IEC/TR 62795

Edition 1.0 2013-09

TECHNICAL REPORT

colour inside

Interoperation guide for field device tool (FDT) / device type manager (DTM) and electronic device description language (EDDL)

IEC/TR 62795:2013(E)

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

Useful links:

IEC publications search - www.iec.ch/searchpub

The advanced search enables you to find IEC publications by a variety of criteria (reference number, text, technical committee,…).
It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished

Stay up to date on all new IEC publications. Just Published details all new publications released. Available on-line and also once a month by email.

Electropedia - www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 30 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary (IEV) on-line.

Customer Service Centre - webstore.iec.ch/csc

If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: csc@iec.ch.

IEC/TR 62795

Edition 1.0 2013-09

TECHNICAL REPORT

colour inside

Interoperation guide for field device tool (FDT) / device type manager (DTM) and electronic device description language (EDDL)

INTERNATIONAL ELECTROTECHNICAL COMMISSION

PRICE CODE U

ICS 25.040.40; 35.240.50

ISBN 978-2-8322-1116-8

Warning! Make sure that you obtained this publication from an authorized distributor.

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

INTEROPERATION GUIDE FOR FIELD DEVICE TOOL (FDT) / DEVICE TYPE MANAGER (DTM) AND ELECTRONIC DEVICE DESCRIPTION LANGUAGE (EDDL)

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 62795, 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/240/DTR	65E/330/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.

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

At present, there are two International Standards for device integration that describe the properties of automation system components to be used in host systems. They are IEC 61804 for electronic device description language (EDDL) and IEC 62453 for field device tools (FDT), with both standards having their own characteristics. The number of manufacturers and users using EDDL or FDT technologies is increasing, and investment in both of them is therefore increasing too.

EDDL technology enables the integration of real product details using the tools of the engineering life cycle and specifies EDDL as a generic language for describing the properties of automation system components. EDDL technology allows to transfer the properties of a device to a data set, called electronic device description (EDD), that can be interpreted by configuration tools in a host system. EDD files, representing the behavior and attributes of a device, can be stored in the field device or control system. The interaction between a field device and a control system uses various communication protocols such as specified in IEC 61784-1, CP 1-1 (FF) or IEC 61784-1, CP 9-1 (HART®[1]).

FDT is an open and independent software interface specification. An FDT specification describes software interface and relationships within the framework of the project (FDT container, FDT frame applications) and device software components in a Device Type Manager (DTM). An FDT framework is independent from the devices and fieldbus system, while the DTM depends on specific device and fieldbuses.

Both technologies are supported by automation vendors and users, so that there is a need to generate a DTM based on an EDD as long as the field device integration (FDI) technology is not published as an International Standard.

FDI (IEC 62769) covers device integration and device management technology, combining base concepts and technology aspects of the EDDL (IEC 61804), FDT (IEC 62543) and OPC UA (IEC 62541-1). The combination of those different proven technologies ensures a secure life cycle and the ability to address all challenges of device integration and device management in a scalable manner.

EDDL and FDT are complimentary in a way that an EDDL can be converted into an FDT-DTM. As long as FDI is not available, converting EDD into a DTM helps to combine the two standards and use EDDs in an FDT frame environment in case there is no specific DTM available. The conversion criterion is based on EDDL and FDT technologies.

1 HART® is an example of a suitable product available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of this product.

INTEROPERATION GUIDE FOR FIELD DEVICE TOOL (FDT) / DEVICE TYPE MANAGER (DTM) AND ELECTRONIC DEVICE DESCRIPTION LANGUAGE (EDDL)

1 Scope

This Technical Report provides the general requirements for converting an EDD into a DTM. Using this TR, an FDT/DTM developer can develop an EDD-DTM conversion tool that can be used to import, parse, and manage EDD to generate the corresponding DTM. A conversion tool versus a DTM written independent of an EDD helps the DTM generation to maintain consistency in function, data and presentation styles.

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 61804 (all parts), *Function blocks (FB) for process control*

IEC 61804-3:2010, *Function blocks (FB) for process control – Part 3 Electronic Device Description Language (EDDL)*

IEC 62453-1:2009, *Field device tool (FDT) interface specification – Part 1: Overview and guidance*

3 Terms, definitions, and abbreviations

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 62453-1, IEC 61804-3, as well as the following apply.

