



BSI Standards Publication

Field device tool (FDT) interface specification

Part 41: Object model integration profile – Common
object model (IEC TR 62453-41:2016)

National foreword

This Published Document is the UK implementation of CLC/TR IEC 62453-41:2019. It is identical to IEC TR 62453-41:2016. It supersedes PD CLC/TR 62453-41:2009, which is withdrawn.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 02869 0

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 30 April 2019.

Amendments/corrigenda issued since publication

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

English Version

**Field device tool (FDT) interface specification - Part 41: Object
model integration profile - Common object model
(IEC TR 62453-41:2016)**

Spécification des interfaces des outils des dispositifs de
terrain (FDT) - Partie 41: Profil d'intégration des modèles
objet - Modèle objet commun
(IEC TR 62453-41:2016)

Field Device Tool (FDT)-Schnittstellenspezifikation - Teil 41:
Profil zur Integration des Objektmodells - Allgemeines
Objektmodell (COM)
(IEC TR 62453-41:2016)

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 62453-41:2019) consists of the text of the IEC TR 62453-41:2016 prepared by 65E: "Devices and integration in enterprise systems", of IEC technical committee 65: "Industrial process measurement, control and automation".

This document supersedes CLC/TR 62453-41:2009.

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 61784	series	Industrial communication networks - Profiles	EN 61784	series
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.....	12
INTRODUCTION.....	14
1 Scope.....	15
2 Normative references.....	15
3 Terms, definitions, abbreviations and conventions.....	15
3.1 Terms and definitions.....	15
3.2 Abbreviations.....	16
3.3 Conventions.....	16
4 Implementation concept.....	17
4.1 Technological orientation.....	17
4.2 Implementation of abstract FDT object model.....	17
4.2.1 General.....	17
4.2.2 FDT Frame Application (FA).....	17
4.2.3 Device Type Manager (DTM).....	18
4.2.4 Presentation object.....	19
4.2.5 FDT-Channel object.....	19
4.3 Object interaction.....	19
4.3.1 Parameter interchange via XML.....	19
4.3.2 Examples of usage.....	21
4.4 Implementation of DTM data persistence and synchronization.....	23
4.4.1 Persistence overview.....	23
4.4.2 Persistence interfaces.....	24
4.5 DTM state machine.....	24
5 General concepts.....	26
5.1 General.....	26
5.2 Overview of task related FDT interfaces.....	26
5.3 Return values of interface methods.....	29
5.4 Dual interfaces.....	29
5.5 Unicode.....	29
5.6 Asynchronous versus synchronous behavior.....	29
5.7 ProgIds.....	30
5.8 Implementation of DTM, DTM device type and hardware identification information.....	30
5.8.1 Device identification.....	30
5.8.2 Protocol-specific transformation style sheet (xsl).....	33
5.8.3 Semantic identification information.....	33
5.8.4 Device assignment.....	33
5.8.5 Regular expression specification.....	34
5.9 Implementation of slave redundancy.....	34
5.9.1 General.....	34
5.9.2 Topology import/export.....	35
6 Implementation of FDT services: FDT interfaces.....	35
6.1 Overview of the FDT interfaces.....	35
6.2 FDT objects.....	35
6.2.1 FDT object model.....	35
6.2.2 Availability of interface methods.....	38

6.3	Device Type Manager	42
6.3.1	Interface IDtm	42
6.3.2	Interface IDtm2	51
6.3.3	Interface IDtmActiveXInformation	52
6.3.4	Interface IDtmApplication	54
6.3.5	Interface IDtmChannel	56
6.3.6	Interface IDtmDocumentation	57
6.3.7	Interface IDtmDiagnosis	58
6.3.8	Interface IDtmImportExport	60
6.3.9	Interface IDtmInformation	62
6.3.10	Interface IDtmInformation2	63
6.3.11	Interface IDtmOnlineDiagnosis	64
6.3.12	Interface IDtmOnlineParameter	65
6.3.13	Interface IDtmParameter	68
6.3.14	Interface IFdtCommunicationEvents	69
6.3.15	Interface IFdtCommunicationEvents2	72
6.3.16	Interface IFdtEvents	73
6.3.17	Interface IDtmHardwareIdentification	76
6.3.18	Interface IDtmSingleDeviceDataAccess	78
6.3.19	Interface IDtmSingleInstanceDataAccess	81
6.4	DTM ActiveXControl	83
6.4.1	Interface IDtmActiveXControl	83
6.4.2	Init	83
6.4.3	PrepareToRelease	84
6.5	FDT Channel	85
6.5.1	Interface IFdtChannel	85
6.5.2	Interface IFdtChannelActiveXInformation	88
6.5.3	Interface IFdtCommunication	90
6.5.4	Interface IFdtChannelSubTopology	97
6.5.5	Interface IFdtChannelSubTopology2	101
6.5.6	Interface IFdtChannelScan	101
6.5.7	Interface IFdtFunctionBlockData	103
6.6	Channel ActiveXControl	105
6.6.1	Interface IFdtChannelActiveXControl	105
6.6.2	Interface IFdtChannelActiveXControl2	106
6.7	Block Type Manager	107
6.7.1	Interface IBtm	108
6.7.2	Interface IBtmInformation	109
6.7.3	Interface IBtmParameter	109
6.8	BTM ActiveXControl	110
6.8.1	General	110
6.8.2	Interface IBtmActiveXControl	110
6.9	Frame Application	111
6.9.1	Interface IDtmEvents	111
6.9.2	Interface IDtmEvents2	120
6.9.3	Interface IDtmScanEvents	121
6.9.4	Interface IDtmAuditTrailEvents	123
6.9.5	Interface IFdtActiveX	125
6.9.6	Interface IFdtActiveX2	126

6.9.7	Interface IFdtBulkData	129
6.9.8	Interface IFdtContainer	131
6.9.9	Interface IFdtDialog.....	134
6.9.10	Interface IFdtTopology	135
6.9.11	Interface IDtmRedundancyEvents.....	141
6.9.12	Interface IDtmSingleDeviceDataAccessEvents.....	142
6.9.13	Interface IDtmSingleInstanceDataAccessEvents	145
6.9.14	Interface IFdtBtmTopology	146
7	FDT sequence charts.....	147
7.1	DTM peer to peer communication	147
7.1.1	General	147
7.1.2	Establish a peer-to-peer connection between DTM and device	147
7.1.3	Asynchronous connect for a peer-to-peer connection	147
7.1.4	Asynchronous disconnect for a peer-to-peer connection.....	148
7.1.5	Asynchronous transaction for a peer-to-peer connection	148
7.2	Nested communication	149
7.2.1	General	149
7.2.2	Generate system topology.....	150
7.2.3	Establish a system connection between DTM and device	152
7.2.4	Asynchronous transaction for a system connection	153
7.3	Topology scan.....	154
7.3.1	Scan network	154
7.3.2	Cancel topology scan	155
7.3.3	Provisional scan result notifications	156
7.3.4	Scan for communication hardware	157
7.3.5	Manufacturer-specific device identification.....	158
7.4	Registration of protocol-specific FDT schemas.....	160
7.5	Configuration of a fieldbus master	162
7.6	Starting and releasing applications	163
7.7	Channel access	164
7.8	DCS Channel assignment.....	165
7.9	Printing of DTM-specific documents.....	169
7.10	Printing of Frame Application-specific documents	170
7.10.1	General	170
7.10.2	Processing a document.....	171
7.10.3	Rules for use of DTM-specific style sheets	173
7.11	Propagation of changes.....	174
7.12	Locking.....	175
7.12.1	Locking for non-synchronized DTMs	175
7.12.2	Locking for synchronized DTMs.....	176
7.13	Instantiation and release	178
7.13.1	Instantiation of a new DTM.....	178
7.13.2	Instantiation of an existing DTM	178
7.13.3	Instantiation of a DTM ActiveX® user interface	179
7.13.4	Release of a DTM user interface	179
7.14	Persistent storage of a DTM	180
7.14.1	State machine of instance data.....	180
7.14.2	Saving instance data of a DTM.....	182
7.14.3	Reload of a DTM object for another instance	183

7.14.4	Copy and versioning of a DTM instance	183
7.15	Audit trail	184
7.16	Comparison of two instance data sets	185
7.16.1	Comparison without user interface	185
7.16.2	Comparison with user interface	186
7.17	Failsafe data access	187
7.18	Set or modify device address with user interface	188
7.19	Set or modify known device addresses without user interface	189
7.20	Display or modify all child device addresses with user interface	190
7.21	Device initiated data transfer	191
7.22	Starting and releasing DTM user interface in modal dialog	192
7.23	Parent component handling redundant slave	193
7.24	Initialization of a Channel ActiveX control	195
7.24.1	General	195
7.24.2	Supports IFdtChannelActiveXcontrol2	195
7.24.3	Does not support IFdtChannelActiveXControl2	195
7.25	DTM upgrade	196
7.25.1	General	196
7.25.2	Saving data from a DTM to be upgraded	196
7.25.3	Loading data in the replacement DTM	197
7.26	Usage of IDtmSingleDeviceDataAccess::ReadRequest / Write Request	198
7.27	Instantiation of DTM and BTM	199
8	Installation issues	201
8.1	Registry and device information	201
8.1.1	Visibility of business objects of a DTM	201
8.1.2	Component categories	201
8.1.3	Registry entries	202
8.1.4	Installation issues	202
8.1.5	Microsoft's standard component categories manager	203
8.1.6	Building a Frame Application-database of supported devices	203
8.1.7	DTM registration	203
8.2	Paths and file information	204
8.2.1	Path information provided by a DTM	204
8.2.2	Paths and persistency	204
8.2.3	Multi-user systems	204
9	Description of data types, parameters and structures	205
9.1	Ids	205
9.2	Data type definitions	205
Annex A (normative)	FDT IDL	207
Annex B (normative)	Mapping of services to interface methods	223
B.1	General	223
B.2	DTM services	223
B.3	Presentation object services	227
B.4	General channel services	227
B.5	Process channel services	228
B.6	Communication Channel Services	228
B.7	Frame Application Services	229
Annex C (normative)	FDT XML schemas	232

C.1	General.....	232
C.2	FDTDataTypesSchema.....	232
C.3	FDTApplicationIdSchema	248
C.4	FDTUserInformationSchema.....	248
C.5	DTMInformationSchema	250
C.6	DTMFunctionCallSchema	253
C.7	DTMParameterSchema	254
C.8	DTMDocumentationSchema	262
C.9	DTMProtocolsSchema	264
C.10	DTMSystemTagListSchema.....	265
C.11	DTMAuditTrailSchema.....	266
C.12	DTMDeviceStatusSchema	268
C.13	DTMFunctionsSchema	269
C.14	DTMChannelFunctionsSchema.....	273
C.15	DTMOnlineCompareSchema	276
C.16	FDTFailSafeDataSchema	277
C.17	DTMTopologyScanSchema.....	277
C.18	FDTOperationPhaseSchema	278
C.19	DTMInitSchema	279
C.20	FDTUserMessageSchema	279
C.21	DTMInfoListSchema	281
C.22	FDTTopologyImportExportSchema	282
C.23	DTMDeviceListSchema	286
C.24	DTMSystemGuiLabelSchema	288
C.25	DTMStateSchema	288
C.26	DTMEnvironmentSchema	289
C.27	FDTConnectResponseSchema	290
C.28	TypeRequestSchema	290
C.29	FDTScanRequestSchema.....	291
C.30	FDTxxxIdentSchema	292
C.31	FDTxxxDeviceTypeIdentSchema	292
C.32	FDTxxxScanIdentSchema	293
C.33	DTMIdentSchema.....	293
C.34	DTMScanIdentSchema.....	294
C.35	DTMDeviceTypeIdentSchema.....	296
C.36	DTMItemListSchema	298
C.37	BtmDataTypesSchema	303
C.38	BtmInformationSchema	305
C.39	BtmParameterSchema.....	306
C.40	BtmInitSchema.....	308
C.41	BtmInfoListSchema	309
Annex D (informative)	FDT XML styles – Documentation	310
Annex E (informative)	FDT XSL Transformation.....	314
E.1	Identification transformation	314
E.2	Hint.....	314
Annex F (normative)	Channel schema	316
F.1	FDTBasicChannelParameterSchema	316
F.2	Template for Channel Schema.....	317
Annex G (normative)	FDT version interoperability guide.....	318

