.4.6 Network Management Info.....	36
4.4.7 Function Info.....	37
4.4.8 Report Info.....	37
4.4.9 Document Reference Info.....	37
4.5 Implementation of DTM Functions.....	37
4.5.1 DTM User Interface.....	37
4.5.2 Function access control.....	38
4.5.3 Handling of standard UI elements in modeless DTM UI interfaces.....	38
4.5.4 Command functions.....	39
4.6 User management.....	39
4.6.1 General.....	39
4.6.2 Multi-user access.....	39
4.6.3 User levels.....	39
4.7 Implementation of FDT and system topology.....	42
4.7.1 General.....	42
4.7.2 Topology management.....	43
4.7.3 Data exchange between Frame Applications.....	45
4.8 Implementation of Modularity.....	45
4.9 Implementation of FDT communication.....	45
4.9.1 Handling of communication requests.....	45
4.9.2 Handling of communication errors.....	46
4.9.3 Handling of loss of connection.....	46
4.9.4 Point-to-point communication.....	46
4.9.5 Nested communication.....	47
4.9.6 Dynamic changes in network.....	47
4.10 Identification.....	48
4.10.1 DTM instance identification.....	48
4.10.2 Hardware identification.....	48
4.11 Implementation of DTM data persistence and synchronization.....	49

4.11.1	Persistence overview	49
4.11.2	Relations of DTMDataset	50
4.11.3	DTMDataset structure	51
4.11.4	Types of persistent DTM data	52
4.11.5	Data synchronization	52
4.12	Implementation of access to device data and IO information	53
4.12.1	Exposing device data and IO information	53
4.12.2	Data access control	54
4.12.3	Routed IO information	56
4.12.4	Comparison of DTM and device data	56
4.12.5	Support for multirole devices	57
4.13	Clone of DTM instances	58
4.13.1	General	58
4.13.2	Replicating a part of topology with Parent DTM and a subset of its Child DTMs	58
4.13.3	Cloning of a DTM without its children	58
4.13.4	Delayed cloning	58
4.14	Lifecycle concepts	59
4.15	Audit trail	59
4.15.1	General	59
4.15.2	Audit trail events	59
5	Technical concepts	60
5.1	General	60
5.2	Support of .NET Common Language Runtime versions	62
5.2.1	General	62
5.2.2	Rules for FDT .NET assemblies	62
5.2.3	DTM rules	62
5.2.4	Frame Application rules	62
5.2.5	FDT CLR extension concept	63
5.3	Support for 32-bit and 64-bit target platforms	63
5.4	Object activation and deactivation	64
5.4.1	General	64
5.4.2	Assembly loading and object creation	64
5.4.3	Assembly dependencies	65
5.4.4	Shared assemblies	65
5.4.5	Object deactivation and unloading	66
5.5	Datatypes	67
5.5.1	General	67
5.5.2	Serialization / deserialization	67
5.5.3	Support of XML	68
5.5.4	Optional elements	68
5.5.5	Verify	68
5.5.6	Clone	68
5.5.7	Equals	69
5.5.8	Lists	69
5.5.9	Nullable	70
5.5.10	Enumeration	70
5.5.11	Protocol-specific datatypes	70
5.5.12	Custom datatypes	70

5.6	General object interaction.....	73
5.6.1	General	73
5.6.2	Decoupling of FDT Objects.....	73
5.6.3	Parameter interchange with .NET datatypes	74
5.6.4	Interaction patterns	74
5.6.5	Properties	74
5.6.6	Synchronous methods.....	74
5.6.7	Asynchronous methods	75
5.6.8	Events pattern	81
5.6.9	Exception handling.....	82
5.7	Threading	86
5.7.1	Introduction.....	86
5.7.2	Threading rules.....	87
5.8	Localization support	88
5.8.1	General	88
5.8.2	Access to localized resources and culture-dependent functions.....	89
5.8.3	Handling of cultures	89
5.8.4	Switching the User Interface language.....	90
5.9	DTM User Interface implementation	90
5.9.1	General	90
5.9.2	Resizing	90
5.9.3	Private dialogs	92
5.10	DTM User Interface hosting	92
5.10.1	General	92
5.10.2	Hosting DTM WPF controls	92
5.10.3	Hosting DTM WinForms controls	93
5.11	Static Function implementation	94
5.12	Persistence	96
5.12.1	Overview	96
5.12.2	Data format.....	97
5.12.3	Adding / reading / writing / deleting of data	97
5.12.4	Searching for data.....	99
5.13	Comparison of DTM and device data	100
5.13.1	Comparison of datasets using IDeviceData / IInstanceData	100
5.13.2	Comparison of datasets using IComparison	101
5.14	Tracing	101
5.15	Report generation	101
5.15.1	General	101
5.15.2	Report types	102
5.15.3	DTM report data format	102
5.15.4	Report data exchange	103
5.16	Security	103
5.16.1	General	103
5.16.2	Strong naming of assemblies.....	103
5.16.3	Identification of origin.....	104
5.16.4	Code access security	104
5.16.5	Validation of FDT compliance certification	104
6	FDT Objects and interfaces.....	106
6.1	General.....	106

6.2	Frame Application	107
6.3	DTM Business Logic.....	109
6.3.1	DTM BL interfaces	109
6.3.2	State machines related to DTM BL	114
6.3.3	State machine of instance data.....	120
6.4	DTM User Interface	123
6.5	Communication Channel.....	124
6.6	Availability of interface methods	125
7	FDT datatypes	126
7.1	General.....	126
7.2	Datatypes – Base.....	127
7.3	General datatypes.....	127
7.4	Datatypes – DtmInfo / TypeInfo	128
7.5	Datatypes – DeviceIdentInfo.....	130
7.6	Datatypes for installation and deployment.....	135
7.6.1	Datatypes – SetupManifest.....	135
7.6.2	Datatypes – DtmManifest	136
7.6.3	Datatypes – DtmUiManifest	137
7.7	Datatypes – Communication	137
7.8	Datatypes – BusCategory	143
7.9	Datatypes – Device / Instance Data	143
7.9.1	General	143
7.9.2	Datatypes used in reading and writing DeviceData.....	150
7.10	Datatypes for export and import.....	152
7.10.1	Datatypes – TopologyImportExport.....	152
7.10.2	Datatypes – ImportExportDataset	153
7.11	Datatypes for process data description	154
7.11.1	Datatypes – ProcessDataInfo	154
7.11.2	Datatypes – Process Image.....	159
7.12	Datatypes – Address information	160
7.13	Datatypes – NetworkDataInfo	164
7.14	Datatypes – DTM functions.....	166
7.15	Datatypes – DTM messages	168
7.16	Datatypes for delegation of DTM UI dialog actions	170
7.17	Datatypes – CommunicationChannelInfo.....	170
7.18	Datatypes – HardwareIdentification and scanning	172
7.18.1	General	172
7.18.2	Datatypes – DeviceScanInfo.....	172
7.18.3	Example – HardwareIdentification and scanning for HART	173
7.19	Datatypes – DTM report types	174
7.20	Information related to device modules in a monolithic DTM	174
8	Workflows	176
8.1	General.....	176
8.2	Instantiation, loading and release	176
8.2.1	Finding a DTM BL object.....	176
8.2.2	Instantiation of a new DTM BL.....	178
8.2.3	Configuring access rights	180
8.2.4	Loading a DTM BL	181
8.2.5	Loading a DTM with Expert user level.....	182

8.2.6	Release of a DTM BL	183
8.3	Persistent storage of a DTM	184
8.3.1	Saving instance data of a DTM	184
8.3.2	Copy and versioning of a DTM instance	185
8.3.3	Dataset commit failed	186
8.3.4	Export a DTM dataset to file	186
8.4	Locking and DataTransactions in multi-user environments	187
8.4.1	General	187
8.4.2	Propagation of changes	188
8.4.3	Synchronizing DTMs in multi-user environments	190
8.5	Execution of DTM Functions	191
8.5.1	General	191
8.5.2	Finding a DTM User Interface object	191
8.5.3	Instantiation of an integrated DTM graphical user interface	192
8.5.4	Instantiation of a DTM UI triggered by the DTM BL	193
8.5.5	Instantiation of a modal DTM UI triggered by DTM BL	194
8.5.6	Release of a DTM User Interface	195
8.5.7	Release of a DTM UI triggered by the DTM BL	196
8.5.8	Release of a DTM User Interface triggered by itself	197
8.5.9	Release of a non-modal DTM User Interface triggered by a standard action	198
8.5.10	Progress indication for prolonged DTM actions	199
8.5.11	Starting an application	200
8.5.12	Terminating applications	201
8.5.13	Execution of command functions	201
8.5.14	Execution of a command function with user interface	201
8.5.15	Opening of documents	202
8.5.16	Interaction between DTM User Interface and DTM Business Logic	203
8.5.17	Interaction between DTM Business Logic and DTM User Interface	205
8.5.18	Interaction between DTM User Interface and DTM Business Logic with Cancel	206
8.5.19	Retrieving information about available Static Functions	207
8.5.20	Executing a Static Function	208
8.5.21	Executing a Static Function with multiple arguments	209
8.6	DTM communication	210
8.6.1	General	210
8.6.2	Establishing a communication connection	211
8.6.3	Cancel establishment of communication connection	212
8.6.4	Communicating with the device	212
8.6.5	Frame Application or Child DTM disconnect a device	213
8.6.6	Terminating a communication connection	214
8.6.7	DTM aborts communication connection	215
8.6.8	Communication Channel aborts communication connection	216
8.7	Nested communication	216
8.7.1	General	216
8.7.2	Communication request for a nested connection	217
8.7.3	Propagation of errors for a nested connection	218
8.8	Topology planning	219
8.8.1	General	219

8.8.2	Adding a DTM to the topology	219
8.8.3	Removing a DTM from topology.....	220
8.8.4	Frame Application creates topology	221
8.8.5	DTM generates sub-topology.....	222
8.8.6	Physical Layer and DataLinkLayer.....	224
8.9	Instantiation, configuration, move and release of Child DTMs.....	224
8.9.1	General	224
8.9.2	Instantiation and configuration of Child DTM BL.....	224
8.9.3	Interaction between Parent DTM and Child DTM.....	225
8.9.4	Interaction between Parent DTM and Child DTM using IDtmMessaging	227
8.9.5	Parent DTM moves a Child DTM.....	227
8.9.6	Parent DTM removes Child DTM	228
8.10	Topology scan.....	229
8.10.1	General	229
8.10.2	Scan of network topology	229
8.10.3	Cancel topology scan.....	230
8.10.4	Scan based DTM assignment	231
8.10.5	Manufacturer-specific device identification.....	232
8.11	Configuration of communication networks	234
8.11.1	Configuration of a fieldbus master	234
8.11.2	Integration of a passive device	235
8.12	Using IO information	235
8.12.1	Assignment of symbolic name to process data	235
8.12.2	Creation of Process Image	237
8.12.3	Validation of changes in process image while PLC is running.....	238
8.12.4	Changing of variable names using process image interface.....	239
8.13	Managing addresses	240
8.13.1	Set DTM address with user interface	240
8.13.2	Set DTM addresses without user interface	241
8.13.3	Display or modify addresses of all Child DTMs with user interface.....	242
8.14	Device-initiated data transfer.....	243
8.15	Reading and writing data.....	244
8.15.1	Read/write instance data	244
8.15.2	Read/write device data.....	246
8.16	Comparing data	248
8.16.1	Comparing device dataset and instance dataset	248
8.16.2	Comparing different instance datasets	248
8.17	Reassigning a different DtmDeviceType at a device node	249
8.17.1	General	249
8.17.2	DTM detects a change in connected device type.....	250
8.17.3	Search matching DtmDeviceTypes after incompatible device exchange.....	252
8.17.4	Reassign DtmDeviceType after incompatible device exchange.....	253
8.18	Copying part of FDT Topology	255
8.18.1	Cloning of a single DTM without Children.....	255
8.18.2	Cloning of a DTM with all its Children	256
8.19	Sequences for audit trail.....	256
8.19.1	General	256
8.19.2	Audit trail of parameter modifications in instance dataset	256
8.19.3	Audit trail of parameter modifications in device dataset.....	