3.1.1
agent interpreter
analytical software for parsing EDD

Note 1 to entry: An agent interpreter can parse out the elements in the EDD document and map to the DTM.

Note 2 to entry: An agent interpreter can realize the interaction between the DTM and the physical device.

3.1.2
application
software functional unit that is specific to the solution of a problem in industrial-process measurement and control

Note 1 to entry: An application may be distributed across multiple resources, and may communicate with other applications.

[SOURCE: IEC 61499-1:2005, 3.5]

3.1.3
device type manager
DTM
device-specific application software components

3.1.4
field device tool
FDT
interfaces for data exchange among the field devices and control systems, engineering tools, and resource management system tools

3.2 Abbreviations

BTM	Block type manager
COM	Component object model
CP	Communication profile
DCS	Distributed control system
DD	Device description
DLL	Dynamic link library
DTM	Device type manager
EDD	Electronic device description
EDDL	Electronic device description language
ERP	Enterprise resource planning
FA	Frame application
FB	Function block
FDI	Field device integration
FDT	Field device tool
FF	Fieldbus Foundation
GUI	Graphical user interface
I/O	Input/output
ID	Identifier
IT	Information technology
MES	Manufacturing execution systems
OLE	Object linking and embedding
OPC	Open connectivity via open standards
PLC	Programmable logic controller
SCADA	Supervisory, control and data acquisition
UML	Unified modeling language
UUID	Universal unique identifier
XML	Extensible markup language

4 Typical methods to convert EDD to FDT/DTM

4.1 Overview

There are three typical methods to integrate EDD documents into FDT:

- Using a universal DTM to integrate EDD documents.

- Using a converting tool importing an EDD document to generate a DTM.
- Developing a DTM based on an EDD.

4.2 Using a universal DTM to integrate the EDD document

4.2.1 General

FDT/DTM developers provide a universal DTM, which contains an EDD interpreter. The implementation of the DTM is very flexible, so an EDD interpreter can be embedded into the COM component of the DTM. The interpreter can be implemented in the DTM interface to not change the external form of the DTM. When the host sends a request to this universal DTM server, the DTM server acquires the information on the requested data by looking for a data sheet generated by the interpreter, which then bases itself on this information to communicate directly with the DTM through a communication device. When the DTM is running, the EDD can be imported, the interpreter analyses the documents, and then generates the functions and data according to the EDD. The corresponding functions and data are released after the FA releases the DTM. Before the next time it is used, the file needs to be imported and treated accordingly. Since the universal DTM needs to parse different EDD files, and generate a corresponding DTM, the EDD file format is very essential The EDD file format needs to be defined in order to parse correctly. If the EDD file format is not unified, the interpreter cannot parse or not exactly as the analytical results are given, and as a result, it will not generate a DTM or generated DTMs will be different, affecting consistency.

The procedure for using a universal FDT/DTM to integrate the EDD document is as follows:

1) Users start updating the list of FDT application devices, more precisely, the list of DTM device types.
2) The FDT application instantiates the general device FDT, calling the DTM interface functions IDtmInformation::GetInformation or calling the DTM interface functions IDtmInformation2::GetDeviceIdentificiationInformation.
3) The universal DTM reads all the EDD files of a predefined path, parsing EDD file type information such as the protocol which the devices support, manufacturer identification, manufacturer name, device type ID, device name, EDD file path names.
4) The universal DTM writes the information into XML documents according to XML schema defined by the FDT, and returns the XML documents back to the FDT application. The XML documents can contain more than one device type information.
5) The FDT frame application generates a list of device types according to XML documents. At this point the devices' name defined in the EDD files will appear in the list.
6) Users create a new project in the FDT application, and select a communication DTM type (i.e. an instance of a communication DTM) for the project.
7) Users select a type of universal DTM from the list of DTM device types (universal DTM) and add it into communication DTM as a sub-project.
8) During the instancing of a universal DTM, the FDT frame application calls a DTM interface function IDtm:InitNew. An input parameter of the function is an XML document, which contains the device type information of a specific DTM. Since the information contains the EDD file path information, the universal DTM reads one EDD file according to this information and parses this file then generates the corresponding internal data model or internal database according to the parameters of this EDD.
9) The communication DTM calls the interface function of the universal DTM IDtmParamenter::GetParamenters to obtain information associated with the parameters of the device.