G.1	Overview.....	318
G.2	General.....	318
G.3	Component interoperability.....	318
G.4	FDT type library	320
G.5	DTM and device versions	320
G.6	Persistence.....	320
G.7	Nested communication	321
G.7.1	General	321
G.7.2	Data exchange.....	321
G.7.3	Communication channel upgrade.....	321
G.7.4	Scenarios	321
G.7.5	OnAddChild	322
G.8	Implementation hints	322
G.8.1	Interfaces	322
G.8.2	Persistence.....	322
Annex H (informative)	Implementation with Net technology	323
H.1	How FDT supports .NET based development	323
H.2	Microsoft .NET Framework 1.1 and 2.0 compatibility	323
H.3	Side-by-side installation and related problems	323
H.4	How to avoid compatibility issues	324
Annex I (informative)	Trade names.....	325
Bibliography		326
Figure 1	– Part 41 of the IEC 62453 series	14
Figure 2	– Frame Application interfaces	18
Figure 3	– DTM interfaces.....	18
Figure 4	– FDT Client/server relationship via XML	19
Figure 5	– Data access and storage	21
Figure 6	– Communication	22
Figure 7	– Documentation	22
Figure 8	– Parameter verification in case of failsafe devices	23
Figure 9	– State machine of a DTM	24
Figure 10	– Device identification	30
Figure 11	– Structural overview.....	32
Figure 12	– Interfaces of FDT objects – DTM and DtmActiveXControl	36
Figure 13	– Interfaces of FDT object – Frame Application	37
Figure 14	– FDT objects – FDT-Channel	37
Figure 15	– FDT objects – BTM and BtmActiveXControl	38
Figure 16	– Peer to peer connection between DTM and device	147
Figure 17	– Asynchronous connect (peer to peer).....	148
Figure 18	– Asynchronous disconnect (peer to peer)	148
Figure 19	– Asynchronous transaction (peer to peer).....	149
Figure 20	– System-topology.....	150
Figure 21	– Generation of system topology by Frame Application.....	151
Figure 22	– Generation of system topology – Participation of DTM.....	152

Figure 23 – System connection (across communication hierarchy)	153
Figure 24 – Asynchronous transactions (system connection)	154
Figure 25 – Scan network topology	155
Figure 26 – Cancel topology scan	156
Figure 27 – Provisional topology scan	157
Figure 28 – Scan for communication hardware	158
Figure 29 – Manufacturer-specific device identification	160
Figure 30 – Add protocol-specific schemas to Frame Applications schema sub path.....	161
Figure 31 – Frame Application reads protocol-specific device identification information of DTMDDeviceTypes.....	162
Figure 32 – Bus master configuration	163
Figure 33 – Starting and releasing applications	164
Figure 34 – Channel access.....	165
Figure 35 – DCS channel assignment single DTM	166
Figure 36 – Sequence of channel assignment for a single DTM.....	167
Figure 37 – Modular DTM structure	168
Figure 38 – Channel assignment for modular DTMs.....	169
Figure 39 – Printing of DTM-specific documents	170
Figure 40 – Printing of Frame Application-specific documents.....	171
Figure 41 – Report generation (Frame Application style).....	172
Figure 42 – Report generation (device vendor-specific style)	173
Figure 43 – Propagation of changes	174
Figure 44 – Locking for non-synchronized DTMs	176
Figure 45 – Locking for synchronized DTMs	177
Figure 46 – Instantiation of a new DTM	178
Figure 47 – Instantiation of an existing DTM.....	179
Figure 48 – Instantiation of a DTM user interface.....	179
Figure 49 – Release of a DTM user interface.....	180
Figure 50 – State machine of instance data set	181
Figure 51 – Persistence states of a data set	182
Figure 52 – Saving instance data of a DTM	183
Figure 53 – Copy and versioning of a DTM instance	184
Figure 54 – Audit trail	185
Figure 55 – Comparison without user interface	186
Figure 56 – Comparison with user interface	187
Figure 57 – Failsafe data access.....	188
Figure 58 – Set or modify device address with user interface	189
Figure 59 – Set or modify known device addresses without user interface	190
Figure 60 – Display or modify all child device addresses with user interface.....	191
Figure 61 – Device initiated data transfer	192
Figure 62 – Modal DTM user interface.....	193
Figure 63 – Handling of a redundant slave	194
Figure 64 – Init of Channel ActiveX with IFdtChannelActiveXControl2	195

Figure 65 – Init of Channel ActiveX® without IFdtChannelActiveXControl2	196
Figure 66 – Saving data from a DTM to be upgraded	197
Figure 67 – Loading data in the replacement DTM	198
Figure 68 – Usage of IDtmSingleDeviceDataAccess	199
Figure 69 – General sequence of creation and instantiation of blocks	200
Figure E.1 – XSLT role	315
Table 1 – Definition of DTM state machine	25
Table 2 – Task related DTM interfaces	26
Table 3 – Task related DTM ActiveX® interfaces	27
Table 4 – Task related FDT-Channel interfaces	27
Table 5 – Task related Channel ActiveX® interfaces	27
Table 6 – Task related BTM interfaces	28
Table 7 – Task related BTM ActiveX® interfaces	28
Table 8 – Task related Frame Application interfaces	28
Table 9 – Semantic identification information	33
Table 10 – Regular expressions	34
Table 11 – Availability of DTM methods in different states	39
Table 12 – Availability of Frame Application interfaces	41
Table 13 – Description of instance data set states	181
Table 14 – Description of persistent states	182
Table 15 – Component categories	201
Table 16 – Combinations of categories	202
Table 17 – Example for DTM registration	202
Table 18 – FDT-specific Ids	205
Table 19 – Basic data types	205
Table 20 – Helper objects for documentation	206
Table B.1 – General services	223
Table B.2 – DTM services related to installation	223
Table B.3 – DTM services related to DTM information	224
Table B.4 – DTM services related to DTM state machine	224
Table B.5 – DTM services related to function	225
Table B.6 – DTM services related to documentation	225
Table B.7 – DTM services to access the instance data	225
Table B.8 – DTM services to access diagnosis	226
Table B.9 – DTM services to access the device data	226
Table B.10 – DTM services related to network management information	226
Table B.11 – DTM services related to online operation	226
Table B.12 – DTM services related to FDT-Channel objects	227
Table B.13 – DTM services related to import and export	227
Table B.14 – DTM services related to data synchronization	227
Table B.15 – General channel services	228
Table B.16 – Channel services for IO related information	228

Table B.17 – Channel services related to communication.....	228
Table B.18 – Channel services related sub-topology management.....	229
Table B.19 – Channel services related to functions.....	229
Table B.20 – Channel services related to scan.....	229
Table B.21 – FA services related to general events.....	229
Table B.22 – FA services related to topology management.....	230
Table B.23 – FA services related to redundancy.....	230
Table B.24 – FA services related to storage of DTM data.....	230
Table B.25 – FA services related to DTM data synchronization.....	231
Table B.26 – FA services related to presentation.....	231
Table B.27 – FA services related to audit trail.....	231
Table C.1 – Description of general XML attributes.....	232
Table C.2 – Description of general XML elements.....	237
Table C.3 – Device classification ID.....	239
Table C.4 – Device classification according to IEC TR 62390:2005, Annex G.....	240
Table C.5 – Description of applicationId attribute.....	248
Table C.6 – Description of applicationId elements.....	248
Table C.7 – Description of user information attributes.....	249
Table C.8 – Description of user information elements.....	249
Table C.9 – Description of DTM information attributes.....	250
Table C.10 – Description of DTM information elements.....	250
Table C.11 – Description of function call attributes.....	254
Table C.12 – Description of parameter document attributes.....	254
Table C.13 – Description of parameter document elements.....	255
Table C.14 – Description of documentation attributes.....	263
Table C.15 – Description of documentation elements.....	263
Table C.16 – Description of protocols element.....	265
Table C.17 – Description of system tag attributes.....	265
Table C.18 – Description of system tag elements.....	265
Table C.19 – Description of audit trail attributes.....	267
Table C.20 – Description of audit trail elements.....	267
Table C.21 – Description of device status attribute.....	268
Table C.22 – Description of device status elements.....	268
Table C.23 – Description of function attributes.....	269
Table C.24 – Description of function elements.....	270
Table C.25 – Description of channel functions attributes.....	273
Table C.26 – Description of channel function elements.....	274
Table C.27 – Description of comparison attribute.....	276
Table C.28 – Description of comparison elements.....	276
Table C.29 – Description of fail safe attributes.....	277
Table C.30 – Description of fail safe elements.....	277
Table C.31 – Description of topology scan elements.....	278
Table C.32 – Description of operation phase attribute.....	278

Table C.33 – Description of operation phase element	278
Table C.34 – Description of DTM init element	279
Table C.35 – Description of user message attributes	279
Table C.36 – Description of user message elements	280
Table C.37 – Description of DTM info list elements	281
Table C.38 – Description of topology attributes	282
Table C.39 – Description of topology elements	282
Table C.40 – Description of device list attributes	286
Table C.41 – Description of device list elements	287
Table C.42 – Description of gui label element	288
Table C.43 – Description of DTM state element	289
Table C.44 – Description of frame version element	289
Table C.45 – Description of connect response element	290
Table C.46 – Description of type request element	290
Table C.47 – Description of scan request attributes	291
Table C.48 – Description of scan request elements	291
Table C.49 – Description of common identification attributes	293
Table C.50 – Description of common identification element	294
Table C.51 – Description of scan identification attributes	294
Table C.52 – Description of scan identification elements	294
Table C.53 – Description of device type identification element	296
Table C.54 – Description of item list attributes	298
Table C.55 – Description of item list elements	299
Table C.56 – Description of BTM data type attributes	303
Table C.57 – Description of BTM data type elements	304
Table C.58 – Description of BTM information elements	305
Table C.59 – Description of BTM parameter elements	306
Table C.60 – Description of BTM init element	308
Table C.61 – Description of BTM info list element	309
Table F.1 – Description of basic channel attribute	316
Table F.2 – Description of basic channel elements	316
Table F.3 – Description of xxx channel parameter attribute	317
Table F.4 – Description of xxx channel parameter elements	317
Table G.1 – Interoperability between components of different versions	319

INTERNATIONAL ELECTROTECHNICAL COMMISSION

FIELD DEVICE TOOL (FDT) INTERFACE SPECIFICATION –**Part 41: Object model integration profile –
Common object model****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 62453-41, 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:

This second edition cancels and replaces the first edition published in 2009, and constitutes a technical revision.

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

- a) correction of specification for bus master configuration;
- b) correction of specification for propagation of changes;
- c) correction of description of DTM services for online operation.

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

Enquiry draft	Report on voting
65E/437/DTR	65E/485/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 62453, 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, typically in the area of the process industry, the unambiguous definition of engineering interfaces that are easy to use for all those involved is of great importance.

A device-specific software component, 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 is, in general, open for all kind of fieldbuses and thus meets the requirements for integrating different kinds of devices into heterogeneous control systems.

Figure 1 shows how IEC TR 62453-41 is incorporated in the structure of the IEC 62453 series.

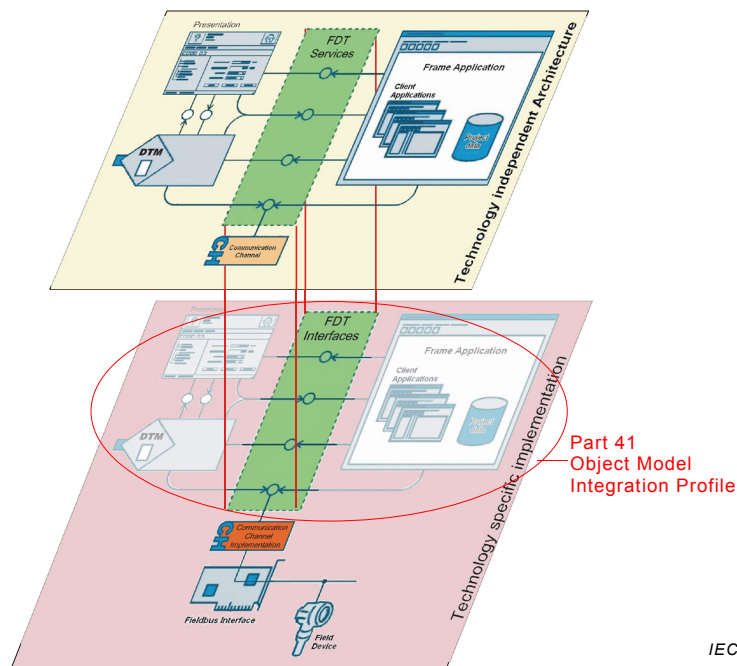


Figure 1 – Part 41 of the IEC 62453 series

FIELD DEVICE TOOL (FDT) INTERFACE SPECIFICATION –

Part 41: Object model integration profile – Common object model

1 Scope

This part of IEC 62453, which is a technical report, defines how the common FDT principles are implemented based on the Microsoft¹ COM technology, including the object behavior and object interaction via COM interfaces.