8.19.4	Audit trail of function calls	258
8.19.5	Audit trail of general notification	259
9	Installation.....	259
9.1	General.....	259
9.2	Common rules.....	259
9.2.1	Predefined installation paths	259
9.2.2	Manifest files	262
9.2.3	Paths in manifest files	263
9.2.4	Common command line arguments	263
9.2.5	Digital signatures of setup components.....	264
9.3	Installation of FDT core assemblies	264
9.4	Installation of communication protocols.....	264
9.4.1	General	264
9.4.2	Registration	264
9.4.3	Protocol manifest	264
9.5	Installation of DTMs	265
9.5.1	General	265
9.5.2	Registration	266
9.5.3	DTM manifest	267
9.5.4	DTM User Interface manifest.....	268
9.6	DTM setup	269
9.6.1	Structure.....	269
9.6.2	DTM setup manifest	270
9.6.3	DTM device identification manifest	271
9.6.4	Setup creation rules	273
9.7	DTM deployment.....	274
9.8	Paths and file information	276
9.8.1	Path information provided by a DTM	276
9.8.2	Paths and persistence.....	276
9.8.3	Multi-user systems	276
10	Life cycle concept.....	276
10.1	General.....	276
10.2	Technical concept	277
10.2.1	General	277
10.2.2	DtmManifest / DtmInfo.....	278
10.2.3	TypeInfo	278
10.2.4	Supported DataSet formats	279
10.2.5	DeviceIdentInfo.....	279
10.2.6	Dataset.....	280
10.2.7	DeviceScanInfo.....	280
10.3	DTM setup	280
10.4	Life Cycle Scenarios	281
10.4.1	Overview	281
10.4.2	Search for device type in DTM setups.....	282
10.4.3	Search for installed DTMs	283
10.4.4	Dataset migration for reassigned DTM	285
11	Frame Application architectures	286
11.1	General.....	286
11.2	Standalone application	286

11.3	Remoted user Interface	286
11.4	Distributed multi-user application	287
11.5	OPC UA	287
Annex A	(normative) FDT2 Use case model	289
A.1	Use case model overview	289
A.2	Actors	289
A.3	Use cases	290
A.3.1	Use case overview	290
A.3.2	Observation use cases	291
A.3.3	Operation use cases	292
A.3.4	Maintenance use cases	294
A.3.5	Planning use cases	299
A.3.6	Main Operation	301
A.3.7	OEM Service	302
A.3.8	Administration	302
Annex B	(normative) FDT interface definition and datatypes	303
Annex C	(normative) Mapping of services to interface methods	304
C.1	General	304
C.2	DTM services	304
C.3	Presentation object services	308
C.4	General channel services	308
C.5	Process channel services	308
C.6	Communication Channel Services	309
C.7	Frame Application Services	310
Annex D	(normative) FDT version interoperability guide	313
D.1	Overview	313
D.2	General	313
D.3	Component interoperability	314
Annex E	(normative) FDT1.2.x / IEC 62453-42 Backward-Compatibility	315
E.1	Overview	315
E.2	Parallel FDT topologies	315
E.3	Mixed FDT topologies	316
E.4	FDT1.2.x / IEC 62453-42 Adapters	318
E.5	FDT1.2.x XML / IEC TR 62453-42 Datatype Transformers	319
E.5.1	General	319
E.5.2	Installation and Registration of Protocol-specific Transformers	320
E.5.3	Interaction between FDT2 and FDT1.2 components using Transformers	321
E.6	Sequences related to backward compatibility	322
E.6.1	General	322
E.6.2	Dataset migration from FDT1.x DTM to FDT2.x DTM	322
Annex F	(informative) Implementation Hints	324
F.1	IAAsyncResult pattern	324
F.2	Threading Best Practices	325
Annex G	(informative) Trade names	326
Annex H	(informative) UML Notation	327
H.1	General	327
H.2	Class diagram	327
H.3	Statechart diagram	

H.4	Use case diagram	331
H.5	Sequence diagram	332
H.6	Object diagram.....	336
Annex I (informative) Physical Layer Examples.....		337
I.1	General.....	337
I.2	Interbus S	337
I.3	PROFIBUS.....	337
I.4	PROFINET.....	337
Annex J (informative) Predefined SemanticIds.....		339
J.1	General.....	339
J.2	Data	339
J.3	Images.....	339
J.4	Documents.....	339
Bibliography		341
Figure 1 – Relation of IEC 62453-42 to the IEC 62453 series.....		21
Figure 2 – IEC 62453-42 Object Model.....		32
Figure 3 – Frame Application		32
Figure 4 – DTM Business Logic.....		34
Figure 5 – DTM, Device Type and Device Ident Info		35
Figure 6 – Process Data Info.....		36
Figure 7 – Logical topology and physical topology		43
Figure 8 – FDT and logical topology		43
Figure 9 – DTMs and physical topology		44
Figure 10 – Point-to-point communication		46
Figure 11 – Nested communication		47
Figure 12 – Identification of connected devices		49
Figure 13 – FDT storage and synchronization mechanism.....		50
Figure 14 – Relation between DTMDataset, DTM instance, and device.....		50
Figure 15 – DTMDataset structure		51
Figure 16 – Data Synchronization.....		53
Figure 17 – Routed IO information.....		56
Figure 18 – Multirole Device.....		57
Figure 19 – FDT .NET Assemblies		60
Figure 20 – FDT Object implementation.....		61
Figure 21 – FDT CLR extension concept		63
Figure 22 – Example: Assembly.LoadFrom().....		64
Figure 23 – Example: Assembly dependencies		65
Figure 24 – Example: Datatype definition		67
Figure 25 – Example: Data cloning.....		69
Figure 26 – Example: Methods without data cloning.....		69
Figure 27 – Protocol-specific datatypes		70
Figure 28 – Protocol manifest and type info attributes.....		71
Figure 29 – Example: Protocol assembly attributes.....		72

Figure 30 – Example: Handling of protocol-specific assemblies in Frame Application.....	72
Figure 31 – Decoupled FDT Objects in IEC 62453-42	73
Figure 32 – IAsyncResult pattern: blocking call.....	76
Figure 33 – Example: Blocking use of asynchronous interface	76
Figure 34 – IAsyncResult pattern (simplified): blocking call	77
Figure 35 – IAsyncResult pattern: non-blocking call	77
Figure 36 – Example: Non-blocking use of asynchronous interface	78
Figure 37 – IAsyncResult pattern (simplified depiction): non-blocking call	78
Figure 38 – IAsyncResult pattern: canceling an operation	80
Figure 39 – IAsyncResult pattern: providing progress events	81
Figure 40 – Frame Application's host window providing scroll bars.....	91
Figure 41 – Control using internal scrollbars.....	91
Figure 42 – Example: Hosting a DTM WPF control in a WPF Frame Application	93
Figure 43 – Example: Hosting a DTM WPF control in a WinForms Frame Application	93
Figure 44 – Example: Hosting DTM WinForms controls in a WinForms Frame Application	94
Figure 45 – Example: Hosting a DTM WinForms control in a WPF Frame Application	94
Figure 46 – Relation of StaticFunctionDescription to Static Function	95
Figure 47 – DTMDataset structure.....	96
Figure 48 – Example: Initialization of DTMDataset with DTM data	98
Figure 49 – Example: Writing of DTM data in DTMDataset.....	98
Figure 50 – Example: Reading of DTM data from a DTMDataset.....	99
Figure 51 – Example: Creation of a BulkData.DTMDataset with descriptor	100
Figure 52 – Example: Searching for DTMDatasets with specific descriptor	100
Figure 53 – Skeleton of a DTM-specific report fragment.....	103
Figure 54 – Example: Authenticode check	104
Figure 55 – Example: Conformity record file	105
Figure 56 – Example: checking conformity record file	106
Figure 57 – Frame Application interfaces.....	107
Figure 58 – DTM Business Logic interfaces (Part 1)	110
Figure 59 – DTM Business Logic interfaces (Part 2)	111
Figure 60 – State machine of DTM BL	115
Figure 61 – Online state machine of DTM.....	117
Figure 62 – Modifications of data through a DTM.....	120
Figure 63 – ModifiedInDtm: State machine of instance data	121
Figure 64 – ModifiedInDevice: State machine related to device data	122
Figure 65 – DTM UI interfaces	123
Figure 66 – Communication Channel interfaces	124
Figure 67 – FdtDatatype and FdtList	127
Figure 68 – DtmInfo / TypeInfo – datatypes	129
Figure 69 – DeviceIdentInfo – datatypes.....	131
Figure 70 – DeviceIdentInfo – Example for HART	132
Figure 71 – Example: DeviceIdentInfo creation.....	134

Figure 72 – Example: Using DeviceIdentInfo	135
Figure 73 – Example: DeviceIdentInfoTypeAttribute	135
Figure 74 – SetupManifest – datatypes	135
Figure 75 – DtmManifest – datatypes	136
Figure 76 – DtmUiManifest – datatypes	137
Figure 77 – Communication datatypes – Connect	138
Figure 78 – Communication datatypes – Transaction	138
Figure 79 – Communication datatypes – Disconnect	139
Figure 80 – Communication datatypes – Subscribe	139
Figure 81 – Communication datatypes – Scanning	140
Figure 82 – Communication datatypes – Address setting	140
Figure 83 – Example: Communication – Connect for HART	142
Figure 84 – Example: Communication – CommunicationType for HART	143
Figure 85 – BusCategory – datatypes	143
Figure 86 – Device / Instance data – datatypes	144
Figure 87 – Example: Providing information on data of a HART device	146
Figure 88 – Example: Providing information on module data of a PROFIBUS device	147
Figure 89 – Example: Providing information on data	148
Figure 90 – Example: Providing information on structured data	149
Figure 91 – EnumInfo – datatype	150
Figure 92 – Read and Write Request – datatypes	150
Figure 93 – ResponseInfo – datatype	151
Figure 94 – TopologyImportExport – datatypes	152
Figure 95 – ImportExportDataset – datatypes	153
Figure 96 – ProcessDataInfo – datatypes	154
Figure 97 – IOSignalInfo – datatypes	155
Figure 98 – Example: ProcessDataInfo for HART (UML)	157
Figure 99 – Example: ProcessDataInfo creation for HART	158
Figure 100 – Example: Using ProcessData for HART	159
Figure 101 – Example: IOSignalInfoType attribute	159
Figure 102 – ProcessImage – datatypes	160
Figure 103 – AddressInfo – datatypes	161
Figure 104 – Example: AddressInfo creation	162
Figure 105 – Example: Using AddressInfo	163
Figure 106 – Example: DeviceAddressTypeAttribute	163
Figure 107 – NetworkDataInfo – datatypes	164
Figure 108 – Example: NetworkDataInfo creation example	165
Figure 109 – Example: NetworkDataInfo using example	166
Figure 110 – Example: NetworkDataTypeAttribute example	166
Figure 111 – DTM Function – datatypes	167
Figure 112 – DTM Messages – datatypes	169
Figure 113 – ActionItem – datatypes	170
Figure 114 – CommunicationChannelInfo – datatypes	170