4.2.2 Basic contents of the transformation from EDD to DTM

The transformation steps could be as follows:

1) Defining the transformation model

This requirement will be based on the EDDL specification to design a conversion model. The model needs to achieve the following contents: parameter type, communication mode, method, menu and other functions.

The characteristics of the DTM need to conform to the specification according to the actual requirements, and the requirements of being able to provide consistent data and functionality.

The features of the DTM need to demonstrate the data and functions which are involved in EDDL. It can support only some of the features in EDDL. The usual functions are: the displaying and setting of parameters, reading and writing parameters, user interface and so on.

2) The specifications of the DTM interface, i.e. it is required to be able to display the EDD interface correctly.

   The EDD's interface is defined by the menu, so it is better to correspond with the EDDL menu when designing the DTM interface, and that will maintain the consistency of the interface.

3) After the specifications of how describing to resolve the EDD document, the interaction between the instance and field devices as well as between the instance and the framework should be handled.

   The interaction between the instance and field devices needs to communicate with the hardware, which requires a communication DTM to provide communication services. During the interaction, an FDT FA provides a standardized method to handle the interaction between Framework applications and each instance.

4) The specification of object mapping between EDDL and FDT

   Object mapping is divided into two steps. First, it is required to map the elements of the table, such as TYPE, ADDRESSBLOCK parsed from the EDD document, to the member variables of the device object, and map the METHOD COMMAND to the member functions of device object. Second, it is required to map member variables and methods of the device object into FDT.

5) Guarantee that DTM can well manage the EDD document loaded in it well.

4.2.3 Procedure

The procedure for the transformation from EDD to DTM is shown in Figure 1.

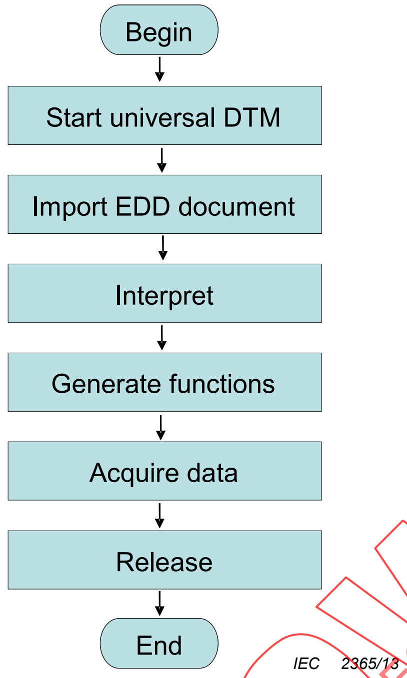

Figure 1 – EDD to DTM conversion flowchart

The general process of converting EDD to DTM is as follows:

1) Start universal DTM

   There should be a universal DTM in the application framework, which contains the internal implementation of EDD resolution. Before importing the EDD, you need to load the DTM in the framework of the application.

2) Import the EDD

   Import the EDD into a universal DTM which has implemented the function of EDD interpreter.

3) Interpret EDD

   EDD files written using EDDL according to IEC 61804 should parse out all the information in the EDD document.

4) Generate functions

   According to EDDL standards, it is required to achieve all the functions that an EDD contains during the course of parsing an EDD into the DTM. These functions include how to display and set parameters, how to read and write parameters, the user interface and so on.

5) Acquire data

   According to IEC 61804 it is required to give the corresponding data of parameters described by EDD during the parsing of the EDD to generate a DTM.

6) Release

   This universal DTM using an EDD interpreter on the fly can be generally useful for host system vendors as a part of their host system because all devices that have EDD provided by manufacturers are supported in the FDT host system.

4.3 Using a converting tool to integrate EDD documents

4.3.1 General

There will be a direct generation of the corresponding DTM after the FDT/DTM developers or users parse through the EDD file. This DTM is independent of the EDD files and can be used alone. However, this approach often requires the use of a specialized conversion tool. With the support of this conversion tool, we can import the EDD file and do the associated analytical processing, to get a separate DTM which can run thanks to the conversion tool. This method also requires to do format standardized work for the EDD file, in order to parse the file for the conversion tool and generate the corresponding DTM. In this method, a device DTM can be generated from one or more than one file, and this DTM can represent more than one device type. The device type information and corresponding devices' parameter information can exist in the binary DLL files of the DTM in an entirely hard-coded format.