This part specifies the technology-specific implementation of the protocol-specific functionality and communication services.

This part of IEC 62453 is informative, however when this part is applied its requirements need to be implemented as specified.

This part specifies FDT version 1.2.1.

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 61784 (all parts), *Industrial communication networks – Profiles*

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

IEC 62453-2:—2, *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

ActiveX^{®3}

GUI component technology based on the Microsoft Component Object Model (COM/DCOM)

Note 1 to entry: Former standard was OLE controls (OCX).

¹ See Annex I.

² To be published concurrently with this technical report.

³ See Annex I.

3.1.2**asynchronous function**

non-blocking function in which the calling process continues execution while the function is executed in the background

3.1.3**CLSID**

UUID for a COM class

3.1.4**ProgID**

human readable ID for a COM class

3.1.5**synchronous function**

blocking function in which the calling process stops execution while the function is executed

3.2 Abbreviations

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

API	Application Programming Interface
BTM	Block Type Manager
CLSID	Class ID
COM	Common Object Model
DCOM	Distributed COM
DLL	Dynamic Linked Library
DOM	Document Object Model
DTM	Device Type Manager
FA	Frame Application
FB	Function Block
FDT	Field Device Tool
GUI	Graphical User Interface
GUID	Globally Unique Identifier (a UUID)
HART® ⁴	Highway Addressable Remote Transducer
IID	Interface ID
LCID	Locale ID
MIDL	Microsoft Interface Definition Language
MSDN®	Microsoft Developer Network
OCX	OLE Controls
OLE	Object Linking and Embedding
ProgID	Programmatic ID
XDR	XML Data Reduced
XSL	Extensible Stylesheet Language
XSLT	XSL Transformations

3.3 Conventions

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

⁴ See Annex I.

4 Implementation concept

4.1 Technological orientation

The ActiveX® technology introduced by Microsoft makes it possible to define interfaces which contain not only data but also functions. These possibilities have already been successfully used in conjunction with OPC (originally: OLE for Process Control) definition.

Within this technical report, implementation of FDT-only user interfaces in ActiveX® technology are specified for the engineering components of field devices. If the engineering system implements the corresponding interfaces, the ActiveX® technology provides the automatic integration of the components and takes care of the interaction between the engineering system and the software components of the devices. Furthermore the FDT interface specification allows the integration of device components with integrated user interfaces as well as the embedding of ActiveX® controls provided by the device component for special engineering tasks.

The implementation of FDT's client/server architecture defined in this technical report is based on Microsoft COM.

This part of IEC 62453 specifies COM interfaces (what the interfaces are), not the implementation (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 seem 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.

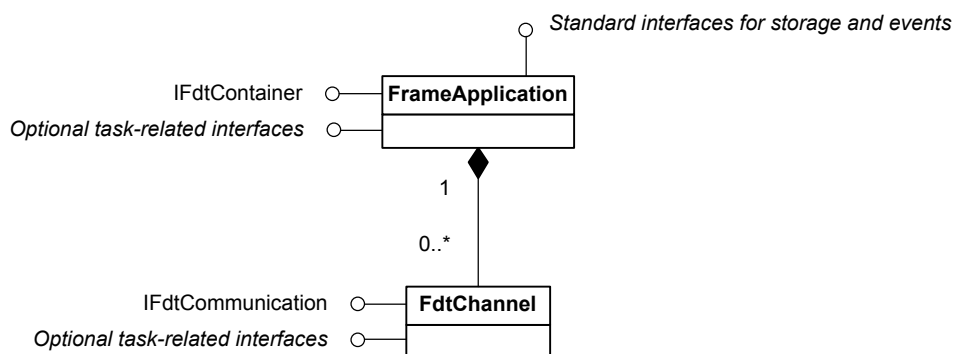
4.2 Implementation of abstract FDT object model

4.2.1 General

The FDT objects are implemented as COM objects. The general expectation is that these objects may be implemented as inproc as well as outproc servers. The Frame Application is responsible for organizing the execution of the objects in a distributed system (based on a vendor-specific implementation).

4.2.2 FDT Frame Application (FA)

From a DTM point of view, all task-related interfaces for the interaction with Frame Application are available via the main interface IFdtContainer of the Frame Application (see Figure 2).



IEC

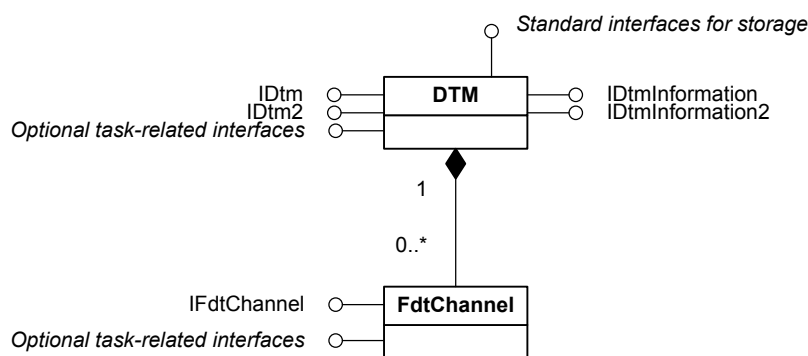
Figure 2 – Frame Application interfaces

In a complex plant environment, there are also complex communication networks for linking the process devices. No DTM should need any information about the topology of a system network. So it is up to the Frame Application to organize the routing for accessing a device. The Frame Application has to provide in each case a peer-to-peer connection (physical or logical). So its up to the Frame Application to manage the multi user access to a device.

4.2.3 Device Type Manager (DTM)

4.2.3.1 DTM interfaces

The services of a DTM are provided by means of COM interfaces (see Figure 3).



IEC

Figure 3 – DTM interfaces

The interfaces IDtm and IDtmInformation have to be implemented by each DTM (see 5.2). These interfaces provide the services and information for controlling a DTM. From a Frame Application's point of view, all task-related interfaces for the interaction with the device functionality are available via these interfaces. Which task related interfaces are provided depends on the capability of the DTM and the corresponding device. Each interface is described in detail in the appropriate subclause.

4.2.3.2 Block Type Manager (BTM)

The BTM implements similar interfaces to those specified for a DTM. Block-specific schemas replace device related XML schemas to provide block information. For example, BtmInformationSchema replaces DtmInformationSchema.

4.2.4 Presentation object

A presentation object as defined by this specification may be an ActiveX® object (allows integration into GUI of FA) or a standalone program, that may be integrated by a DTM.

IEC TR 62453-61 provides a style guide for a common look and feel of ActiveX® user interfaces.

4.2.5 FDT-Channel object

The FDT-Channel object implements at least the IFdtChannel interface that gives access to all parameters of the channel which describes the channel itself.

If the device provides communication functionality, like a fieldbus adapter or a gateway, the FDT-Channel object shall implement further interfaces for the communication via this channel (Communication Channel). Each interface of the FDT-Channel object is described in detail, see the appropriate subclause.

As defined in IEC 62453-2, Communication Channels represent the gateway from the FDT-specific to the Frame Application-specific communication. At least the interface IFdtCommunication shall be implemented for Communication Channels.

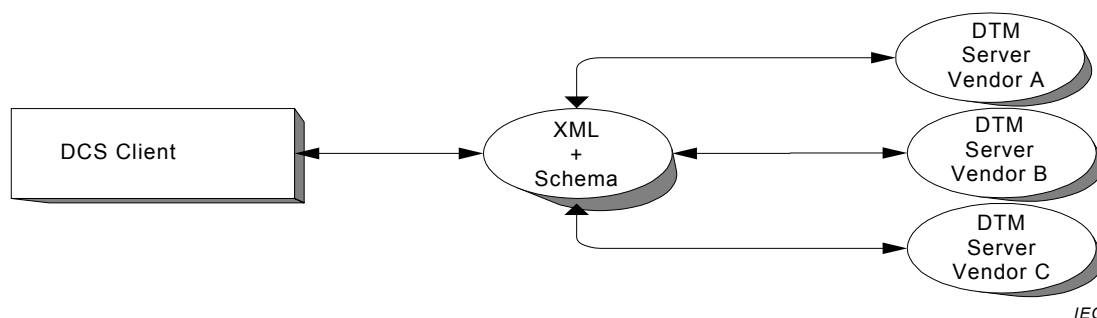
IFdtCommunication always provides the communication functionality for DTMs to access their fieldbus devices. All actions that belong to the physical fieldbus shall be done by using this interface.

4.3 Object interaction

4.3.1 Parameter interchange via XML

Data exchange between objects is implemented based on transport of XML documents (XDR format) via COM interfaces. One example for this data exchange is the parameter interchange via XML is to provide a way to exchange information between Frame Application and DTMs (see Figure 4). Typically, in process control systems, multiple use cases like observing, channel assignment, or master configuration, need information about the configuration of a device.

XML is not meant to replace proprietary formats; it is meant to provide access to data that is stored in a proprietary format. It is recommended that data is stored locally in the fashion that makes the most sense. XML provides an extendable standard to connect FDT components. The data exchange is done via XML documents. Within these documents XML tags are used to delimit pieces of data. XML leaves the interpretation of the data to the application that reads it. To get a common understanding of the exchanged data FDT uses XML schemas for validation. For the data access are standardized tools like the DOM (W3C's document object model) available.



IEC

Figure 4 – FDT Client/server relationship via XML

XML schemas are valid XML syntax themselves and are used to validate XML data. They allow the validation of the document structure and the data types of the elements.

W3C's document object model (DOM) is a standard internal representation of the document structure. It aims to make it easy for programmers to access components and delete, add, or edit their content, attributes and style. In essence, the DOM makes it possible for programmers to write applications that work properly on all browsers and servers, and on all platforms. While programmers may need to use different programming languages, they do not need to change their programming model.

An XML parser usually generates the DOM. Microsoft® provides such an XML parser. So the DOM API is accessible in VC++®⁵, Visual Basic®⁶ and VBScript.

The parsing of XML documents with XML schemas ensures a valid DOM with well-defined elements.

It is recommended that the FDT developer always work with the DOM because

- the XML parser generates the DOM from the transferred XML data,
- the schemas ensure a valid DOM with well defined elements,
- the DOM supplies standard tree- and collection-methods for data access,
- the DOM can generate the XML data for the data transfer.

The validation of a XML document ensures that the content of an attribute is valid according to the XML schemas.

- Non string data types (Ui4,enumeration,etc.): Empty or invalid attribute values will be detected during the validation of the document.
- Data types 'string' and 'bin.hex'(hexadecimal digits representing octets): Empty values are possible. No parsing error is generated.

To avoid interoperability problems developers should consider following basic rules.

- Receiving XML documents:
To ensure a robust implementation, developers should be aware that XML documents with empty attributes of type 'string' or 'bin.hex' could be received. FDT components should be capable to handle this in a proper way.
- Providing XML documents:
Empty optional attributes should be avoided generally, because they are a potential cause of interoperability problems and a waste of resources. Also optional XML elements should be removed if they contain no information. Exception: Empty 'string' attributes can make sense in some cases (as shown in the example below).

If a Frame Application can not provide information about login location and session description, the corresponding attributes should be removed.

If a Frame Application can not provide attribute values, the attributes should not be provided.

Negative Example:

```
<FDT xmlns="x-schema:FDTUserInformationSchema.xml" xmlns:fdt="x-schema:FDTDataTypesSchema.xml">
  <FDTUserInformation projectName="Project1" userName="ThisUser" userLevel="maintenance"
    loginLocation="" sessionDescription=""/>
</FDT>
```

⁵ See Annex I.

⁶ See Annex I.