Figure 115 – Example: Channel information	171
Figure 116 – DeviceScanInfo – datatypes.....	172
Figure 117 – Example: HARTDeviceScanInfo – datatype	173
Figure 118 – DTM Report – datatypes	174
Figure 119 – Information related to device modules	175
Figure 120 – Finding a DTM BL object.....	177
Figure 121 – Instantiation of a new DTM BL	179
Figure 122 – Configuration of user permissions	181
Figure 123 – Loading a DTM BL	182
Figure 124 – Loading a DTM with Expert user level	183
Figure 125 – Release of a DTM BL.....	184
Figure 126 – Saving data of a DTM	185
Figure 127 – Dataset commit failed	186
Figure 128 – Export a DTM dataset to file.....	187
Figure 129 – Propagation of changes	189
Figure 130 – Synchronizing DTMs in multi-user environments.....	190
Figure 131 – Finding a DTM User Interface	192
Figure 132 – Instantiation of a DTM User Interface	193
Figure 133 – Instantiation of a DTM UI triggered by DTM BL.....	194
Figure 134 – Instantiation of a modal DTM UI triggered by DTM BL.....	195
Figure 135 – Release of a DTM User Interface	196
Figure 136 – Release of a DTM UI triggered by the DTM BL	197
Figure 137 – Release of a DTM User Interface triggered by itself.....	198
Figure 138 – Release of a non-modal DTM UI triggered by a standard action	198
Figure 139 – Progress indication for prolonged DTM actions.....	199
Figure 140 – Starting an application	200
Figure 141 – Execute a command function	201
Figure 142 – Execute a command function with user interface	202
Figure 143 – Opening a document.....	203
Figure 144 – Interaction triggered by the DTM User Interface	204
Figure 145 – Interaction triggered by the DTM Business Logic	205
Figure 146 – Interaction triggered and canceled by the DTM User Interface	206
Figure 147 – Retrieving information about available Static Functions	207
Figure 148 – Example: Information about available Static Functions	208
Figure 149 – Executing a Static Function.....	209
Figure 150 – Executing a Static Function with multiple Arguments	210
Figure 151 – Establishing a communication connection	211
Figure 152 – DTM cancels ongoing Connect operation	212
Figure 153 – Communicating with the device.....	213
Figure 154 – Child DTM disconnects	214
Figure 155 – Child DTM terminates a connection.....	215
Figure 156 – Child DTM aborts a connection	215
Figure 157 – Communication Channel aborts a connection	216

Figure 158 – Example: Nested communication behavior	217
Figure 159 – Example: Nested communication data exchange	218
Figure 160 – Add DTM to topology	220
Figure 161 – Removing a DTM from topology	221
Figure 162 – Frame Application creates topology	222
Figure 163 – DTM generates sub-topology	223
Figure 164 – Instantiation and configuration of Child DTM BL	225
Figure 165 – Interaction between Parent DTM and Child DTM	226
Figure 166 – Interaction using IDtmMessaging	227
Figure 167 – Parent DTM moves a Child DTM	228
Figure 168 – Parent DTM removes Child DTM	229
Figure 169 – Scan of network topology.....	230
Figure 170 – Cancel topology scan	231
Figure 171 – Scan based DTM assignment.....	232
Figure 172 – Manufacturer-specific device identification	233
Figure 173 – Configuration of a fieldbus master.....	234
Figure 174 – Integration of a passive device.....	235
Figure 175 – Assignment of process data	236
Figure 176 – Creation of process image	238
Figure 177 – Validation of changes while PLC is running	239
Figure 178 – Changing of variable names using process image interface	240
Figure 179 – Set DTM address with UI	241
Figure 180 – Set DTM addresses without UI.....	242
Figure 181 – Display or modify child addresses with UI.....	243
Figure 182 – Device-initiated data transfer	244
Figure 183 – Read/write instance data	245
Figure 184 – Read/write device data	247
Figure 185 – Comparing device dataset and instance dataset.....	248
Figure 186 – Compare instance data with persisted dataset.....	249
Figure 187 – DTM triggers ActiveTypeChanged event.....	251
Figure 188 – Find matching DtmDeviceTypes after incompatible device exchange	253
Figure 189 – Reassign a DtmDeviceType after incompatible device exchange.....	254
Figure 190 – Clone DTM without children	255
Figure 191 – Clone DTM with all children	256
Figure 192 – Audit trail of parameter modifications in instance dataset.....	257
Figure 193 – Audit trail of parameter modifications in device.....	258
Figure 194 – Audit trail of function calls.....	258
Figure 195 – GAC and FDT_Registry	261
Figure 196 – Installation paths (with example DTM).....	262
Figure 197 – Example: Protocol manifest.....	265
Figure 198 – Search for installed DTMs.....	266
Figure 199 – Example: DtmManifest.....	268
Figure 200 – Example: DtmUiManifest.....	269

Figure 201 – DTM setup structure	270
Figure 202 – Example: DtmSetupManifest	271
Figure 203 – Example: DeviceIdentManifest	273
Figure 204 – DTM deployment	275
Figure 205 – Overview DTM identification.....	277
Figure 206 – Identification attributes in DTM setup	281
Figure 207 – Check DTM Setup for list of supported types	283
Figure 208 – Scan installed DTMs	284
Figure 209 – Dataset migration to a reassigned DtmDeviceType	285
Figure 210 – Client / Server Application	286
Figure 211 – Example for distributed multi-user application.....	287
Figure 212 – OPC UA server based on IEC TR 62453-42	288
Figure A.1 – Main use case diagram	289
Figure A.2 – Observation use cases	291
Figure A.3 – Operation use cases	293
Figure A.4 – Maintenance use cases	295
Figure A.5 – Planning use cases	299
Figure E.1 – Example: IEC TR 62453-42 Frame Application with FDT1.2.x backward- compatibility support	315
Figure E.2 – IEC TR 62453-42 Frame Application with FDT1.2.x Device DTM	316
Figure E.3 – IEC TR 62453-42 Frame Application with FDT1.2.x Comm. and Gateway DTM	317
Figure E.4 – IEC TR 62453-42 Frame Application with FDT1.2.x Gateway DTM	317
Figure E.5 – IEC TR 62453-42 – FDT1.2 interaction using transformer.....	322
Figure E.6 – Dataset migration from FDT1.x DTM to FDT2.x DTM.....	323
Figure H.1 – Note	327
Figure H.2 – Class	327
Figure H.3 – Association	327
Figure H.4 – Navigable Association	328
Figure H.5 – Composition.....	328
Figure H.6 – Aggregation	328
Figure H.7 – Dependency.....	328
Figure H.8 – Association class	328
Figure H.9 – Abstract class, Generalization and Interface	329
Figure H.10 – Interface related notations	329
Figure H.11 – Multiplicity.....	330
Figure H.12 – Enumeration datatype	330
Figure H.13 – Elements of UML statechart diagrams.....	330
Figure H.14 – Example of UML state chart diagram	331
Figure H.15 – UML use case syntax	331
Figure H.16 – UML sequence diagram.....	332
Figure H.17 – Empty UML sequence diagram frame	332
Figure H.18 – Object with life line and activation.....	333
Figure H.19 – Method calls	333

Figure H.20 – Modeling guarded call and multiple calls.....	333
Figure H.21 – Call to itself.....	334
Figure H.22 – Continuation / StateInvariant	334
Figure H.23 – Alternative fragment.....	335
Figure H.24 – Option fragment	335
Figure H.25 – Loop combination fragment	335
Figure H.26 – Break notation	335
Figure H.27 – Sequence reference	336
Figure H.28 – Objects	336
Figure H.29 – Object association.....	336
Table 1 – FDT User levels.....	40
Table 2 – Role dependent Access Rights and User Interfaces for DTMs.....	41
Table 3 – Description of properties related to data access control.....	55
Table 4 – Supported CLR versions	62
Table 5 – Frame Application interfaces.....	108
Table 6 – DTM Business Logic interfaces	112
Table 7 – Availability of interfaces depending of type of DTM.....	113
Table 8 – Definition of DTM BL state machine	116
Table 9 – Definition of online state machine	118
Table 10 – Description of instance dataset states	121
Table 11 – Description of dataset states regarding online modifications	122
Table 12 – DTM UI interfaces.....	124
Table 13 – Communication Channel interfaces	125
Table 14 – Availability of DTM BL methods in different states	125
Table 15 – FDT base datatypes.....	127
Table 16 – FDT General datatypes.....	128
Table 17 – DtmInfo datatype description.....	129
Table 18 – DeviceIdentInfo datatype description.....	131
Table 19 – DeviceIdentInfo – Example for HART	133
Table 20 – SetupManifest datatype description.....	136
Table 21 – DtmManifest datatype description	136
Table 22 – DtmUiManifest datatype description	137
Table 23 – Communication datatype description	141
Table 24 – BusCategory datatype description	143
Table 25 – DeviceData datatype description	145
Table 26 – Reading and Writing datatype description.....	150
Table 27 – Reading and Writing datatype description.....	151
Table 28 – TopologyImportExport datatype description	153
Table 29 – ImportExportDataset datatype description	153
Table 30 – ProcessDataInfo datatype description	155
Table 31 – IOSignalInfo datatype description.....	156
Table 32 – ProcessImage datatype description.....	160

Table 33 – AddressInfo datatype description	161
Table 34 – NetworkDataInfo datatype description	165
Table 35 – DTM Function datatype description	168
Table 36 – DTM Messages datatype description	169
Table 37 – ActionItem datatype description	170
Table 38 – CommunicationChannelInfo datatype description	171
Table 39 – DeviceScanInfo datatype description	172
Table 40 – Example: HARTDeviceScanInfo datatype description	173
Table 41 – Reporting datatype description	174
Table 42 – Predefined FDT installation paths	259
Table 43 – Predefined setup properties	263
Table 44 – Setup command line parameters	263
Table 45 – DTM identification	278
Table 46 – DtmType – user readable description of supported types	278
Table 47 – TypeInfo identification	279
Table 48 – DtmType – Dataset support identification	279
Table 49 – Dataset identification	280
Table 50 – DeviceScanInfo – scanned device identification	280
Table 51 – Setup information	281
Table 52 – Changing DTM— overview	282
Table A.1 – Actors	290
Table A.2 – Observation use cases	291
Table A.3 – Operation use cases	293
Table A.4 – Maintenance use cases	296
Table A.5 – Planning use cases	299
Table C.1 – General services	304
Table C.2 – DTM services related to installation	304
Table C.3 – DTM service related to DTM Information	304
Table C.4 – DTM services related to DTM state machine	305
Table C.5 – DTM services related to function	305
Table C.6 – DTM services related to documentation	306
Table C.7 – DTM services to access the instance data	306
Table C.8 – DTM services to access diagnosis	306
Table C.9 – DTM services to access to device data	306
Table C.10 – DTM services related to network management information	307
Table C.11 – DTM services related to online operation	307
Table C.12 – DTM services related to FDT-Channel objects	307
Table C.13 – DTM services related to import and export	308
Table C.14 – DTM services related to data synchronization	308
Table C.15 – DTM UI state control	308
Table C.16 – General channel service	308
Table C.17 – Channel services for IO related information	309
Table C.18 – Channel services related to communication	309

Table C.19 – Channel services related sub-topology management	309
Table C.20 – Channel services related to functions	310
Table C.21 – Channel services related to scan	310
Table C.22 – FA services related to general events	310
Table C.23 – FA services related to topology management	311
Table C.24 – FA services related to redundancy	311
Table C.25 – FA services related to storage of DTM data	311
Table C.26 – FA services related to DTM data synchronization	311
Table C.27 – FA related to presentation	312
Table C.28 – FA services related to audit trail	312
Table D.1 – Interoperability between components of different versions	314
Table E.1 – Adapter interface mappings	319

INTERNATIONAL ELECTROTECHNICAL COMMISSION

FIELD DEVICE TOOL (FDT) INTERFACE SPECIFICATION –

**Part 42: Object model integration profile –
Common Language Infrastructure**

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.

The main task of IEC technical committees is to prepare International Standards. However, a technical committee may propose the publication of a technical report when it has collected data of a different kind from that which is normally published as an International Standard, for example "state of the art".