4.3.2 Basic contents

The basic contents of EDD to DTM transformation are the following:

1) Defining the transformation model

   This requirement will be based on EDDL specification to design a conversion model. This model needs to achieve the following contents: the parameter type, communication mode, method, menu and other functions.

2) Defining the intermediate object model

   The intermediate conversion process of the basic object model may be defined using XML or UML. EDDL documents are loaded into the object model to generate the intermediate objects.

3) Mapping intermediate objects to FDT objects

   Variables, methods, GUI elements, etc. are seen as intermediate objects. Finally, they are mapped to FDT objects and implemented in DTM.

4) Managing EDD documents

   During the preparing, starting and finishing process of the conversion, it is required to read EDD documents, to generate related objects and to release the reference, without changing the content of EDD documents. Intermediate documents may be saved or not. It depends on the vendors, and is implementation specific.

4.3.3 Procedure

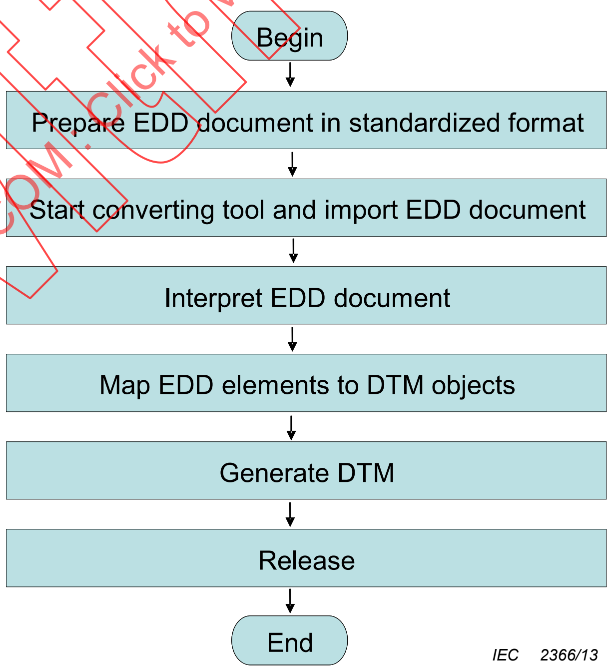

Figure 2 – EDD to DTM general process of conversion flowchart

The general process of converting EDD to DTM shown in Figure 2 is as follows:

1) Prepare EDD document in standardized format

The EDD document format should meet the requirement of the converting tool. This work can be done by hand or automatic software tools, and carried out by vendors or the third parties. The converting tool can only work with the well formatted EDD documents, and generate corresponding objects.

2) Start converting tool and import EDD document

   Import the EDD document, prepared in standardized format, into the converting tool. The converting tool takes the charge of verifying the integrity, reading file data, checking errors, unzipping or decrypting if necessary, and interpreting EDD document.

   The conversion tool parses the elements contained in the EDD document, such as variables, methods, GUI and so on.

3) Map EDD elements to DTM objects

   The elements obtained through the EDD document interpreting is mapped to DTM objects. The mapping methods depend on the vendors of the converting tool, but have to guarantee the consistence.

4) Generate DTM

   The converting tool needs to arrange objects into DTM. The DTM finally generated should have the capability as EDD document specified and runs according to FDT specification. Once the DTM comes into play, it can run independently and have nothing to do with the original EDD document.

4.4 Generate an FDT/DTM based on EDD documents

FDT/DTM developers develop DTM completely in accordance with the EDD document. In the end, they can develop a DTM similar to a general one. The new developed DTM can run independently, and it is out of the management of EDD. This method usually requires developers to understand the EDD. This specification does not contain such content.

5 General requirements for the conversion from EDD to FDT

5.1 Overview

Clause 5 gives the various requirements for integrating EDD into FDT and provides a guide to convert EDD document to related DTM. It gives consistent functions on the description of the basic parameters, business logic, error processing, storage management, interface styles after converting EDD into FDT. The conversion of an EDD file into a DTM file provides a good integration of both technologies, so that just one technology has to be handled in the systems.

5.2 Requirements for the agent interpreter

5.2.1 General

The agent interpreter is the interpreter software which is used to parse the EDD. The main task of the agent interpreter is to interpret the elements in the EDD and to realize the mapping between EDD and DTM. According to the different conversion methods used, the position of the agent interpreter may be different. According to 4.2, the agent interpreter can be located in the universal DTM or as part of the source code. It also can be used as stand-alone COM component in the DTM outside, called by the DTM. From the perspective of standardization, it would be better to develop the interpreter as a separate part independent from the DTM, and to unify its interface and function for the DTM to call it. For the method described in 4.3, the interpreter itself can be implemented in the same way as the methods in 4.2.