Positive Example:

```
<FDT xmlns="x-schema:FDTUserInformationSchema.xml" xmlns:fdt="x-schema:FDTDataTypesSchema.xml">
  <FDTUserInformation projectName="Project1" userName="ThisUser" userLevel="maintenance"/>
</FDT>
```

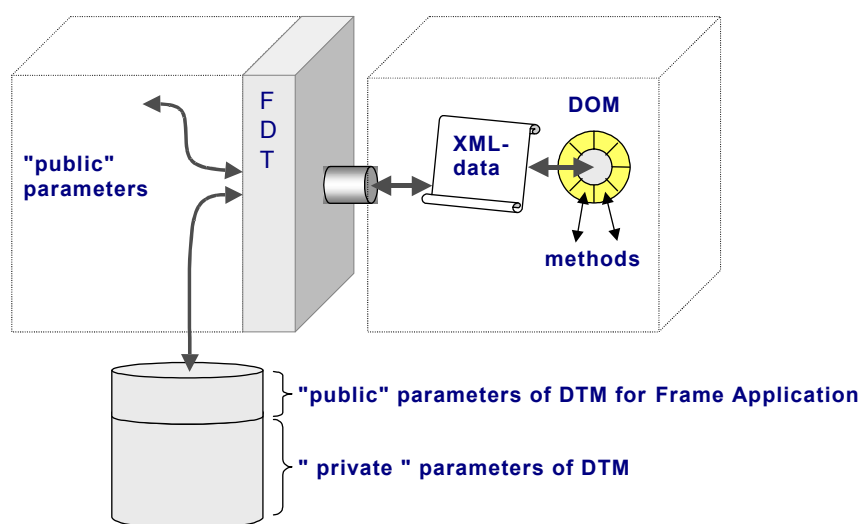
If the session description is generally available, but the user has not typed in something, then it makes sense to use an empty attribute.

Acceptable Example:

```
<FDT xmlns="x-schema:FDTUserInformationSchema.xml" xmlns:fdt="x-schema:FDTDataTypesSchema.xml">
  <FDTUserInformation projectName="Project1" userName="ThisUser" userLevel="maintenance"
    sessionDescription=""/>
</FDT>
```

4.3.2 Examples of usage

Parameter interchange between DTM and Frame Application is done via XML document. Object oriented access to data is provided when using an XML parser that generates an in-memory representation of the XML data (e.g. a DOM). Instance data to be stored (persistence) can also be handled as an XML document to simplify the DTM development and to have a homogeneous data handling within a DTM. But also if the data are stored as XML the content of these data is only known by DTM (see Figure 5).



IEC

Figure 5 – Data access and storage

The XML document for communication includes device data and the necessary information for routing to establish peer-to-peer connection between DTM and field device (see Figure 6).

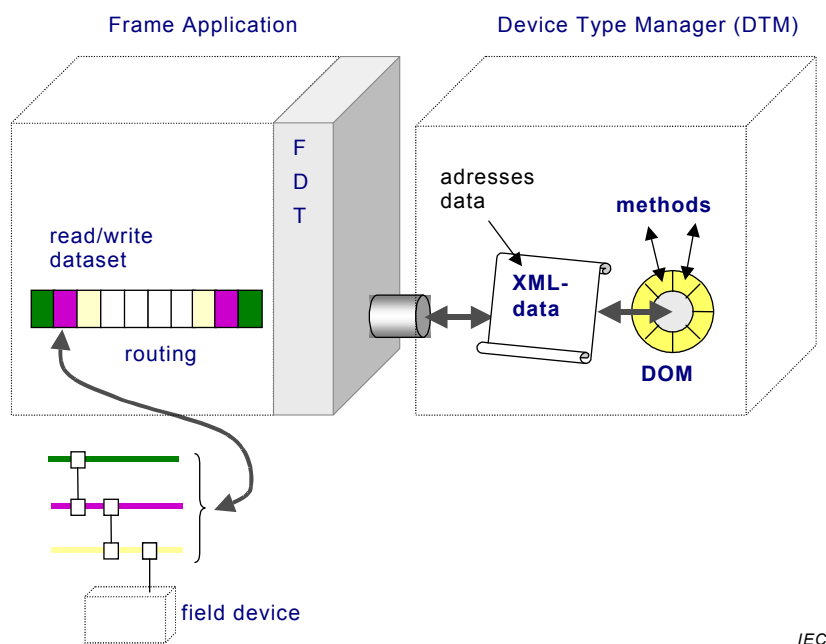


Figure 6 – Communication

Here the DTM only knows the address information for a peer-to-peer connection. The Frame Application adds during runtime all necessary routing information.

For documentation of field devices within the project documentation and field device-specific documentation XML is used in conjunction with XML style sheets (XSL) for layout (see Figure 7). A default style sheet is supported by the Frame Application.

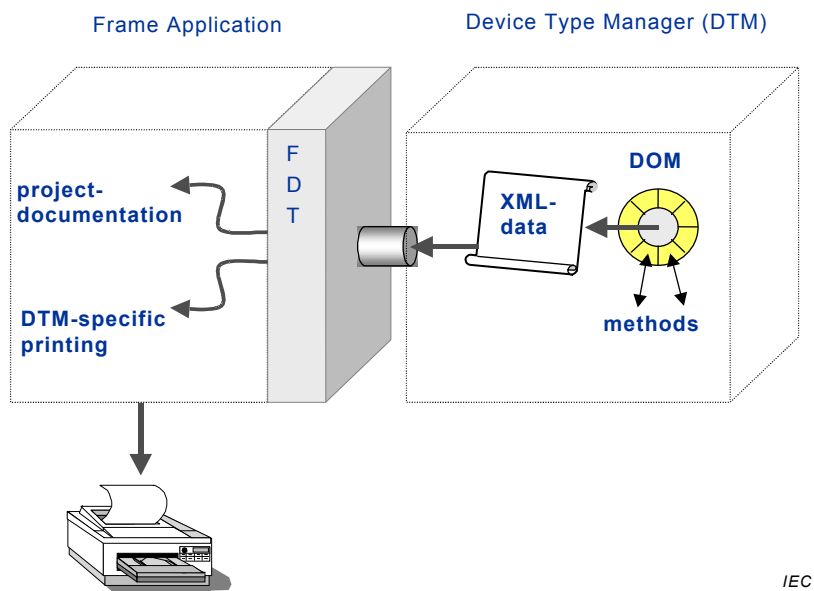


Figure 7 – Documentation

In case of failsafe field devices the instance data that are stored in the controller (e.g. PLC) after upload from the field device shall be verified by the DTM (see Figure 8). The interchange format of this data is also an XML document.

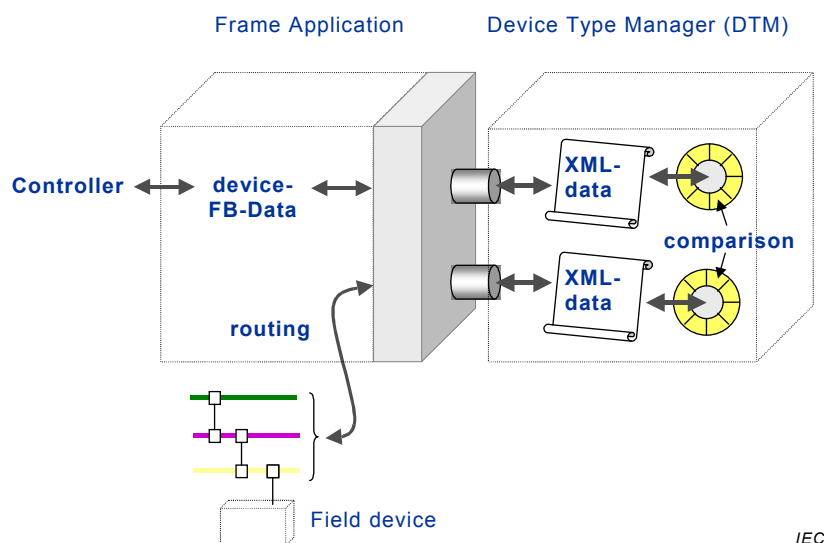


Figure 8 – Parameter verification in case of failsafe devices

4.4 Implementation of DTM data persistence and synchronization

4.4.1 Persistence overview

The synchronization of the DTM data set is done via the interface IFdtContainer.

To use the storage component of the Frame Application, a DTM has to implement the standard COM-interfaces IPersistPropertyBag and IPersistStreamInit. It is not determined how the DTM performs the storage or which kind of private data of a DTM is stored.

The Frame Application requests the storage of private data of a DTM. A DTM shall be able to re-establish its complete state when this is requested by the Frame Application. This is done by calling the function IPersistXXX::Load of the DTM. Some DTMs do not support reload of a DTM object by calling IPersistXXX::Load several times. With an IPersistXXX::Save request a DTM shall store its private data within the storage provided by the Frame Application. A DTM object for a new instance shall be initialized if the IPersistXXX::InitNew method is called by the Frame Application.

DTMs using additional own data storage shall provide all data which are necessary for commissioning via the IPersistXXX interface. Private data not provided via the IPersistXXX interface shall be offered to the Frame Application for import and export via the IDtmImportExport interface. Also an IStream object is used to store and retrieve such import and export data.

In order to simplify the DTM development, it is up to a DTM to implement one of the defined persistent interfaces (IPersistStreamInit or IPersistPropertyBag) according to the Microsoft® standard. The Frame Application shall be able to handle both.

References to DTMs do not belong to the instance data of a DTM. A DTM shall not store any references to other DTMs. A DTM can get information concerning its parents or childs via IFdtTopology::GetParentNodes() and IFdtTopology::GetChildNodes().

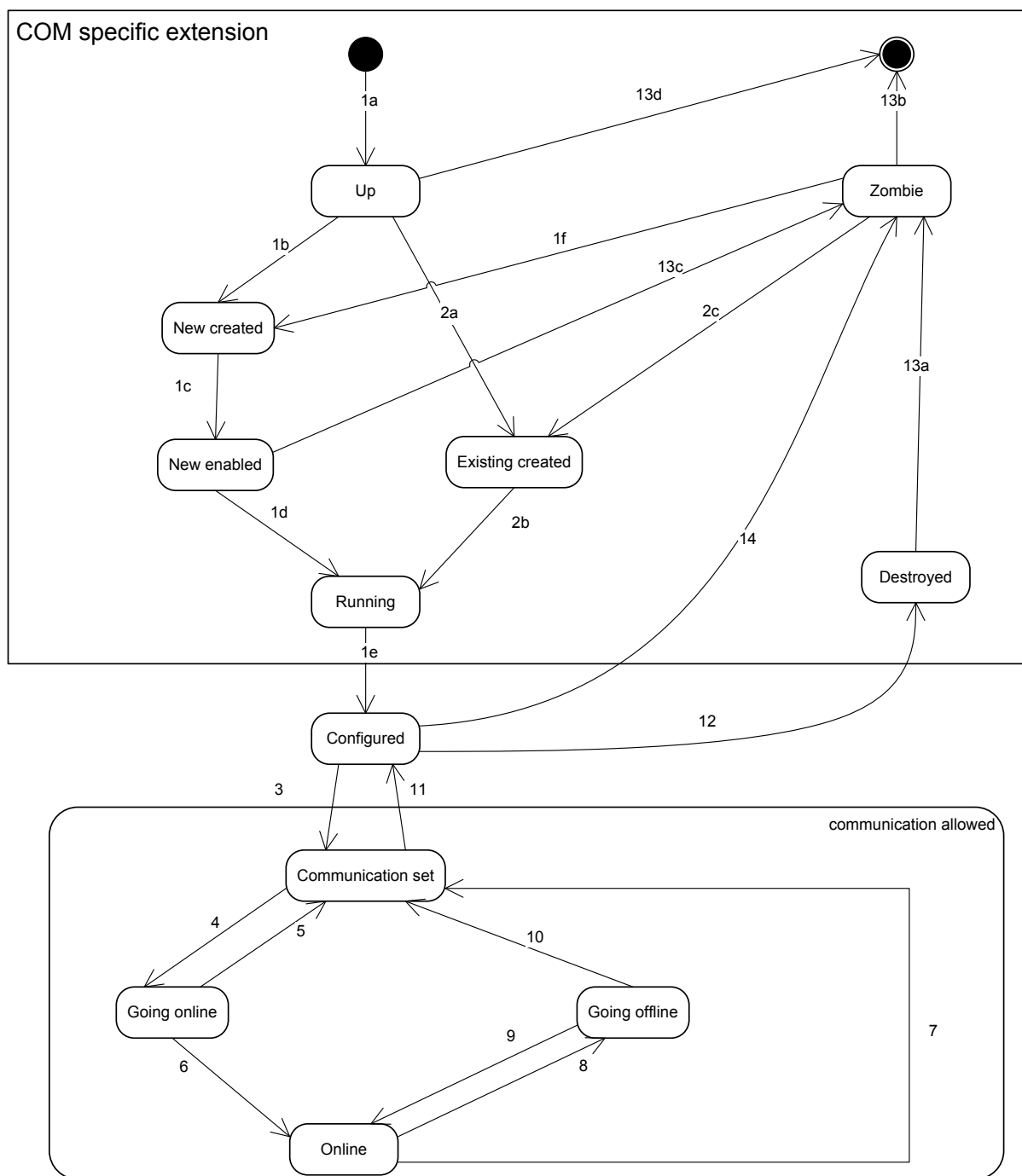
DTM should report data load errors via standard COM error mechanism (HRESULT not equal to S_OK). Optionally, DTM can write further human readable error information to standard COM global error info (Win32 SetLastError method).