IEC TR 62453-42, which is a technical report, has been prepared by subcommittee 65E: Devices and integration in enterprise systems, of IEC technical committee 65: Industrial-process measurement, control and automation:

The text of this technical report is based on the following documents:

Enquiry draft	Report on voting
65E/439/DTR	65E/486/RVC

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

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

A list of all parts 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 publication will remain unchanged until the stability date indicated on the IEC web site under "<http://webstore.iec.ch>" in the data related to the specific publication. At this date, the publication will be

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

A bilingual version of this publication may be issued at a later date.

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

INTRODUCTION

This Part of IEC 62543, which is a technical report, is an interface specification for developers of FDT (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, the unambiguous definition of engineering interfaces that are easy to use for all those involved is of great importance.

A device-specific software component, called DTM (Device Type Manager), is supplied by the field device manufacturer with its device. The DTM is integrated into engineering tools via the FDT interfaces defined in this specification. The approach to integration, in general, is open for all kind of fieldbusses and thus meets the requirements for integrating different kinds of devices into heterogeneous control systems.

Figure 1 shows how IEC TR 62453-42 is related to the IEC 62453 series.

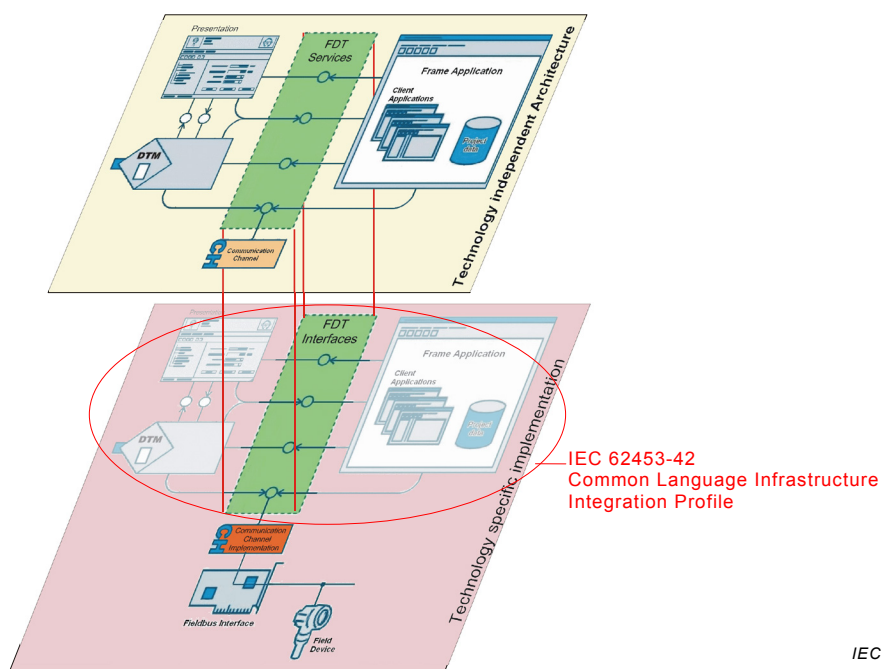


Figure 1 – Relation of IEC 62453-42 to the IEC 62453 series

The document structure is:

- Clause 3 explains the used terms, definitions and conventions
- Clause 4 introduces the general concepts of IEC 62453-42
- Clause 5 describes the technical concepts used to implement IEC 62453-42 and how FDT concepts are mapped to .NET Framework
- Clause 6 provides an overview of the FDT Objects, their interfaces and behavior
- Clause 7 presents an overview of the IEC 62453-42 datatypes
- Clause 8 shows the interaction of FDT Objects at runtime
- Clause 9 explains rules related to installation and deployment of DTMs

- Clause 10 explains how FDT life cycle concepts are implemented
- Clause 11 shows examples for Frame Application architectures

FIELD DEVICE TOOL (FDT) INTERFACE SPECIFICATION –

Part 42: Object model integration profile – Common Language Infrastructure

1 Scope

This part of IEC 62453, which is a technical report, defines how the common FDT principles are implemented based on the .NET technology, including the object behavior and object interaction via .NET interfaces.

This document specifies FDT version 2.0.

2 Normative references

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

IEC 62453-1:—¹, *Field Device Tool (FDT) interface specification – Part 1: Overview and guidance*

IEC 62453-2:—¹, *Field Device Tool (FDT) interface specification – Part 2: Concepts and detailed description*

3 Terms, definitions, abbreviations and conventions

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 62453-1, IEC 62453-2 as well as the following apply.

3.1.1 action

execution of a function which may involve several calls to interface methods of different FDT Objects

3.1.2 asynchronous methods

methods that trigger execution of asynchronous operations

Note 1 to entry: See also 5.6.7.

3.1.3 asynchronous operation

operation that is performed while the FDT object (client) that has requested the operation does not wait for the result, but the client is notified when the operation is finished

¹ To be published concurrently with this technical report.

3.1.4

bulk data

device node-specific persisted data, which is stored besides DTM instance data

Note 1 to entry: Example for bulk data: accumulated historical data, used for trend analysis.

3.1.5

bulk operation

operation to perform one or more tasks at a group of devices nodes

Note 1 to entry: Examples for bulk operation: up- or download for a group of devices, parameter adjustment for a group of devices or report generation for a group of devices.

3.1.6

clone DTM instance

process of creating a new device node in the FDT topology based on an existing device node

Note 1 to entry: This includes copying DTM instance(see 3.1.11) and resetting device node-specific DTM data.

Note 2 to entry: The identification attributes of the device are changed.

3.1.7

Communication Channel

component representing access to a fieldbus segment or to other means of communication

3.1.8

compatibility

feature of a component (hardware or software) that enables it to be interoperable with another component

3.1.9

backward compatibility

feature of a component (hardware or software) that enables it to replace an other version of the component

3.1.10

compatibility attributes

attributes used to find compatible components, to replace components or to validate compatibility after a component replacement

Note 1 to entry: Compatibility attributes are required to check whether a component is compatible with another component.

Note 2 to entry: Compatibility attributes are used to define compatibility in regard to 3.1.8.2, compatibility.

3.1.11

copy DTM instance

process of creating a new device node in the FDT topology based on an existing device node

Note 1 to entry: This includes loading the original DTM dataset to initiate the new DTM instance.

Note 2 to entry: The identification attributes of the device are not changed.

3.1.12

copy device node in FDT Topology

(see 3.1.11 copy DTM instance)

3.1.13

DataTransaction

transaction regarding the data of a DTM (persistent or device data)

3.1.14**delegate**

reference type that can be used to encapsulate a method

3.1.15**device configuration**

process of changing data related to device-specific characteristics/basic behavior

Note 1 to entry: Example for such characteristics may be the structure of a remote I/O or the type of measurement procedure like absolute pressure or differential pressure.

3.1.16**device data**

configuration data that resides on the device

3.1.17**device parameterization**

process of changing device-specific data in order to adjust application-specific behavior

3.1.18**device node**

node in the topology, which represents a device

Note 1 to entry: A DtmDeviceType is assigned to a device node, which is instantiated to operate the device in online or offline modes. See Figure 14.

3.1.19**device type check**

process of checking the device type when a DTM is going online with a connected physical device

Note 1 to entry: The DTM shall reject to go online if the connected physical device type is not supported. The check shall be based on same information as in DeviceIdentInfo. See 6.3.2.3.

3.1.20**DD-Interpreter DTMs**

DTMs which interpret device descriptions at runtime

3.1.21**DTM**

software component containing device-specific application software, including DTM Business Logic, DTM User Interface and related objects (e.g. Communication Channel)

Note 1 to entry: Older FDT specification documents used the term “DTM” for the object DTM Business Logic as well as for the whole component consisting of DTM BL, DTM UI and channels.

3.1.22**DTM Business Logic****DTM BL**

part of the DTM, which contains all the functionality to access storage and communication and which manages the instance data of a DTM

3.1.23**DTM Identifier**

identifier, which is used to identify a DTM (DTM BL class)

Note 1 to entry: In order to uniquely identify a DTM BL class, the property DtmInfo.Id is used.

3.1.24**DTM instance modal**

prevents user interaction with other windows of the DTM instance (see modal window)

3.1.25**DTM User Interface (DTM UI)**

part of the DTM, which is displayed to a human user

3.1.26**DtmType**

either DtmDeviceType, DtmModuleType or DtmBlockType

Note 1 to entry: All DtmTypes provide identification in DtmInfo class.

Note 2 to entry: DtmType is described by TypeInfo.

3.1.27**DtmDeviceType**

element of a DTM software supporting one or more device types

Note 1 to entry: DtmDeviceType is described by DeviceTypeInfo class.

3.1.28**DtmModuleType**

element of a DTM software supporting one or more device module types

Note 1 to entry: DtmModuleType is described by class ModuleTypeInfo.

3.1.29**DtmBlockType**

element of a DTM software supporting one or more block types

Note 1 to entry: DtmBlockType is described by class BlockTypeInfo.

3.1.30**FDT Object**

object defined by FDT (e.g. Frame Application, DTM Business Logic, DTM User Interface, Communication Channel)

3.1.31**FDT Protocol Annex**

document defining support for a communication protocol for FDT

Note 1 to entry: Examples for such documents are “PROFIBUS protocol annex” and “HART protocol annex”. Within IEC 62453 the standard parts with numbers 3xy define support for communication protocols.

3.1.32**FDT Application Profile Annex**

document defining support for a type of application for FDT

Note 1 to entry: An example for such a document is the “PLC Tool interface” (defined for FDT1.2). Other such documents may be defined at a later time also for FDT2.

3.1.33**fieldbus message**

data in a protocol-specific telegram

3.1.34**Frame Application modal**

prevents user interaction with windows of the Frame Application (see modal window)

3.1.35**hardware platform**

hardware on which FDT software is executed

Note 1 to entry: Different hardware platforms may be supported that are based on different architectures and display formats, for example PC and others.

3.1.36

identification attributes

attributes which describe the identity of a component. These attributes are typically displayed to users or used to validate and ensure compatibility of components

3.1.37

incompatibility

situation where a component is not interoperable or where a component can not replace an other component

3.1.38

instance data

configuration data that resides in the DTM instance

3.1.39

lifetime of DTM instance

time span of executing a DTM BL (from state 'created' till state 'released')

3.1.40

Link

logical relation of a DTM to a physical device (not the communication connection)

3.1.41

modal window

prevents user interaction with all windows of the process

3.1.42

online data

configuration data that resides on the device and can be accessed by communicating with the device

Note 1 to entry: Online data may be a subset of device data (i.e it may be that not all device data is accessible by communicating with the device).

3.1.43

operation

procedure that may involve one or more method calls between FDT Objects

3.1.44

pattern

a standard solution to common problems in software design

3.1.45

platform

combination of hardware platform, target platform and target CLR, that defines the environment for execution of FDT software

3.1.46

project

generic term for the sum of information related to a set of devices

Note 1 to entry: The definition of project is specific for a Frame Application.

3.1.47

proxy object

object which functions as a representative of an other object

Note 1 to entry: The proxy pattern is a often used software design pattern.

Note 2 to entry: IEC 62453-42 uses the proxy pattern for interaction between DTM BL and DTM UI, between DTM BLs, between DTM BL and Communication Channels, for supporting multiple .NET Framework versions in a Frame Application and for providing backward compatibility to FDT1.2.x.

3.1.48 reassign

3.1.48.1 reassign

<DTM replacement>

Change the TypeInfo assigned to a device node from one TypeInfo.Id to the same TypeInfo.Id in a different DTM

3.1.48.2 reassign

<device replacement>

change the TypeInfo.Id to another TypeInfo.Id within the same DTM or to another DTM

3.1.49 reassignment

process of assigning a different DtmType to a device node with assigned DtmType

Note 1 to entry: It is possible that for the previously assigned DtmType already a dataset exists. This dataset should be considered in the reassignment.

3.1.50 replacing installation

installation of a new version of a DTM which replaces a currently installed DTM version

Note 1 to entry: A Frame Application is notified about the installation, but the DtmDeviceTypes do not need to be reassigned. DTM Updates (see update) and DTM Upgrades (see upgrade) replace installations of older versions of the DTMs.

3.1.51 revision

identification of modification of non-FDT components, e.g. device firmware or device hardware

Note 1 to entry: Not all fieldbus specifications supported by FDT and/or device types provide a version identification which allows to derive compatibility statements.

Note 2 to entry: In contrast to a version, revisions require additional fieldbus or device type-specific knowledge to derive compatibility or interchangeability predictions.

3.1.52 SemanticInfo

identifier that provides a reference to semantics defined in a specific context

Note 1 to entry: The reference is provided by the SemanticId, the context is provided by the ApplicationDomain that accompanies the SemanticId.

Note 2 to entry: There may be several semantics provided for an information item, e.g. a parameter may be described in a fieldbus profile as well as in a device profile (e.g. for drives), that is why several semantic infos may be provided for an information item.

3.1.53 set point

target value that an automatic control system will aim to reach

Note 1 to entry: For example a boiler control system may have a temperature set point, which is the temperature the control system aims to attain.

3.1.54**Sibling DTMs**

DTM instances in an FDT Project, which are classified by their relation to the same Communication Channel

3.1.55**surrogate process**

process hosting an object on behalf of client processes

Note 1 to entry: A surrogate process can have other qualities than the client process. E.g. it can be used to load a different .NET Framework.

3.1.56**synchronous operation**

operation that is performed while the object that requested the operation is waiting for the result

3.1.57**target CLR**

common language runtime, which defines the environment for execution of FDT software

Note 1 to entry: An example for target CLR is the CLR 4.0.

3.1.58**target platform**

native data size supported by the machine and operation system, on which the FDT software is executed

Note 1 to entry: IEC 62453-42 defines support for 32-bit and for 64-bit target platforms.

3.1.59**transformer**

component for the translation between FDT1.2.x XML documents and IEC 62453-42 datatypes

Note 1 to entry: Transformers are provided by the FDT Group for each communication protocol specified in an FDT Protocol Annex specification.

3.1.60**update**

process to replace a component with a later (up to date) revision (update revision) that includes error corrections

3.1.61**update revision**

(minor) revision of a component that includes error corrections and small enhancements

Note 1 to entry: In comparison to an upgrade revision an update revision includes no major functional enhancements or new features. An Update Revision shall be backwards compatible to previous revisions of the same component.

3.1.62**upgrade**

process to replace a component with a later revision that includes functional enhancements and/or new features (upgrade revision)

3.1.63**upgrade revision**

revision of a component that includes functional enhancements and/or new features compared to a previous revision of the component

Note 1 to entry: An Upgrade Revision shall be backwards compatible to previous revisions of the same component.

3.1.64

version

an instance of a software product derived by modification or correction of a preceding software product instance (see [33]²)

Note 1 to entry: The format of a version is: Major.Minor[.build[.revision]] for more information see <http://msdn.microsoft.com/en-us/library/hdxyt63s>

Note 2 to entry: Version is used in FDT2 for identification of FDT software components and for corresponding compatibility attributes.

3.2 Abbreviations

For the purposes of this document, the abbreviations given in IEC 62453-1, IEC 62453-2 as well as the following apply.

API	Application Programming Interface
BTM	Block Type Manager
CLR	Common Language Runtime
CLS	Common Language Specification
CSS	Cascading Style Sheet
DCS	Distributed Control System
DD	Device Description
DLL	Dynamic Link Library
DOM	Document Object Model
DTM	Device Type Manager
FA	Frame Application
FDT	Field Device Tool
FDT1.2.x	FDT implementation according IEC 62453-41
FDT2	FDT implementation according IEC 62453-42
GUI	Graphical User Interface
GUID	Globally Unique Identifier (a UUID)
HART® ³	Highway Addressable Remote Transducer
IID	Interface ID
IO	Input / Output
LCID	Locale ID
MSDN®	Microsoft Developer Network
PLC	Programmable Logic Controller
WPF	Windows Presentation Foundation
XDR	XML data reduced
XSL	eXtensible Stylesheet Language
XSLT	XSL Transformations

3.3 Conventions

The conventions for the UML notation used in this document are defined in Annex H.

² Numbers in square brackets refer to the Bibliography.

³ See Annex G.

This document specifies requirements to software. Different levels of requirement may be recognized by the used wording.

Wording	Indicates
'shall', 'has to', 'have to', or 'Mandatory'	No exceptions allowed.
'should' or 'Recommended'	Strong recommendation. It may make sense in special exceptional cases to differ from the described behavior.
'conditional'	Function or behavior shall be provided, depending on defined conditions.
'can' or 'Optional'	Function or behavior may be provided.

Further conventions are:

Convention	Indicates
Note:	Indicates text (in small letters), which does not express requirements, but provides additional information.
<MethodName>	Angle brackets are used to indicate a reference to an asynchronous method NOTE If looking for definition in Annex B: Such methods are implemented as pair of BeginMethodName()/EndMethodName()

Code examples provided in this document are intended for illustration of the described concepts. They should not be used as is. Developers of FDT software should consider where the developed code is applied and design the software accordingly. For exact specification of protocol-specific implementations, refer to the FDT Protocol Annex documents.

4 Implementation concept

4.1 Technological orientation

The .NET Framework is a software framework by Microsoft. The Framework includes a large library and supports several programming language. Programs written for the .NET Framework execute in a software environment, named the Common Language Runtime (CLR). The class library and the CLR together constitute the .NET Framework. The CLR is a Microsoft-specific implementation of the definition provided by ISO/IEC 23271:2012 and ISO/IEC 23270:2006.

The implementation of FDT's client/server architecture defined in this Technical Report is based on the .NET Framework.

This part of IEC 62453 specifies .NET interfaces (what the interfaces are), not the implementation (not the "how" of the implementation) of those interfaces. It specifies the behavior that the interfaces are expected to provide to client applications that use them. The FDT-specification neither specifies the implementation of DTMs nor the implementation of Frame Applications.

Included are descriptions of architectures and interfaces which seemed most appropriate for those architectures. Like all COM implementations, the architecture of FDT is a client-server model where DTMs are the server components managed by the Frame Application.

4.2 Implementation of abstract FDT object model

Figure 2 provides an overview of how the FDT Objects (defined in IEC 62453-2) are implemented in IEC 62453-42 and how their relationship to each other is implemented. The FDT Objects are implemented as .NET objects.

IEC 62453-42 defines a simplification in regard of the implementation of the object model. Within this implementation the only Channels are Communication Channels. Process Channels are mapped to ProcessDataInfo objects. Communication Channels may not have User Interfaces. The objects “Project” and “Host Channel” are considered as Frame Application-specific implementations and are not defined within this Technical Report.

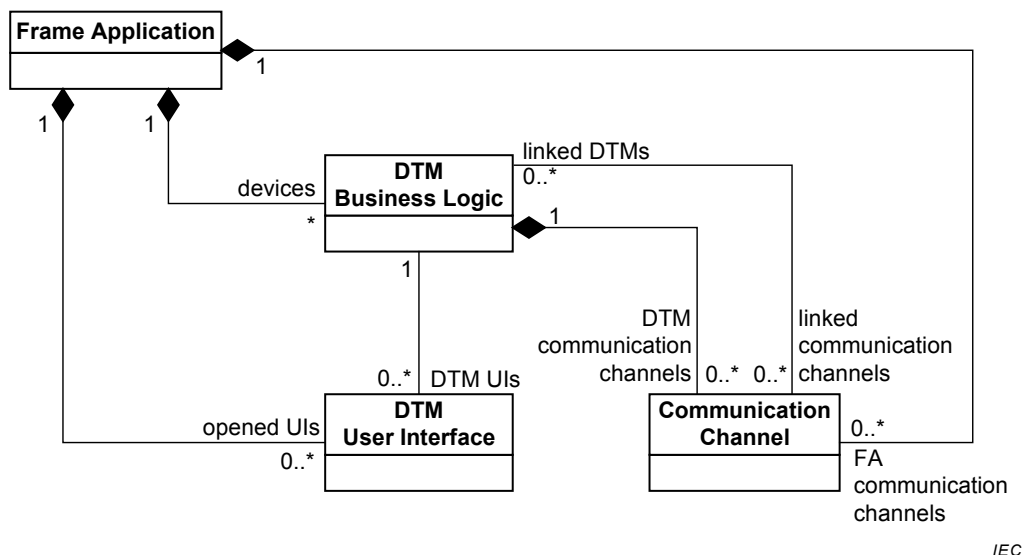


Figure 2 – IEC 62453-42 Object Model

If the Frame Application is a distributed software system, the Frame Application is responsible to organize the instantiation of the objects (based on a vendor-specific implementation).

4.3 FDT Frame Application (FA)

A Frame Application is the runtime environment for the DTMs and provides interfaces which enable the DTM Business Logic and the DTM User Interfaces to interact with its environment. In addition, the Frame Application manages the interaction between the DTM Business Logic and the DTM User Interface by providing a standard messaging interface (see Figure 3).

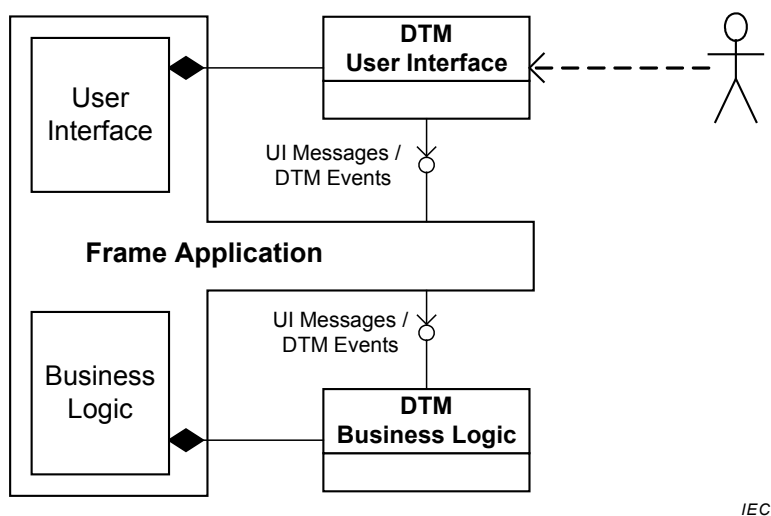


Figure 3 – Frame Application

The messaging interface is used for the transport of DTM-specific messages and events. Contents and format of the messages are proprietary and not understood by the Frame Application.

The Frame Application User Interface represents all functions of the frame related to user interface. The Frame Application Business Logic represents all frame functions related to business logic. Both are not specified by this Technical Report, but are implementation specific parts of the Frame Application (e.g. functional parts or structural parts). These two parts can be comprised in one single application or in separated applications, for example in a server and client application.

Frame Applications can have no, one or multiple Frame Application User Interfaces.

The Frame Application Business Logic part is responsible to execute the DTM Business Logic. It provides services which enable the DTM Business Logic to:

- persist data in the Frame Application persistence storage (see 4.11.1),
- communicate with associated device,
- request displaying of further user interfaces (e.g. user dialogs, additional DTM User Interface),
- browse the FDT topology and interact with other DTMs,
- inform the Frame Application regarding events (error / trace messages, progress etc.),
- interact with the DTM User Interface.