4.4.2 Persistence interfaces

For detailed information about IPersistStreamInit and IPersistPropertyBag please refer to the standard Microsoft® documentation like MSDN®.

4.5 DTM state machine

The following state machine shows the different states of a DTM (see Figure 9). The state machine is based on the general state machine as defined in IEC 62453-2. It is extended to accommodate the specific needs of COM based implementation.



IEC

Figure 9 – State machine of a DTM

Table 1 provides a description of the transitions.

Table 1 – Definition of DTM state machine

ID	Start state	End state	Trigger	Condition	Action
1a	<non existent>	Up	CoCreateInstance()		
1b	Up	New created	IPersistXXX:InitNew()		
1c	New created	New enabled	IDtm:Environment() or IDtm2:Environment2()		
1d	New enabled	Running	IDtm:InitNew()		
1e	Running	Configured	IDtm:Config()		
1f	zombie	New created	IPersistXXX:InitNew()		
2a	Up	Existing created	IPersistXXX:Load()		
2b	Existing created	Running	IDtm:Environment() or IDtm2:Environment2()		
2c	Zombie	Existing created	IPersistXXX:Load()		
3	Configured	Communication set (not connected)	IDtm: SetCommunication()		
4	Communication set (not connected)	Going online (connecting)	Trigger of Online service of the DTM		IDtmCommunication: ConnectRequest()
5	Going online (connecting)	Communication set (not connected)	IDtmCommunicationEvent: OnConnectResponse() or IDtmCommunicationEvent2: OnConnectResponse2()	Connection could not be established, negative response	
6	Going online (connecting)	Online	IDtmCommunicationEvent: OnConnectResponse() or IDtmCommunicationEvent2: OnConnectResponse2()	Connection was established, positive response	
7	Online (connected)	Communication set (not connected)	IDtmCommunicationEvent: OnAbort()		
8	Online (connected)	Going offline (disconnecting)	Online service of DTM is finished or IDtm: PrepareToReleaseCommuni- cation()		IDtmCommunication: DisconnectRequest()
9	Going offline (disconnecting)	Online (connected)	IDtmCommunicationEvent: OnDisconnectResponse()	Connection could not be terminated	
10	Going offline (disconnecting)	Communication set (not connected)	IDtmCommunicationEvent: OnDisconnectResponse()	Connection was terminated	If PreparedTo- Release- Communication was called (see transition 8): IDtmEvents:On- PreparedToRelease- Communication()
11	Communication set (not connected)	Configured	IDtm: ReleaseCommunication()		
12	Configured	Destroyed	IDtm:PrepareToDelete()		
13a	Destroyed	Zombie	IDtm:PrepareToRelease()		IDtmEvents: OnPreparedToRelea- se()

ID	Start state	End state	Trigger	Condition	Action
13b	Zombie	<NonExisting>	Release()		
13c	new enabled	Zombie	IDtm:PrepareToRelease()		IDtmEvents: OnPreparedToRelease()
13d	Up	<NonExisting>	Release()		
14	Configured	Zombie	IDtm:PrepareToRelease()		IDtmEvents: OnPreparedToRelease()

5 General concepts

5.1 General

This clause provides general information about the FDT interfaces, and some background information about how the designers of FDT expect these interfaces to be implemented and used.

5.2 Overview of task related FDT interfaces

All FDT interfaces are task related. Each object shall implement a mandatory set of interfaces expected by all other objects. By implementing optional FDT interfaces an object is able to support additional functionality, for example a DTM may provide documentation in XML format or special diagnostics, a frame-application may provide audit trail functionality.

Each object is able to determine the availability of such optional interfaces of other objects during runtime.

All defined FDT interfaces are fixed and will never be changed. Additional future extensions will be based on additional optional interfaces. A DTM or frame-application is then able to add a higher FDT version support by implementing or using such additional FDT interfaces.

Depending on the functionality of a DTM, additionally to the default set of mandatory interfaces an extra set of interfaces may be mandatory to support (see Table 2).

Table 2 – Task related DTM interfaces

Device Type Manager	Availability	User interface	ActiveX® control user interface	Device with online data	Gateway DTM	Communication DTM for PC/Fieldbus adapter
IPersistXXX	Mandatory					
IDtm	Mandatory					
IDtm2	Mandatory					
IDtmActiveXInformation	Optional		Mandatory			
IDtmApplication	Optional	Mandatory				
IDtmChannel	Optional				Mandatory	Mandatory
IDtmDocumentation	Mandatory					
IDtmDiagnosis	Mandatory					
IDtmImportExport	Optional					
IDtmInformation	Mandatory					
IDtmInformation2	Mandatory					
IDtmOnlineDiagnosis	Mandatory					
IDtmOnlineParameter	Optional			Mandatory		
IDtmParameter	Mandatory					

Device Type Manager	Availability	User interface	ActiveX® control user interface	Device with online data	Gateway DTM	Communication DTM for PC/Fieldus adapter
IFdtCommunicationEvents	Optional			Mandatory	Mandatory	
IFdtCommunicationEvents2	Optional			Mandatory	Mandatory	
IDtmHardwareIdentification	Optional					
IDtmSingleDeviceDataAccess	Optional			Mandatory		
IDtmSingleInstanceDataAccess	Mandatory					
IFdtEvents	Mandatory					

The mandatory interfaces of a DTM ActiveX® are shown in Table 3.

Table 3 – Task related DTM ActiveX® interfaces

DTM ActiveX® control	Availability
IDtmActiveXControl	Mandatory

Depending on the functionality of a channel, additionally to the mandatory default interface an extra set of interfaces may be mandatory to support (see Table 4).

Table 4 – Task related FDT-Channel interfaces

FDT channel	Availability	Channel with user interface	Channel of Gateway DTM	Communication DTM for PC/Fieldus adapter
IFdtChannel	Mandatory			
IFdtChannelActiveXInformation	Optional	Mandatory		
IfdtChannelSubTopology	Optional		Mandatory	Mandatory
IfdtChannelSubTopology2	Optional		Mandatory	Mandatory
IfdtCommunication	Optional		Mandatory	Mandatory
IfdtFunctionBlockData			Mandatory	Mandatory
IfdtChannelScan			Mandatory	Mandatory

The mandatory interfaces of a Channel ActiveX® are shown in Table 5.

Table 5 – Task related Channel ActiveX® interfaces

FDT Channel ActiveX® control	Availability
IfdtChannelActiveXControl	Mandatory
IfdtChannelActiveXControl2	Mandatory

The mandatory interfaces of a BTM are shown in Table 6.

Table 6 – Task related BTM interfaces

BTM	Availability
IBtm	Mandatory
IDtmActiveXControlInformation	Mandatory
IDtmChannel	Optional
IDtmDocumentation	Mandatory
IDtmDiagnosis	Mandatory
IDtmImportExport	Optional
IBtmInformation	Mandatory
IDtmInformation2	Mandatory
IDtmOnlineDiagnosis	Mandatory
IDtmOnlineParameter	Mandatory
IBtmParameter	Mandatory
IFdtCommunicationEvents	Mandatory
IFdtCommunicationEvents2	Mandatory
IDtmHardwareIdentification	Mandatory
IDtmSingleDeviceDataAccess	Mandatory
IDtmSingleInstanceDataAccess	Mandatory
IFdtEvents	Mandatory

The mandatory interfaces of a BTM ActiveX® are shown in Table 7.

Table 7 – Task related BTM ActiveX® interfaces

BTM ActiveX® control	Availability
IBtmActiveXControl	Mandatory

Depending on the functionality of a Frame Application, not all interfaces defined for a Frame Application shall be supported (see Table 8).

Table 8 – Task related Frame Application interfaces

Frame Application	Availability	With user interface
IDtmEvents	Mandatory	
IDtmEvents2	Mandatory	
IDtmAuditTrailEvents	Mandatory	
IFdtActiveX	Optional	Mandatory
IFdtActiveX2	Optional	Mandatory
IFdtBulkData	Optional	
IFdtContainer	Mandatory	
IFdtDialog	Mandatory	
IFdtTopology	Mandatory	
IFdtBtmTopology	Mandatory	
IDtmScanEvents	Optional	
IDtmRedundancyEvents	Optional	
IDtmSingleDeviceDataAccessEvents	Mandatory	
IDtmSingleInstanceDataAccessEvents	Mandatory	

Furthermore, the prefixes FDT, DTM and BTM are reserved for identifiers and names defined in the FDT specification. This prevents conflicts of further releases with private extensions of interfaces or definitions.

In general, all FDT interfaces are designed with fieldbus- and manufacturer-neutral methods. Extensions for a new fieldbus are done via new XML schemas. Functional extensions for new tasks will be provided by new interfaces.

5.3 Return values of interface methods

Interface methods indicate success or failure of a method call by well defined return values (marked as [out, retval]). COM errors (HRESULT) shall not be used to return FDT function related errors, except if it is stated in the specification. If no return value is defined (e.g. for all event methods) it is assumed that the method always succeeds.

5.4 Dual interfaces

All interfaces defined within the FDT specification are implemented as dual interfaces. This decision was made to support C++, Visual Basic®, Java®⁷ and other COM compliant development languages. The functionality of an object is implemented in separate task oriented interfaces, so that only the default interface is accessible via the dispatch interface. This prevents marshalling of the extra interfaces to dispatch-only clients, but the extra interfaces can be made available for a dispatch only-client via a wrapper that holds the other interfaces as properties or merges all methods to a single interface.

Due to the better performance, the developer should use the custom interface. However, in general the dispatch interface can be accepted, because the marshalling time of most of the FDT methods can be neglected compared with the runtime of each method.

5.5 Unicode

All string parameters to the FDT interfaces are BSTRs and are therefore UNICODE strings.

Microsoft MIDL Version 3.0 or later is required to correctly compile the IDL code and generate proxy/stub software. Microsoft Windows NT 4.0 (or later), or Windows 95 with DCOM support is required to properly handle the marshalling of FDT parameters.

Note that in order to implement FDT software that will run on both Microsoft Windows NT and Microsoft Windows 95, it is necessary for these components to test the platform at runtime. In the case of Microsoft Windows 95, usually the conversion of any strings to be passed to Win32 from UNICODE to ANSI needs to be done. Visual Basic® makes this conversion implicitly.

The only limitation within this document is that a NUL character (i.e. 0) is only allowed as the last character of any BSTR method parameter to prevent conversion errors (UNICODE<->ANSI, uppercase<->lowercase, etc.) within the database.

5.6 Asynchronous versus synchronous behavior

In general each function call is synchronous. Within FDT there are two special cases of asynchronous behavior.

- After starting the user interface of a DTM, the DTM works asynchronous to the Frame Application. Asynchronous in this case means that the user works with the DTM and the Frame Application is the server for communication and data access. This state ends by a notification to the Frame Application, when the DTM closes the user interface.

⁷ See Annex I.

- While a DTM has opened its user interface, the DTM uses the asynchronous behavior at the communication interface. The time a communication function call needs to return depends on the system topology and the bus protocol and would block an application for seconds. Dividing a communication function call to a request and a response function causes a non-blocking behavior without the pain of multi-threading implementation. The DTM sends its request or several requests without being disturbed by incoming responses. When a response is available, the DTM gets a notification and can receive the response from the communication component. Due to this mechanism, on one hand a DTM should not implement a timeout control and on the other hand the communication has to provide a response for each request. Only a response can contain the timeout information.

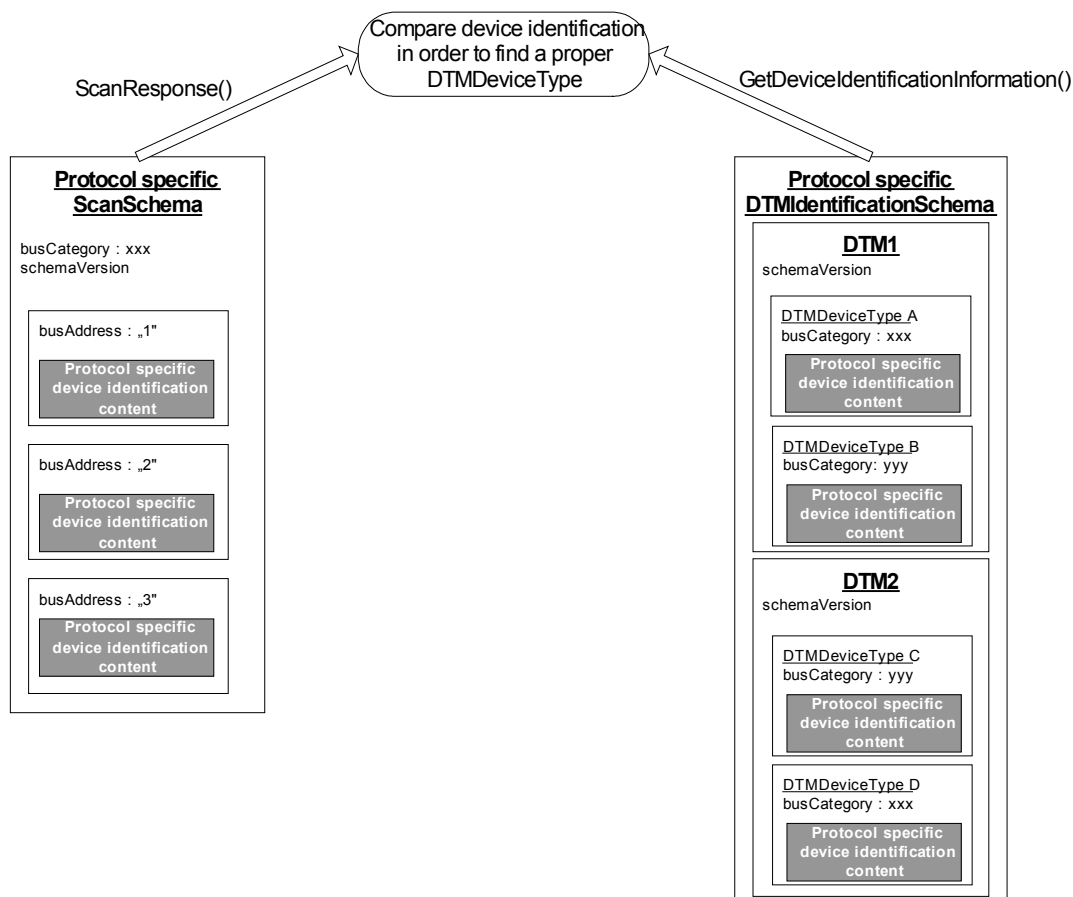
5.7 ProglDs

The usage of proglDs is limited to 39 characters. This decision was made to support C++, Visual Basic®, Java® and other COM compliant development languages.

5.8 Implementation of DTM, DTM device type and hardware identification information

5.8.1 Device identification

Figure 10 shows how the identity information provided by the DTM and the identity information provided as a scan result is converted and used for comparison.

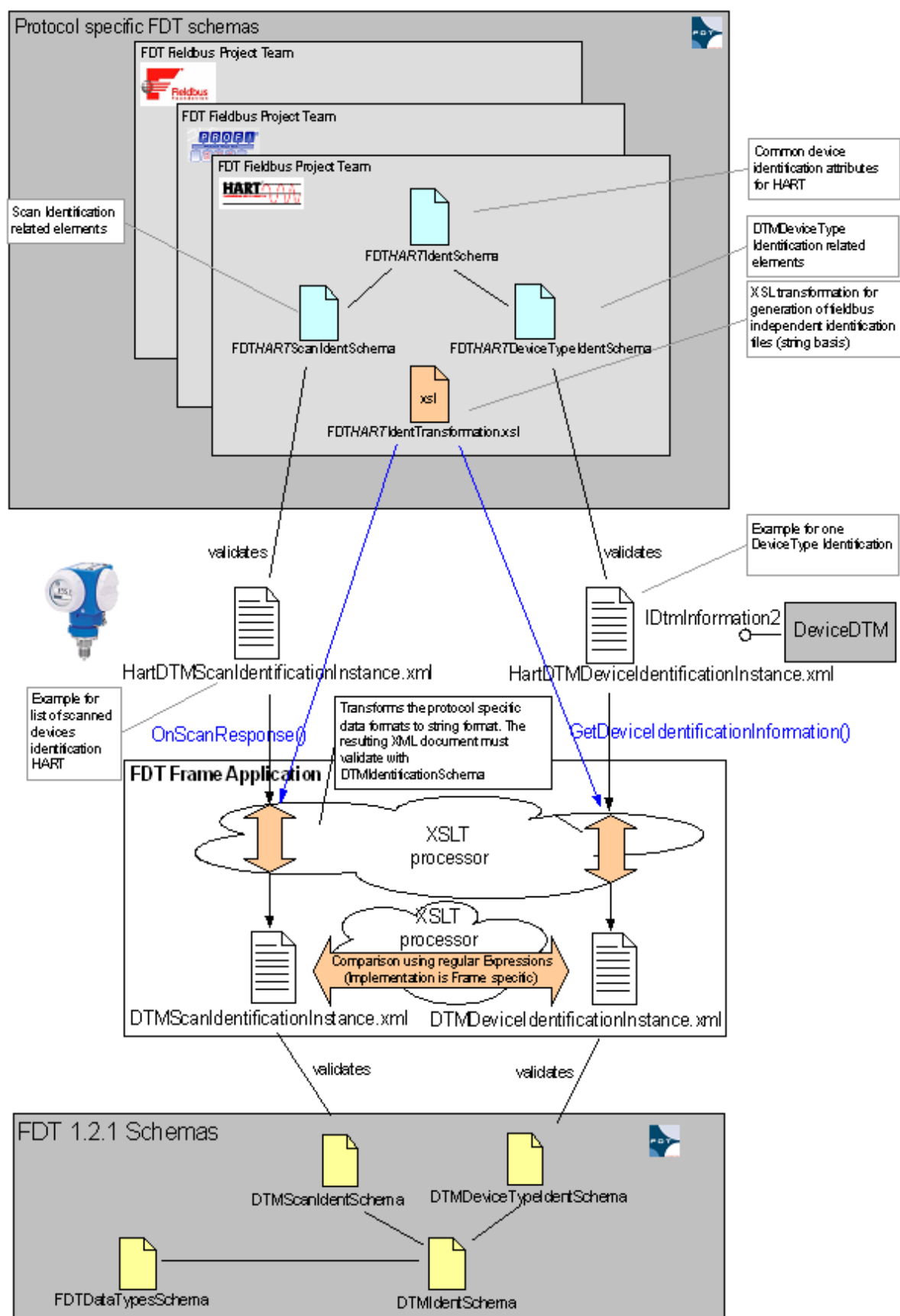


IEC

Figure 10 – Device identification

In order to identify a proper DTMDDeviceType, a Frame Application has to compare identification information of scanned physical devices with identification information of a DTMDDeviceType. These files shall be converted to a fieldbus independent format using a fieldbus-specific XSL transformation.

Figure 11 shows, how protocol-specific schemas are integrated in the FDT specification and used by Frame Application and DTMs (HART® example is shown):



IEC

Figure 11 – Structural overview

5.8.2 Protocol-specific transformation style sheet (xsl)

As shown in the structural overview Figure 11, the protocol-specific FDT specification extension covers a transformation style sheet (.xsl) in addition to schemas. This xsl is to be used by a Frame Application in order to convert the protocol-specific formats included in the identification (scan and DTM) XML documents into strings. The result shall be validated against the protocol independent FDT schemas (DTMScanIdentSchema and DTMDDeviceTypeIdentSchema). The output can be used by a Frame Application to compare scan and DTM values based on string format. A DTM may define a range of supported physical device types by including regular expressions. In order to identify a matching DTMDDeviceType, a Frame Application shall implement a pattern matching according to the regular expression syntax defined in 5.8.5.

5.8.3 Semantic identification information

Table 9 lists identification elements, which have to be provided by the scan and DTM identification mechanism. After xsl transformation, the following values shall be available in string format with the elements listed below in DTMScanIdentSchema and DTMDDeviceTypeIdentSchema.

Table 9 – Semantic identification information

Semantic element name	Description	Scan	DTM
IdAddress	Fieldbus address of the scanned physical device	x	-
IdBusProtocol	Protocol identification (enum)	x	x
IdBusProtocolVersion	Version of bus protocol	(p)	(p)
IdManufacturer	Manufacturer identification	x	?
IdTypeID	Device type identification	x	?
IdSoftwareRevision	Tool relevant version of the physical device – Firmware version	x	?
IdHardwareRevision	Hardware version of the physical device	x	?
IdDeviceTag	Tag name, which set in the physical device or blok	x	-
IdSerialNumber	In order to get a common definition for all kind of protocols, a serial number is defined to be only unique for one manufacturer and device type. For world wide unique identification this attribute shall always be combined with manufacturerID and deviceTypeID.	x	-
IdDTMSupportLevel	Enum: genericSupport, profileSupport, blockspecificProfileSupport, specificSupport(=default)	-	x
Key x shall be provided - is not to be provided ? optional (p) optional, but may be defined mandatory for a specific protocol (see parts 3xy)			

If a semantic element cannot be defined for a fieldbus protocol, the value shall be set to 'NOT_APPLICABLE'.

5.8.4 Device assignment

The comparison of the scan result with DTM device identification information and the device assignment is done by the Frame Application based on its internal rules.

Each element from the scan document can be compared with the DTM device identification information.

The element of the DTM device identification document shall match to the value within the scan document.

5.8.5 Regular expression specification

If the element of the DTM device identification document contains a pattern, the Frame Application shall use regular expressions to compare scan and DTM device identification information (see Table 10).

Table 10 – Regular expressions

MetaCharacter	Description
.	Matches any single character.
[]	Indicates a character class. Matches any character inside the brackets (for example, [abc] matches "a", "b", and "c").
^	If this metacharacter occurs at the start of a character class, it negates the character class. A negated character class matches any character except those inside the brackets (for example, [^abc] matches all characters except "a", "b", and "c"). If ^ is at the beginning of the regular expression, it matches the beginning of the input (for example, ^[abc] will only match input that begins with "a", "b", or "c").
-	In a character class, indicates a range of characters (for example, [0-9] matches any of the digits "0" through "9").
?	Indicates that the preceding expression is optional: it matches once or not at all (for example, [0-9][0-9]? matches "2" and "12").
+	Indicates that the preceding expression matches one or more times (for example, [0-9]+ matches "1", "13", "666", and so on).
*	Indicates that the preceding expression matches zero or more times.
??, +?, *?	Non-greedy versions of ?, +, and *. These match as little as possible, unlike the greedy versions which match as much as possible. Example: given the input "<abc><def>", <.*?> matches "<abc>" while <.*> matches "<abc><def>".
()	Grouping operator. Example: ([0-9]+,)*[0-9]+ matches a list of numbers separated by commas (such as "1" or "1,23,456").
\	Escape character: interpret the next character literally (for example, [0-9]+ matches one or more digits, but [0-9]\+ matches a digit followed by a plus character). Also used for abbreviations (such as \a for any alphanumeric character; see table below). If \ is followed by a number n, it matches the nth match group (starting from 0). Example: <{.*?}>.*?</\0> matches "<head>Contents</head>". Note that in C++ string literals, two backslashes shall be used: "\\+", "\\a", "<{.*?}>.*?</\\0>".
\$	At the end of a regular expression, this character matches the end of the input. Example: [0-9]\$ matches a digit at the end of the input.
	Alternation operator: separates two expressions, exactly one of which matches (for example, T the matches "The" or "the").
!	Negation operator: the expression following ! does not match the input. Example: a!b matches "a" not followed by "b".

5.9 Implementation of slave redundancy

5.9.1 General

Implementation of slave redundancy is defined in IEC 62453-2.

5.9.2 Topology import/export

A Frame Application not aware of DTMs handling redundant slaves is not able to provide redundancy information within a FDT topology export file.

A Frame Application aware of DTMs handling redundant slaves can only add a DTMNode element at ChannelNodes if the appropriate DTM instance has not been added to topology by a IDtmRedundancy::OnAddedRedundantChild() event call. Instead the appropriate DTM element of the topology document should contain a BusInformation element containing the redundant address information.

6 Implementation of FDT services: FDT interfaces

6.1 Overview of the FDT interfaces

The FDT interface specification includes the following:

- DTM
- BTM
- Presentation objects
- DTMAxControl
- BTMAxControl
- FdtChannelAxControl
- FdtChannel
- Frame Application

The behavior of these objects and their interfaces are described in detail in this clause. Developers building FDT objects for DTMs or parts of Frame Application like storage or communication objects shall implement the functionality defined in this clause.