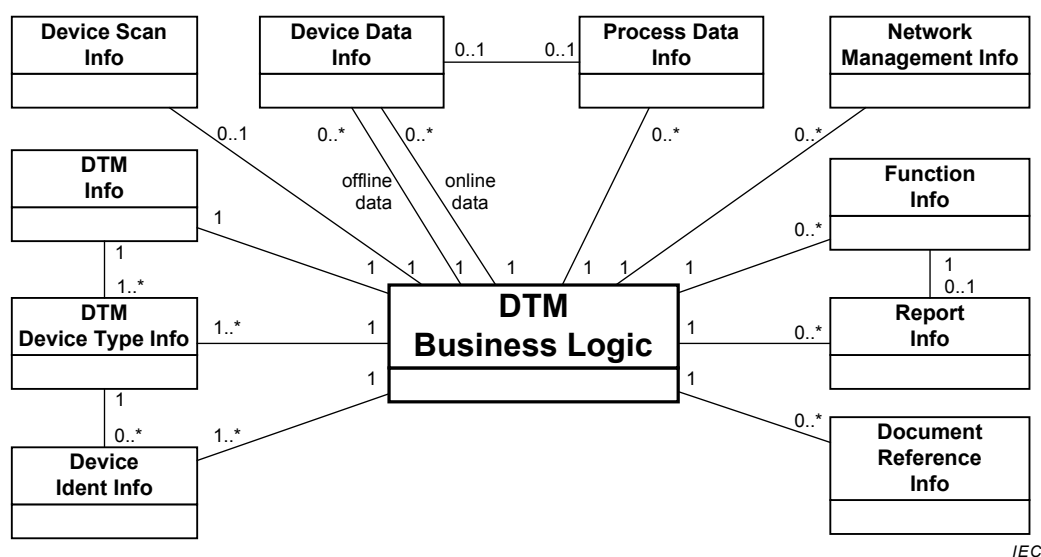
The Frame Application user interface part makes the DTM services available to the users, for example the functions and user interfaces supported by a DTM (see 4.5). It hosts the DTM User Interfaces as part of its own user interface and provides services to:

- interact with the DTM Business Logic (see 4.4)
- request displaying of further user interfaces (e.g. user dialogs, additional DTM User Interface)
- browse the FDT topology and interact with other DTMs
- inform the Frame Application regarding events (error / trace messages, progress etc.)

4.4 DTM Business Logic

4.4.1 General

The Frame Application interacts with the DTM Business Logic through defined interfaces (see Annex B). Figure 4 shows the information objects which are used by the interface definitions implemented by the DTM Business Logic.



NOTE Since this subclause describes the general concept of FDT, the actual implementation in a DTM may differ. (E.g. if a network protocol uses network management, it will be mandatory to provide network management information.)

Figure 4 – DTM Business Logic

4.4.2 Implementation of DTM, DTM Device Type, and Device Ident Info

In order to increase performance in creation of libraries and selection of DTMs, the service `GetTypeInformation` shall be provided by a separate class (the so called `DtmInfoBuilder` class). The `DtmInfoBuilder` is installed together with the DTM. It implements the same interface to provide information as defined for a DTM (see `IDtmInformation` in Annex B). The main advantage of `DtmInfoBuilder` is, that it can be used without instantiating the DTM.

NOTE 1 By using the DtmInfoBuilder it is possible to adapt the available TypeInfos depending on various conditions (e.g. DD-Interpreter DTMs may provide TypeInfos depending on installed DD files).

Figure 5 shows the information datatypes that are provided by a DTM in order to support these Frame Application function and how this info datatypes are used to describe the supported devices (see 7.4 and 7.5 for detailed description).

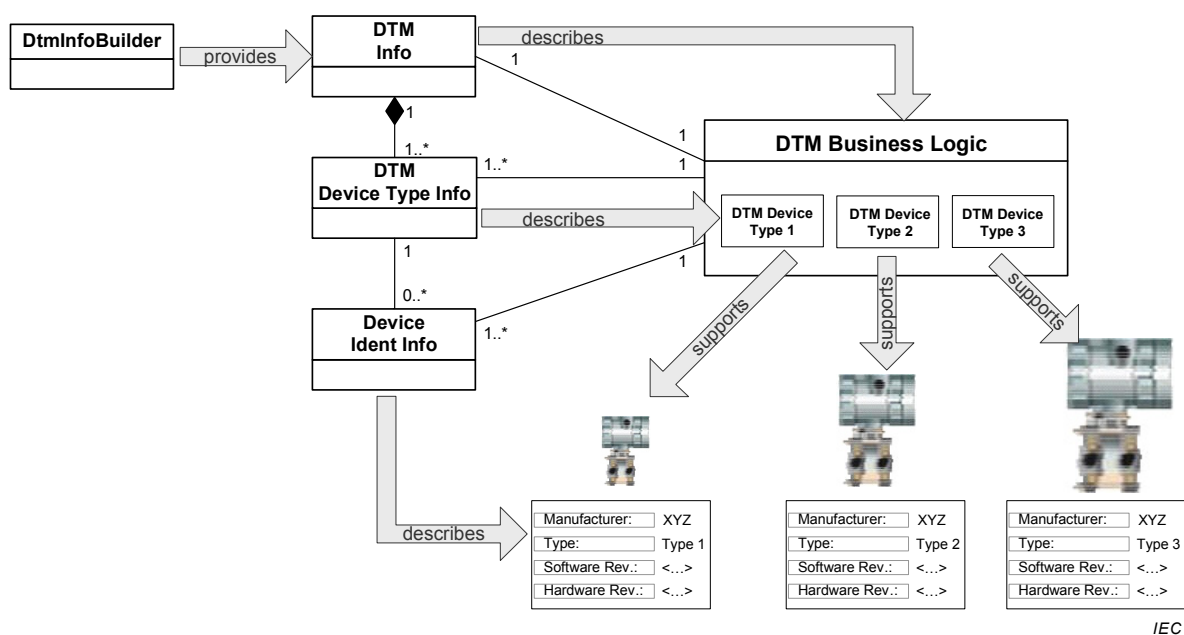


Figure 5 – DTM, Device Type and Device Ident Info

NOTE 2 Since this subclause describes the general concept of FDT, the actual implementation in a DTM may differ.

The representation for a particular physical device type within the DTM is called DTM Device Type. A DTM may provide one or more DTM Device Types. The concrete design and implementation of the DTM Device Types is not in scope of FDT.

Information about physical device types, which can be handled by the DTM Device Types is returned by the service `IDtmInformation:GetDeviceIdentInfo` (see definition of `IDtmInformation` in Annex B). Such information is for example manufacturer, type, hardware and embedded software version of the device. The DTM may even return regular expressions for some specific device identification elements to signal that the DTM Device Type can be used for all devices for which the expression 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 service `IDtmInformation:GetDeviceIdentInfo` is fieldbus-specific and therefore defined by the document describing the protocol profile integration in FDT2. 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.

4.4.3 Implementation of DTM device parameter access

A DTM supports interfaces to read and write device parameters stored in the DTM instance data (offline data) and directly in the connected device (online data). This data is represented by `DataInfo` (see 4.12.1 for detailed description).

A DTM has to expose a defined set of device parameters which are publicly available (see 4.12 for detailed description). Parameters are provided in a bus neutral structure allowing their use without knowledge of the fieldbus protocol.

4.4.4 Process Data Info

Process data provided by devices (e.g. IO signals) are integrated into the functional planning of the control system. The process data related information for the integration of the device into the control system like datatype, signal direction, engineering units, and ranges is provided by the DTM Business Logic for each DTM Device Type (Figure 6), but may also depend on the device instance configuration.

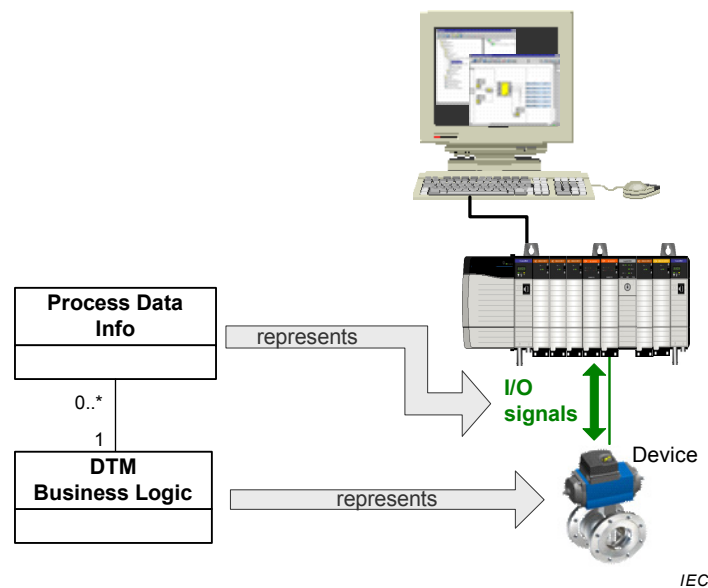


Figure 6 – Process Data Info

The process values provided by a device, both the number and type, may depend on the configuration of the DTM. Thus the number of available Process Data Info objects may also depend on the device configuration. A Frame Application is able to inform the DTM that further configuration changes shall be prohibited because the process data is already integrated into functional planning of the control system. In this case the DTM shall not allow any changes which will affect the definition of the available process values.

The process information is protocol-specific. Each FDT Protocol Annex defines a derived class defining which information shall be contained. However, the properties in the base class Process Data Info provide common information (e.g. IO signal name, tag etc.) in a protocol-independent format to enable Frame Applications without protocol-specific knowledge to integrate the process variables into the system (see 7.11.1 for detailed description).

If the process data value is also available as Device Data Info, then the corresponding element is referenced by the Process Data Info element (see IOSignalRefs in 7.11.1).

A Process Data Info element may have a relation to a Communication Channel. This is a typical case for a Gateway DTM for a remote IO (see IEC 62453-2:–, 4.2.3.2.4). The relation is represented within the Process Data Info element (see IOSignalRef in 7.11.1).

NOTE The Process Data Info replaces Process Channel as defined in FDT1.2.x.

4.4.5 Diagnostic Data Info

A DTM provides a service to retrieve the status of the related device. The status is encoded in a protocol independent way, according to [7].

4.4.6 Network Management Info

The DTM supports an interface to read and write network management information which can for example be used for address management and bus master configuration (see 8.11.1).

The Network Management Info is protocol-specific. It may contain device bus-address, tag and additional bus-specific configuration settings. Each FDT Protocol Annex defines a derived class defining the protocol-specific record. However, the base class Network Management Info provides common information (e.g. bus-address, tag) in a protocol-independent format to enable Frame Applications without protocol-specific knowledge to use it (see 7.13 for detailed description).

4.4.7 Function Info

Each DTM may provide a set of functions comprised by

- DTM User Interfaces,
- DTM functions without DTM User Interface (CommandFunctions) and
- references for external documents.

The Function Info object provides the information about the functions such as name, status (enabled / disabled), etc. (see 7.14 for more detailed description).

A function should provide printable information. A Frame Application may call the documentation interface (see [4], 7.2.7.1) of the DTM to retrieve printable information. This interface returns a corresponding Report Info object which holds the printable information. The Report Info object may indicate the relation to the function.

4.4.8 Report Info

If a DTM provides instance data and/or online data, then the DTM shall implement a device type-specific reporting of the provided data for documentation and archiving purposes. The DTM BL may implement different types of reports that each cover a distinct subset of the instance or online data of a device.

The Report Info exposes a list of report types supported by a DTM (see 7.19). The list may be grouped. The list is static over the lifetime of a DTM instance, there are no dependencies on the current application context.

4.4.9 Document Reference Info

A DTM may provide references to external documents, which are displayed by the Frame Application. These references may be provided as part of the TypeInfo (as static information) or as part of the Function Info (may change dynamically). The Frame Application may provide an own viewer for the documents or rely on the condition that external software is installed for the document type. It is recommended, that DTMs use common document formats (recommended formats are PDF, CHM and HTML). Otherwise DTMs should provide the necessary viewers.

4.5 Implementation of DTM Functions

4.5.1 DTM User Interface

A DTM User Interface (DTM UI) may be a graphical control which is integrated into the user interface of the Frame Application or a proxy object handling the interaction with an external program provided together with the DTM.

A DTM UI may be modeless or modal. A modal DTM UI behaves modal only in respect to the DTM instance. A DTM instance should expect that the modal DTM UI blocks only activity in regard to its own UI (DTM instance modal) and that other UI (Frame Application, other DTMs) still may trigger actions. Modal DTM UIs should be used as sparingly as possible.

A Frame Application may also implement Frame Application modal behavior for modal DTM UIs.

The contents and layout of the DTM User Interfaces is device-specific, but shall follow the DTM Style Guide [6].

NOTE 1 A DTM may provide DTM User Interface for different platforms (e.g. different display formats e.g. PC, mobile device etc.). Which platform is supported by a DTM User Interface is described by corresponding UI Function Info element returned by the DTM (see 7.14). The FDT specification will define which platforms shall be supported.

NOTE 2 The intention is for IEC 62453-42 to allow later extensions in regard to User Interface technologies and platform support.

4.5.2 Function access control

The Frame Application can restrict the invocation of functions provided by a DTM, disable actions (buttons) of the DTM UI (e.g. Apply) and restrict DTM transition to certain states (e.g. restrict transition to state Online by not calling the method EnableCommunication()).

The Frame Application will get the list of the functions provided by the DTM in IFunction.FunctionInfo property. The DTM shall expose all functions available in the DTM in all modes. When the functions are not applicable for the current mode of operations, FunctionItem.Enabled property will indicate that. A DTM shall expose always the same set of functions for a device node (even if the DTM instance was terminated and loaded again). The number and type of listed functions shall not change, but the status of the functions in regard to availability and visibility may change. Frame Application should hide the functions with Hidden property set to TRUE.

The DTM shall indicate when the state of Enable is changed, by IFunction.FunctionsChanged event.

When a user interface for the DTM is invoked, the DTM shall not allow switching the context from within the DTM user interface.

For Example:

If a Diagnostic Function is invoked by the Frame Application, the DTM will present the diagnostic information in the user interface. The user should not be allowed to invoke the configuration screen from within the DTM user interface for the diagnostic function without the permission from the frame. The user shall be able to invoke the configuration screen from the functions exposed to the Frame Application if the permissions in the Frame Application allow it.

If the MainOperation Function is invoked by the Frame Application, the DTM will present all available functions within the user interface. The user may invoke the Configuration screen from within the DTM user interface as well as the Diagnostic function or any other integrated function.

4.5.3 Handling of standard UI elements in modeless DTM UI interfaces

Modeless DTM UIs shall delegate the presentation and handling of their standard dialog elements to the Frame Application. The standard dialog elements are:

- actions with standardized semantics (Apply / Close / Online Help) (see Action Area in [6]) and
- DTM UI-specific status information (data source, summary parameter modification state, UI operation mode, activation of service mode) (see Status Bar in [6]).

To ensure a consistent user interface appearance between the different DTM vendors, a DTM UI may delegate presentation and handling of additional DTM application-specific actions to the Frame Application. Nonetheless DTM UIs are allowed to implement non-standard dialog actions within their own UI area (see Application Area in [6]).

The set of standard dialog actions and their respective semantics is fixed. However, the availability of these actions may change at any time depending from the internal state of the DTM UI. The set of application-specific actions including their individual availability is not fixed. A DTM UI may add, remove, rename, enable or disable application-specific actions at any time depending from its individual requirements. A DTM UI shall inform a Frame Application whenever the availability of its standard actions or the set or availability of its custom actions changes (see events IStandardActions.StandardActionItemSetChanged and IApplicationSpecificActions.ApplicationSpecificActionItemSetChanged in Annex B).

A Frame Application may use dedicated UI elements, e.g. button controls, to provide direct access to the standard dialog actions, as well as indirectly invoke them in the context of user interaction with other Frame Application UI elements. A Frame Application shall always show all custom actions exposed by a DTM UI with dedicated UI elements. These shall be unambiguously associated with the DTM UI as described in the DTM Style Guide [6].

4.5.4 Command functions

Command functions are used to execute actions (commands) either on the DTM BL or within the context of the graphical user interface. Command functions in context of the DTM BL shall not have a GUI, but UI Command functions may show a GUI (see 6.4).

A Command function may have parameters. The information about the parameters, which is provided by the DTM BL, may include default values of the parameters. The actual parameter values are passed when the Command function is executed.

4.6 User management

4.6.1 General

FDT does not define a standard system for user management. The user management is part of product-specific definitions and may be implemented differently for different Frame Applications. However it is still necessary to define a common handling for access permissions, access rules and how components from different vendors communicate information regarding access permissions.

4.6.2 Multi-user access

Some Frame Applications provide multi-user capability. Such a system provides access for multiple users at the same time and may be distributed over several computers. This specification considers the distributed environment as one Frame Application. The Frame Application and the DTM are equally responsible to provide the multi-user access and to ensure consistency of data.

If, within one Frame Application, multiple users access the same device or the same device dataset, the Frame Application shall start a separate DTM instance for each user. All these DTM instances shall have same DTM type, shall be instantiated for the same DtmDeviceType and for the same physical device. Each DTM instance manages a separate instance dataset. These instance datasets are synchronized by means of the persistent dataset (see 4.11.5).

NOTE An example architecture for multi-user scenarios is found in 11.4.

4.6.3 User levels

4.6.3.1 Introduction

DTMs may be integrated in different Frame Applications, which may have varying requirements to restrict visibility and accessibility of device and persistent data, for example for plant safety reasons or to present a customized view to the user. The grade of restriction varies with the types of users supported by a system. Examples for users requiring data access restrictions are:

- a user assigned to observe a plant shall not have access to calibration-specific device parameters and consequently shall not see related DTM functionality,
- a device commissioning specialist needs to have access to calibration data and functions,
- a user assigned to operate a plant shall be able to change (write) set point values and be offered appropriate functionality while a user assigned to observe a plant is not allowed to execute such changes.

4.6.3.2 Access Control Concept

FDT uses a role based access control concept. A Frame Application initializes a DTM Business Logic and the associated User Interface with the same FDT-specific user level (see definition of `IDtm.Init()` and `IDtmUIFunction.<Init()>` in Annex B). The user level is immutable over the lifetime of the Business Logic/User Interface instances.

In terms of access control, every actor in an FDT system may have one of the following three user levels at the interface of a DTM (Table 1):

Table 1 – FDT User levels

User Level Name	Description
Observer	This user level stands for an actor that observes the current process only.
Expert	This user level stands for an actor who has to execute specific use cases, e.g. operation use cases (operation expert) or device maintenance use cases (maintenance expert). This user level allows the Frame Application to configure access and privileges.
Engineer	This user level stands for an actor that has to do the plant planning, device configuration/parameterization and plant maintenance.

The user levels allow a stepwise extension of permissions. The Observer typically has a minimum permission set, the Expert has an intermediate permission set (which is configured by the Frame Application) and the Engineer has a full permission set.

The Expert user level may be considered as a super-set of the actors “Operator” and “Maintenance” as defined in IEC 62453-2. Since a Frame Application may configure the access permissions for the Expert, it is possible to apply fine grained permissions, that can be adapted to different application scenarios.

NOTE For an explanation of the fundamental user levels and use cases that were considered for the design of the FDT specification see Annex A. A Frame Application may support only a subset of these use cases or additional use cases not defined in the annex.

According to the role set by the Frame Application, the DTM Business Logic and User Interface shall control access to device and persistent data (see definition of interfaces `IInstanceData` and `IDeviceData` in Annex B) as well as adapt its user interface appearance. This includes to hide some data or display it as read-only, but also to partially disable DTM-specific functionality (see definition of `IFunction.FunctionInfo` in Annex B), if it requires data access rights that are not associated with the specified user level.

It is mandatory for a DTM to implement a safe and read-only usage for the “Observer” user level. It is also mandatory for a DTM to implement unlimited usage for the “Engineer” user level. It is optional for a DTM to implement configurable custom usage for “Expert” user level. If “Expert” level is not implemented by a DTM but is set by the Frame Application, the DTM shall use the behavior for “Observer” user level.

Table 2 gives an overview about the user interfaces and functions that are expected to be available for the individual user levels. The data access rights should be defined to allow for the execution of these use cases.

The assignment of roles to individual users is Frame Application-specific. The Frame Application may implement an own user management sub-system or use fixed user levels.

Table 2 – Role dependent Access Rights and User Interfaces for DTMs

Use Case	Sub Cases	User level		
		Engineer(M)	Expert(O)	Observer(M)
System Planning	Network Management	If a DTM implements these use cases, it shall expose all related commands and user interfaces.	If the DTM implements these use cases and supports the user role “Expert”, it shall allow Frame Application to configure access to the exposed data, commands and user interfaces	-*)
	Busmaster Configuration			-
	Channel Assignment			-
System Generation	Network Management			O {r}
	DTM matching			O {r}
Device Configuration	-			-
Simulation (Force)	-			-
Offline Operation	Offline Parameterization			-
	Persistent Data Comparison			-
Online Operation	Online Functions (reset + other functionality that requires online device connection)			If a DTM implements these use cases, it shall expose all related commands and user interfaces.
	Online Parameterization	-		
	Calibration	-		
	Device/Persistent Data Comparison	-		
	Adjust SetValues	O {r}		
	Upload	-		
	Download	-		
Bulk Operation	Upload	-		
	Download	-		
Online View	Network Scan	O {r}		
	Online Status	M {r}		
	Online Trend	M {r}		
	Device Identification	M {r}		
	Online View Parameter Set	O {r}		
Report Generation	-	M {r}		
Device-specific Operations	Device vendor-specific (or extended) DTM functions after DTM/Device-specific OEM Service login	-		
M Mandatory (if a DTM implements this use case, it shall expose all related commands and user interfaces in the specified user level)				
O Optional (a DTM may expose the related commands and user interfaces in the specified user level)				
r User level shall have read access to all data related to the use case				
w User level shall have write access to all data related to the use case				
- Use case not supported in this user level (DTM shall not expose any related commands/user interfaces)				
*) A DTM shall allow all user levels to set the device address in the DTM with the method SetAddressInfo().				

4.6.3.3 Frame Application configured access control

It is very difficult and may even be impossible for the DTM vendor to provide correct access control settings for all occasions. The data, which can be accessed, and the functions, which can be used, are changed by the user; depending on where the DTM is used or what is the operational phase of the plant. Here are some examples:

- a) The same user may have different permissions for the same device controlled by the same DTM when the device is connected in the plant or in the instrument shop. In the instrument shop, the user may have all equipment to calibrate the instrument and the corresponding privileges should be granted. Little or no changes may be allowed when the instrument is connected later to the actual running control system.
- b) The same user may have full control when the plant is being engineered, but the changes to the device may be significantly restricted, when the plant is in running state.
- c) The same group of users may have different permissions for different instruments – some of the maintenance personell may be trained to work with transmitters, other may be specialized in valve maintenance.
- d) In a small application one person may be responsible for the entire application and he may have unlimited access to all device maintenance procedures, but in a big application, often the access is controlled according to the individual experience of the technical staff.

To address the different cases, an “Expert” user level is provided. When the DTM supports the “Expert” user level, the data which can be accessed and the function which can be invoked are restricted by the Frame Application depending on the rules in the plant, on the operational phase, the individual user or team experience and other factors.

The Frame Application can use the Expert user level to create additional levels of access to the DTM data and functions for individual user or for a group of users. For example, when the access control is configured for the Operation Expert, the Frame Application may enable the access to Set Point Values, to the Tuning parameters and to Diagnostic functions. In another example, the Frame Application may enable the access to the Calibration parameters, to the calibration functions and to the Online Parameter View when the Device DTM is invoked in the instrument shop environment.

NOTE Be aware that the user level “Operator” as defined in FDT1.x specification is not supported in IEC 62453-42. The term Operator in this document is used to describe an expert for plant operations.