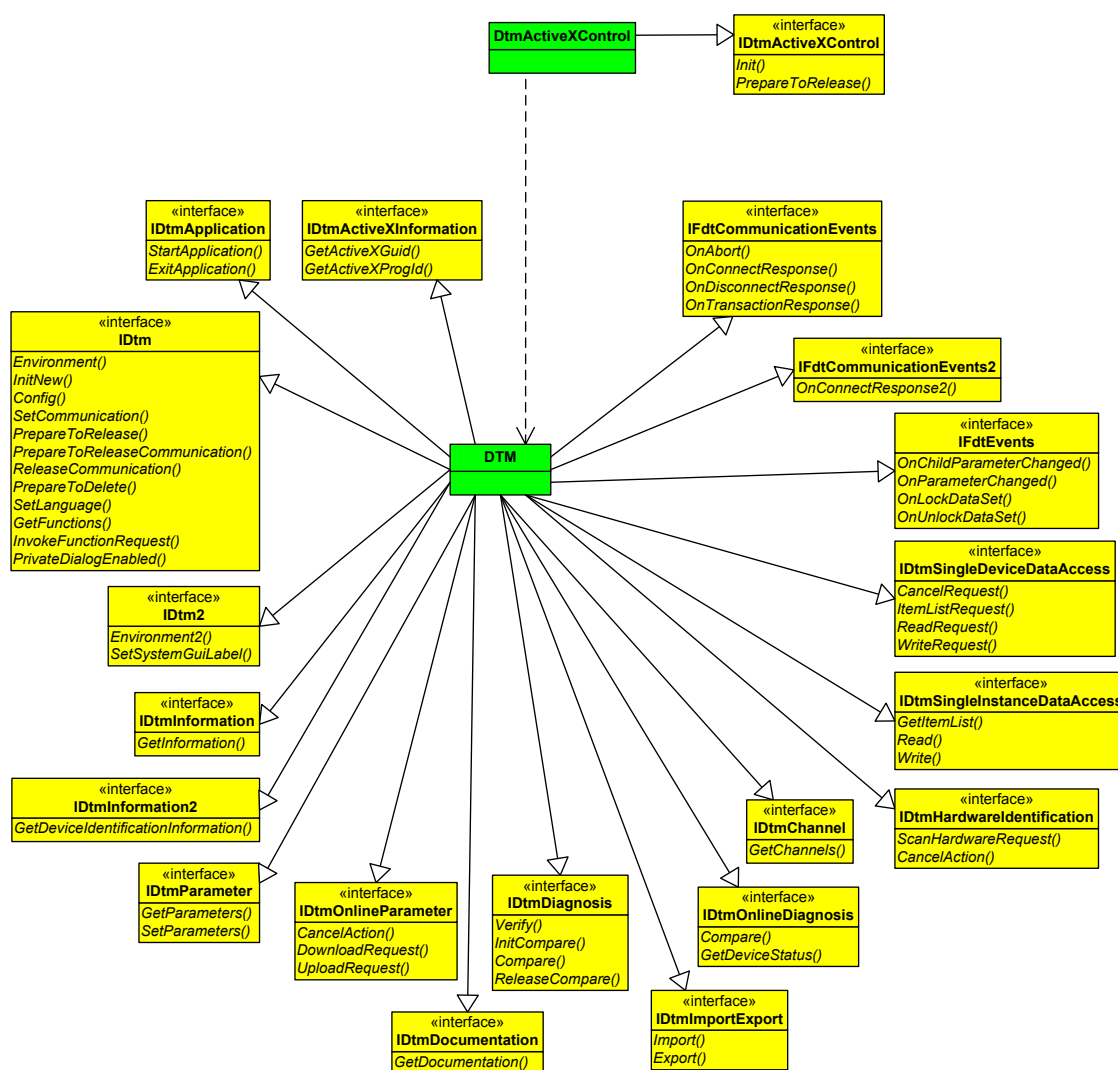
This clause also references and defines expected behavior of both standard COM interfaces and FDT-specific interfaces that FDT compliant objects shall implement.

6.2 FDT objects

6.2.1 FDT object model

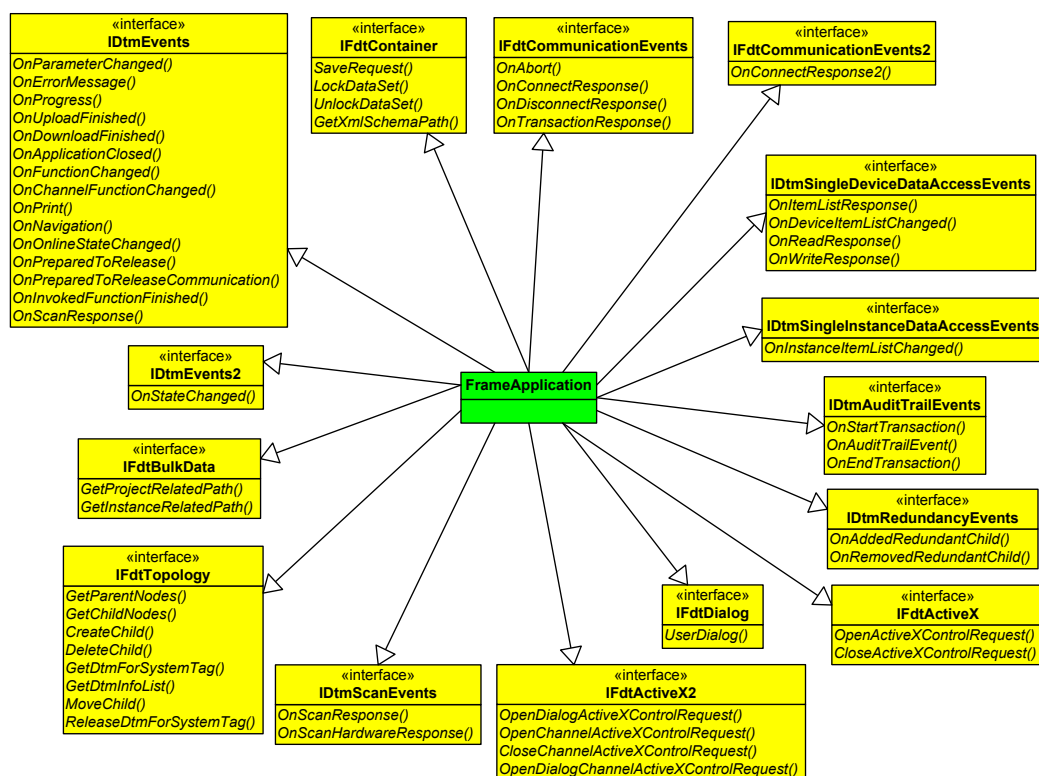
These FDT objects and at least the interfaces represent the tasks for the integration of a field-device-application into a Frame Application. All interfaces of a DTM, of a BTM, of a channel as well as the interfaces of the Frame Application are implemented by one COM object so that a client can access them by calling QueryInterface on one of these interfaces of a server object. So, a client is able to detect availability of optional interfaces of each object during runtime.

The interfaces provided by DTM, ActiveX® and Frame Application are shown in Figure 12 and Figure 13. After the ActiveX® is created by the Frame Application, the DTM cooperates with the ActiveX® object.



IEC

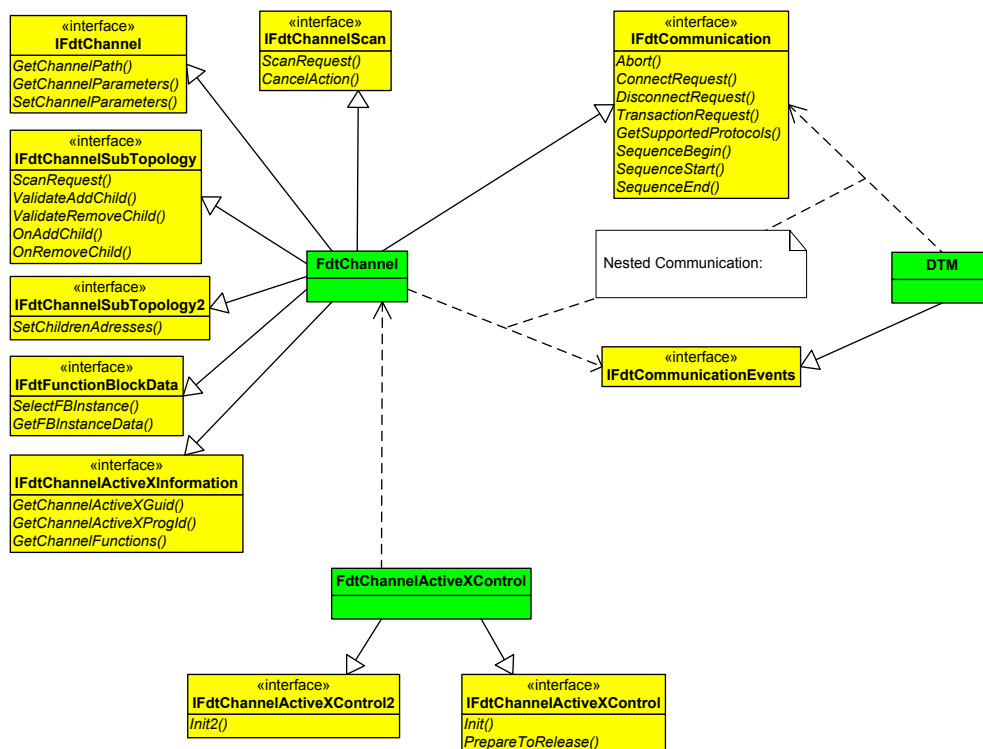
Figure 12 – Interfaces of FDT objects – DTM and DtmActiveXControl



IEC

Figure 13 – Interfaces of FDT object – Frame Application

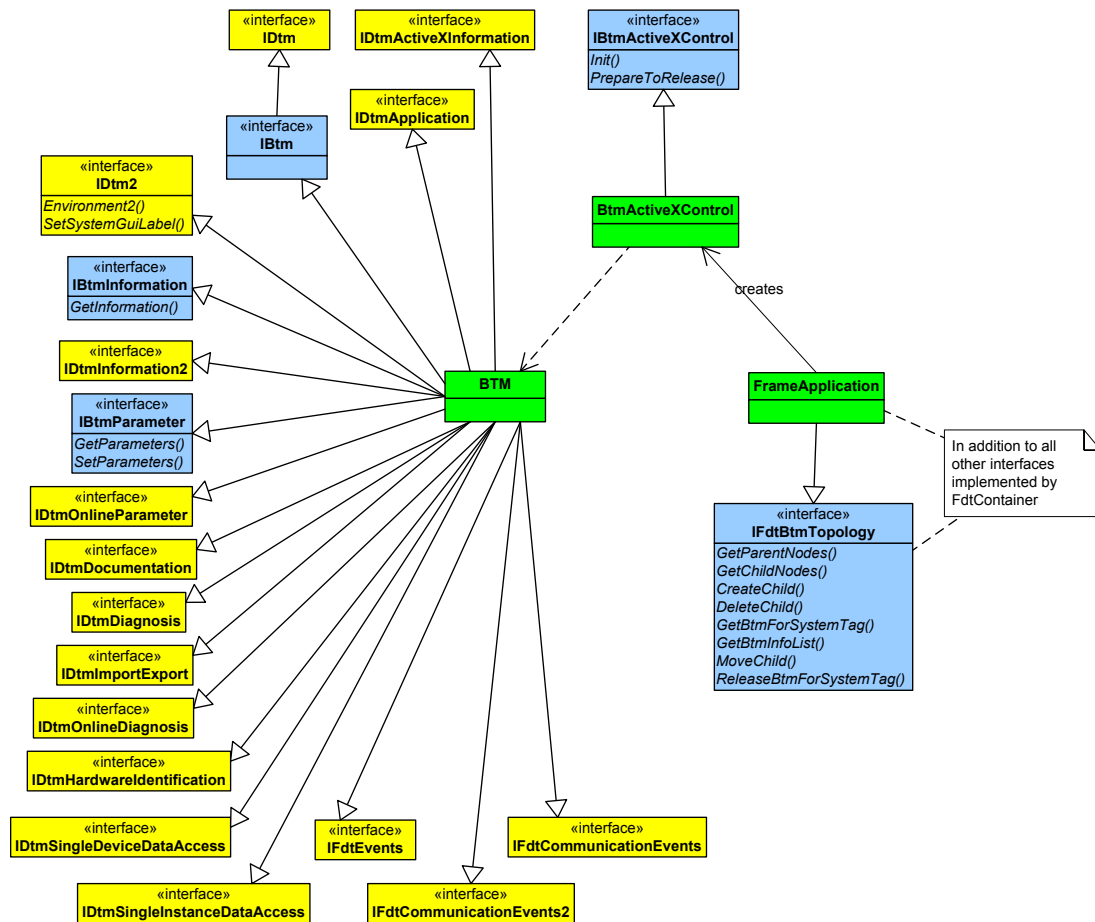
The interfaces provided by a channel are shown in Figure 14. The DTM accesses communication by sending requests to the interface **IFdtCommunication** and receives responses by the interface **IFdtCommunicationEvents**. A hierarchy of channels and DTMs is used to provide nested communication.



IEC

Figure 14 – FDT objects – FDT-Channel

The interfaces related to a BTM are shown in Figure 15.



IEC

Key

indicates interfaces specific for use with BTMs.

Figure 15 – FDT objects – BTM and BtmActiveXControl

6.2.2 Availability of interface methods

The availability of interface methods of the different objects may depend on the state of a DTM instance. This dependency is defined in Table 11 and Table 12.

Table 11 defines the interfaces of a DTM which can be used by a Frame Application at the shown states.

Table 11 – Availability of DTM methods in different states

Interface / Method	up	new created	new enabled	existing created	running	configured	going offline	online	going online	destroyed	communication set	zombie
IPersistXXX												
InitNew()	√											(√)
Load()	√											(√)
Save()						√	√	√	√		√	
IDtm												
Environment()		√		√								
InitNew()			√									
Config()					√							
SetCommunication()						√						
ReleaseCommunication()											√	
PrepareToDelete()						√						
PrepareToRelease()			√			√				√		
SetLanguage()			√		√	√					√	
InvokeFunctionRequest()						√	√	√	√		√	
PrepareToReleaseCommunication()							√	√	√		√	
PrivateDialogEnabled()						√	√	√	√		√	
GetFunctions()					√	√	√	√	√		√	
IDtm2												
Environment2()		√		√								
SetSystemGuiLabel()			√		√	√	√	√	√		√	
IDtmActiveXInformation					√	√	√	√	√		√	
IDtmApplication						√	√	√	√		√	
IDtmChannel					√	√	√	√	√		√	
IDtmDocumentation						√	√	√	√		√	
IDtmDiagnosis						√	√	√	√		√	
IDtmInformation			√		√	√	√	√	√		√	
IDtmInformation2			√		√	√	√	√	√		√	
IDtmImportExport						√						
IDtmOnlineDiagnosis								√	√		√	
IDtmOnlineParameter								√	√		√	
IDtmParameter						√	√	√	√		√	
IDtmHardwareIdentification							√	√	√		√	
IFdtCommunicationEvents							√	√	√		√	
IFdtCommunicationEvents2							√	√	√		√	
IFdtEvents					√	√	√	√	√		√	
IFdtChannel					√	√	√	√	√		√	

Interface / Method	up	new created	new enabled	existing created	running	configured	going offline	online	going online	destroyed	communication set	zombie
IFdtChannelActiveXInformation					√	√	√	√	√		√	
IFdtChannelSubTopology												
OnAddChild()					√	√	√	√	√		√	
OnRemoveChild()					√	√	√	√	√		√	
ScanRequest()								√	√		√	
ValidateAddChild()					√	√	√	√	√		√	
ValidateRemoveChild()					√	√	√	√	√		√	
IIFdtChannelSubTopology2												
SetChildrenAdresses()					√	√	√	√	√		√	
IFdtChannelScan												
ScanRequest()								√	√		√	
CancelAction()								√	√		√	
IFdtCommunication												
Abort()							√	√	√		√	
ConnectRequest()							√	√	√		√	
DisconnectRequest()							√	√	√		√	
TransActionRequest()							√	√	√		√	
GetSupportedProtocols()						√	√	√	√		√	
SequenceBegin()							√	√	√		√	
SequenceStart()							√	√	√		√	
SequenceEnd()							√	√	√		√	
IFdtFunctionBlockData						√	√	√	√		√	
IDtmSingleInstanceDataAccess						√	√	√	√		√	
GetItemList()						√	√	√	√		√	
Read()						√	√	√	√		√	
Write()						√	√	√	√		√	
IDtmSingleDeviceDataAccess												
CancelRequest()							√	√	√		√	
ItemListRequest()						√	√	√	√		√	
ReadRequest()								√	√		√	
WriteRequest()								√	√		√	
At the Zombie State not all DTMs shall support reload instance via IPersistXXX interfaces, e.g. DTMs written in Visual Basic®. Concerning the transition between states and methods with asynchronous behavior: The call of methods, which are defined with an asynchronous behavior (e.g. PrepareToRelease()) will start the transition. The related end state will be reached, when the according method was called (e.g. OnPreparedToRelease()). Concerning the transition between states in case of errors: If the method, which leads to the transition between states, fails (e.g. return value is FALSE or a COM error appears), the state is left unchanged.												

Table 12 defines the interfaces of a Frame Application which can be used by a DTM at the shown states. A Frame Application shall be aware that after IDtm:Environment a DTM complying to older versions of FDT can call any Frame Application interface method.

Table 12 – Availability of Frame Application interfaces

Interface / Method	up	new created	new enabled	existing created	running	configured	going offline	online	going online	destroyed	communication set	zombie
IFdtContainer												
GetXMLSchemaPath()			√	√	√	√	√	√	√		√	
LockDataSet()					√	√	√	√	√		√	
SaveRequest()					√	√	√	√	√		√	
UnlockDataSet()					√	√	√	√	√		√	
IDtmEvents												
OnApplicationClosed()					√	√	√	√	√		√	
OnDownloadFinished()								√			√	
OnErrorMessage()			√	√	√	√	√	√	√		√	
OnFunctionChanged()					√	√	√	√	√		√	
OnChannelFunctionChanged()					√	√	√	√	√		√	
OnInvokedFunctionFinished()					√	√	√	√	√		√	
OnNavigation()						√	√	√	√		√	
OnOnlineStateChanged()							√	√	√		√	
OnParameterChanged()					√	√	√	√	√		√	
OnPreparedToRelease()												√
OnPreparedToReleaseCommunication()							√	√	√		√	
OnPrint()					√	√	√	√	√		√	
OnProgress()						√	√	√	√		√	
OnScanResponse()							√	√			√	
OnUploadFinished()								√			√	
IDtmAuditTrailEvents						√	√	√	√		√	
IFdtActiveX					√	√	√	√	√		√	
IFdtActiveX2					√	√	√	√	√		√	
IFdtBulkData			√		√	√	√	√	√		√	
IFdtDialog					√	√	√	√	√		√	
IFdtTopology						√	√	√	√		√	
IDtmSingleDeviceDataAccessEvents												
OnItemListResponse()						√	√	√	√		√	
OnDeviceItemListChanged()						√	√	√	√		√	
OnReadResponse()								√	√		√	
OnWriteResponse()								√	√		√	
IDtmSingleInstanceDataAccessEvents						√	√	√	√		√	

6.3 Device Type Manager

6.3.1 Interface IDtm

6.3.1.1 General

This interface is the main interface of a DTM according to row one of Table G.1. Via this interface the DTM gets its initialization after the instantiation.

The Frame Application uses the interface to set information, like communication interface or language, the DTM needs during runtime as well as to reset the DTM for release.

At this time the DTM is not connected to an instance data set of a device. At this state, the DTM can be asked for its static information like version, vendor and its capabilities.

If the DTM is initialized it can be asked for its instance independent supported functions.

6.3.1.2 Config

```
HRESULT Config(
    [in] FdtXmlDocument userInfo,
    [out, retval] VARIANT_BOOL* result);
```

Description

Is called by the Frame Application for initialization concerning the current user.

The method is part of the implementation of the Initialize service as defined in IEC 62453-2.

Parameters	Description
userInfo	XML document containing the current user rights and the user role specified by the FDTUserInformationSchema.

Return value

Return value	Description
TRUE	DTM accepted the given data.
FALSE	The operation failed.

Behavior

It informs the DTM during initialization about the role and the rights of the current user.

Comments

In general it is expected that a DTM adapts the provided functionality according to the role of the current user (see 6.3.1.4).

6.3.1.3 Environment

HRESULT Environment(
 [in] BSTR systemTag,
 [in] IFdtContainer* container
 [out, retval] VARIANT_BOOL* result).

Description

Is called by the Frame Application to set the systemTag and the pointer to the Frame Application.

The method is part of the implementation of the initialize service as defined in IEC 62453-2.

Parameters	Description
systemTag	Identifier for the device instance; set by the Frame Application
Container	Back pointer to the Frame Application

Return value

Return value	Description
TRUE	DTM has accepted the data.
FALSE	The operation failed.

Behavior

Is called by the Frame Application to initialize a DTM for a device instance. Furthermore the Frame Application passes the pointer to its own main interface.

Comments

The systemTag is independent of communication tags (e.g. IEC 61784 CPF 9 device tag).

6.3.1.4 GetFunctions

HRESULT GetFunctions(
 [in] FdtXmlDocument operationState,
 [out, retval] FdtXmlDocument* result);

Description

Returns an XML document containing information about standard (defined by applicationID) or additional functionalities (defined by functionId) and documents supported by a DTM.

The method is one implementation of the GetFunctions service as defined in IEC 62453-2.

Parameters	Description
operationState	XML document containing the current operation phase specified by the FDTOperationPhaseSchema.

Return value

Return value	Description
Result	XML document containing actual supported functions specified by the DTMFunctionsSchema.

Behavior

Functions with user interface are started via `IDtmApplication::StartApplication()`, `IDtmActiveXInformation::GetActiveXGuid()` or `IDtmActiveXInformation::GetActiveXProgId()` and work asynchronous.

Comments

The XML document provided by `IDtm::GetFunctions()` can contain functions (`<Function>` or `<StandardFunction>` elements) and groups of functions (`<Functions>`). Both can separately be enabled/disabled or shown/hidden, described by a `<Status>` element.

Because the FDT standard does not define status inheritance, the group (`<Functions>`) status is not inherited by the children (`<Function>` or `<StandardFunction>`). The DTM shall take care about the status of sub-functions.

For example, if the DTM disables a group of functions (`<Functions>`), it shall also disable all functions (`<Function>`) below that group if this is the intended behavior. If the DTM does not disable sub-functions, the Frame Application can still make use of them.

If the DTM has set the attribute 'isPrintable' of a `<Function>` element to true, the Frame Application should offer printing, even if attribute 'isEnabled' of the same `<Function>` element is set to false.

Microsoft Windows® supports that menu functions may be called from the keyboard via their mnemonic access characters. An ampersand ('&') in a menu item string is normally translated into an underline character and used as the mnemonic access character for that menu item.

Because of different FDT Frame Application-specific presentations (e.g. drop-down menu, listbox or combo box) mnemonic access characters should not be used within the XML document provided by `IDtm::GetFunctions()`.

6.3.1.5 InitNew

```
HRESULT InitNew(
    [in] FdtXmlDocument deviceType,
    [out, retval] VARIANT_BOOL* result);
```

Description

Is called by the Frame Application to initialize a newly created instance data set for a specific device type.

The method is part of the implementation of the initialize service as defined in IEC 62453-2.

Parameters	Description
deviceType	XML document containing the manufacturer-specific data like unique identifier for a sub-device type specified by <code>DTMInitSchema</code> .

Return value

Return value	Description
TRUE	DTM is initialized.
FALSE	The operation failed.

Behavior

The Frame Application initializes the DTM for a specific device-type. The supported device types of a DTM are available via `IDtmInformation::GetInformation()`. This initialization is necessary especially for a DTM that supports more than one device type.

Comments

None

6.3.1.6 InvokeFunctionRequest

```
HRESULT InvokeFunctionRequest(  
    [in] FdtUUIDString invokeId,  
    [in] FdtXmlDocument functionCall,  
    [out, retval] VARIANT_BOOL* result);
```

Description

Starts a function of a DTM.

The method is one implementation of the `InvokeFunction` service as defined in IEC 62453-2.

Parameters	Description
invokeId	Identifier for the started function.
functionCall	XML document containing the DTM-specific function id for the requested function or user interface specified by the <code>DTMFunctionCallSchema</code> .

Return value

Return value	Description
TRUE	The function started.
FALSE	The function call failed.

Behavior

See IEC 62453-2.

Comments

None.

6.3.1.7 PrepareToDelete

```
HRESULT PrepareToDelete(  
    [out, retval] VARIANT_BOOL* result);
```

Description

Returns TRUE if the device instance data set can be deleted at the Frame Applications database. Used to inform the DTM that it has to clean up e.g. its log files or protocols. The data set will be deleted by the Frame Application.

The method is one implementation of the `ClearInstanceData` service as defined in IEC 62453-2.