4.7 Implementation of FDT and system topology

4.7.1 General

IEC 62453-42 differentiates two topology views: logical topology and physical topology (see Figure 7).

A logical topology is created by a hierarchy of DTMs. Child DTMs are connected to Parent DTMs via the Communication Channel of the Parent DTM. A Parent DTM may have multiple children. This relation is managed by the Parent DTM. This means that a Parent DTM knows all its Child DTMs.

A Child DTM may be assigned to multiple parents (e.g. if different network paths may be used to access a device). A Child DTM is not notified if it is assigned to a Parent DTM, but it may request a list of parents from the Frame Application by using the method `ITopology.GetParentNodes()`.

A Child DTM can use only one communication path at a time to access the respective device. The Parent DTM providing this communication path will be marked by the Frame Application as ‘primary parent’.

This means, that the logical topology describes the logical relations between the devices on an abstraction level that supports managing the communication between DTMs and devices.

A physical topology is created by defining physical connections between DTMs. Connections are defined between Ports of the DTMs. This means the physical topology describes the actual hardware installation. The connections are managed by the Frame Application. It is possible to use these connections for representation of all kind of network structures.

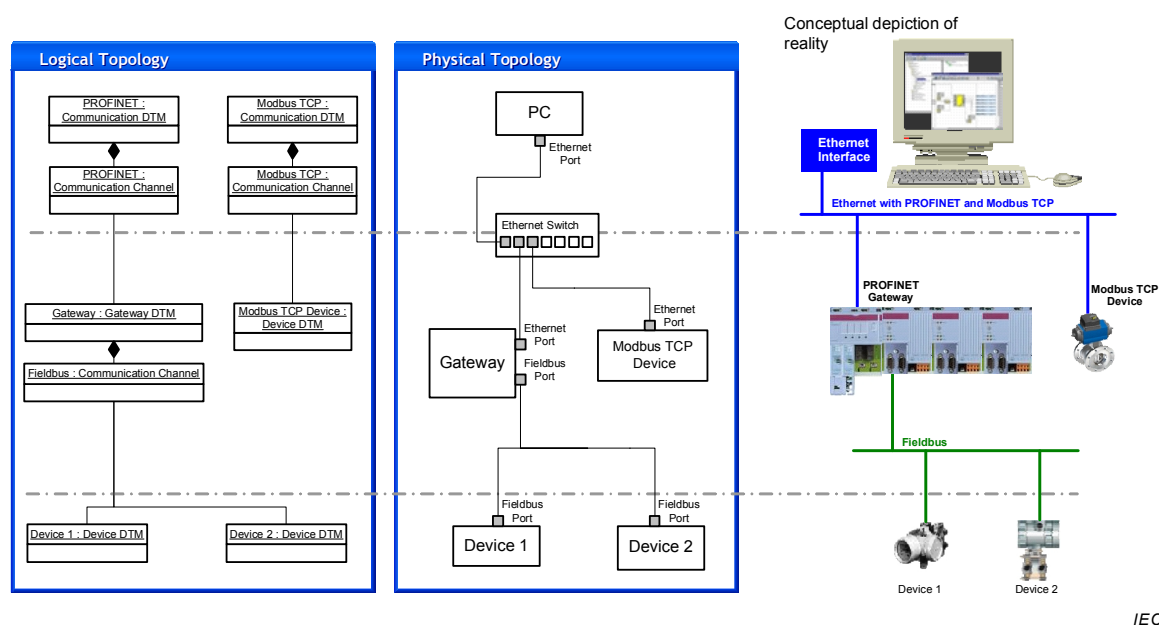


Figure 7 – Logical topology and physical topology

4.7.2 Topology management

4.7.2.1 Logical topology

The Frame Application is responsible for managing the logical topology – it is mandatory to support the logical topology. That means the Frame Application shall organize the routing of data for accessing a device in the plant. Some Frame Applications may require user interactions; others may support automatic operations such as topology import or fieldbus scanning. The sum of all links between DTMs according the logical topology is called FDT topology and further described in IEC 62453-2.

A DTM exposes all required information (see 4.4.2) which enables the Frame Application (and the user) to choose the appropriate DTM for a device, for example name, vendor, version of supported device types and corresponding identification properties.

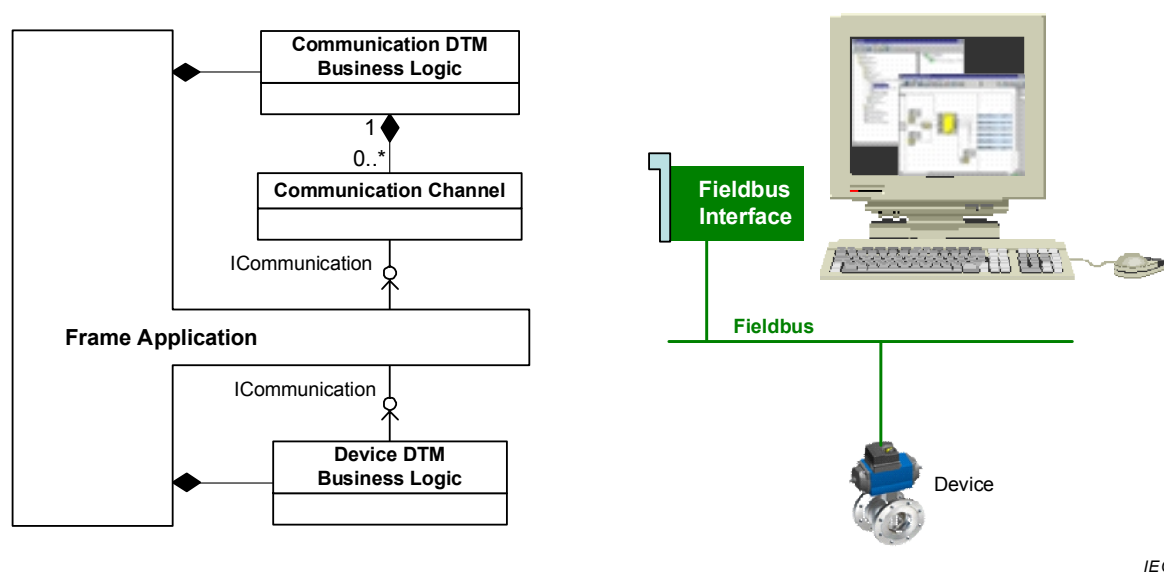


Figure 8 – FDT and logical topology

As shown in Figure 8 a Communication Channel is used as the linking element between Communication DTM and Device DTM. The Communication Channel 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 shall be made by the Communication Channel. The Frame Application has to call the method `ISubTopology.<ValidateAddChild()>` (see definition in Annex B) before link is created. The Communication Channel shall at least check whether the required network protocol of the DTM to be linked fits to its own supported protocol. If this is not the case, then the linking shall be rejected. In addition, the Communication Channel may perform further checks, for example whether the number of linked DTMs exceeds a limit.

Neither the Communication Channel (or corresponding DTM) nor the linked DTM shall need to manage topology information in order to access the respective physical device. The Frame Application supports to request topology information by the methods `ITopology.GetParentNodes()`, `ITopology.GetSiblingNodes()` (and `ITopology.GetChildNodes()` (for all see definition in Annex B).

The rules for identification of DTMs and devices are described in 4.10.

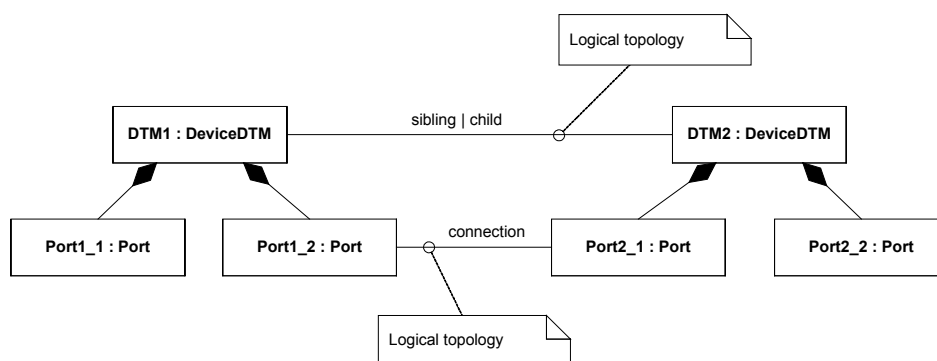
For some communication protocols the order of the devices linked to the network affects the configuration of the network itself. This order is defined when inserting the corresponding DTM into the logical topology (`ITopology.BeginAddChild()`) and can be modified later via `ITopology.BeginRepositionChild()`. The Frame Application always has to maintain this order when returning collections of DTMs in `ITopology.GetChildNodes()` and `ITopology.GetSiblingNodes()`.

If the Frame Application provides a view on the channel, it shall show all Child DTMs in their respective order. In this view, the Frame Application shall allow the user to insert a new DTM at a specific position between the existing Sibling DTMs or to change the position of an existing DTM in regard to its Sibling DTMs.

4.7.2.2 Physical Topology

Advanced topology management requires the additional planning of cable bound or wireless connections between devices. This capability is provided by the Physical Topology.

The management of the Physical Topology is the responsibility of the Frame Application. It is optional for a Frame Application to support the Physical Topology. It is mandatory for a DTM to expose all information which is required to determine whether a physical connection is possible or not. The Physical Topology may not have dependencies to the Logical Topology and shall be handled separately as shown in Figure 9.



IEC

Figure 9 – DTMs and physical topology

The connections are managed by the Frame Application. Information regarding connections may be accessed with the interface `IPhysicalTopology`. See Annex B for a detailed description.

4.7.2.3 Communicating and non-communicating devices

An automation system integrates communicating devices, as well as devices which do not communicate and therefore are not configurable via communication (e.g. power supplies and other network infrastructure elements). Information about such devices may be essential during the planning phase of the communication system and can be used to verify integrity of the network, for instance in regard to bus power overload, communication distance limitations, validity of the design (e.g. correct termination). In order to integrate such devices in an FDT-based system, a DTM may be provided to represent such a 'passive device'. The DTM provides information about the device/equipment to the Communication DTM, which is capable to use this information.

Different protocols require specific information to be provided and may have different limitations to be enforced. Information provided from DTMs for passive devices has to be provided in a standard way and format (in Network Management Info), so that different Communication DTMs can use it in a standard way. DTMs for communicating devices may need to provide similar information. Protocol-specific extensions have to define the information provided by the communicating devices and by the non-communicating devices and also how this information is used. For example, Communication DTMs for a bus powered protocol can use information from non-communicating devices (defining the power source) and communicating devices (defining the power consumption) in order to balance the power on the network.

4.7.3 Data exchange between Frame Applications

The interaction between different Frame Applications is not in the scope of FDT, but the FDT specification defines datatypes (see `TopologyImportExport` definition in Annex B) which can be used for this purpose. These datatype classes may be used by one Frame Application to export the FDT topology information to an XML file which then can be imported in another Frame Application.

4.8 Implementation of Modularity

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

- monolithic DTM with topology description in `NetworkDataInfo`, `ProcessDataInfo`, `ProcessImageInfo`, `DataInfo` and `CommunicationChannelInfo`
- Module DTM, and
- BTM.

4.9 Implementation of FDT communication

4.9.1 Handling of communication requests

In order to optimize the communication, the interface of a Communication Channel allows passing multiple transaction requests in one call to `<CommunicationRequest>` (as a list).

The Communication Channel is expected to process the transaction requests in the order they are provided in the list. The results of the transaction requests may be passed back to the client of Communication Channel sequentially as part of the Progress callback (partial results) and the complete result shall be passed back at the end of the `<CommunicationRequest>` according to the extended `AsyncResult` pattern (see 5.6.7.2).

The relation between communication requests and communication responses can be managed by the `IAAsyncResult` handle that is passed to a client in the call to