An EDD agent interpreter is the core of the integration which is from EDD to FDT. The contents of the generated DTM are based on analytical results that the agent interpreter parses out from the EDD. Therefore, when the agent interpreter parses the EDD, the consistency of the results should be given according to the standard.

Different software providers of DTM can develop different EDD interpreters. In theory, they are all based on the same EDD standard and the functions should be more uniform. But in the actual software development process, different software providers make a choice according to the different needs and constraints of the development cycle. So the agent interpreter may have significant differences in function. Therefore, the standardization of the agent interpreter is necessary, at least to specify which functions are mandatory and which are optional.

Table 1 – EDD basic elements

Categories of elements	Name of the elements
Identification and version information	MANUFACTURER DEVICE_TYPE DEVICE_REVISION EDD_REVISION
Data description	VARIABLE LABEL LABEL HELP TYPE CLASS DEFAULT_VALUE ACTIONS ARRAY MINMAX_VALUE UNIT ACTIONSARRAY ,ITEM_ARRAY BLOCK RECORD COLLECTION LIST FILE …
User interface description	MENU WINDOW DIALOG PAGE,GROUP TABLE,GRID IMAGE CHART GAUGE SCOPESTRIP SWEEP GRAPH YT,XY ACTIONS(METHODs) …
Communication description	COMMAND Data ordering Bit-masks and-positioning Upload and download of office and online configurations Ordering of COMMANDS Control of time conditions Error handling and error messages Relative and absolute addressing…

EDDL is text-based electronic device description language which is not dependent on the system platform. EDDL provides the semantic to describe the identification information, data description, user interface description and communication protocol definition in an EDD that can be located inside the device as listed in Table 1.

Identification and version information: includes the general definition of FDT/DTM developers, manufacturers (MANUFACTURER), device type (DEVICE_TYPE) and version information equipment and so on.

Data description: defines the different data types of field data or abstract representation of the device through the definition of floating point, integer, byte, bit, date, time, text, and different data types. It can be used for the production of data, the configuration of parameters and the description of processing data. The data description can be expressed by different forms of organization such as variables (VARIABLE), record (RECORD) and array (ARRAY).

User interface description: defines the menu, window, table, chart and other visual elements of the description. It gives a visual representation of the hierarchy of the man-machine interactive interface. According to the description of the user interface, the application system can automatically generate an interactive interface to facilitate the access to the field device.

Communication description: mainly used to describe message processing mechanisms in the communication process of the system, including the communication task, fault reporting and related addressing information, message sequence and description of the timing conditions. The interfaces of DTM are as shown in Table 2.

Table 2 – The related interfaces of DTM

The categories of the interfaces	The name of the interfaces
Default interfaces	IDtmInformation IDtm
Access to device parameter	IDtmParameter IDtmOnlineParameter IDtmChannel
Device documentation	IDtmDocumentation
Load/save data	IPersistStreamInit IPersistPropertyBag IDtmExportImport
Start of the user interfaces	IDtmActiveX Information IDtmApplication
Diagnosis functions	IDtmDiagnosis IDtmOnlineDiagnosis
Callback interface	IFdtCommunicationEvents IFdtEvents

DTM is a software component which the device manufacturers develop according to the device function. Through it, the devices can be easily integrated with other manufacturers' equipment to the control system and in the control system engineers can use the equipment more easily. The DTM interface is divided into seven categories, as listed in Table 2.

1) Default interfaces

   The interface shall be implemented by each DTM because it contains the basic information and the functions of a DTM.

2) Access to device parameter

   This interface provides the access to the device parameters. The DTM reads the parameters by using the DtmParameter interface. The returned information is a representation of an instance data in memory. It depends on the type of device and available parameters in the types of the fieldbus. The interface makes the application framework to access the device online.

DTM provides an instance of data sets and the actual memory representation. Whether it could get the parameters or not can be determined by the DTM, the device type and bus type.

FDT framework can obtain the process channel by IDtmChannel interfaces.

3) Device documentation

It provides the information about the device instance. DTM provides information related to the device instance via IDtmDocumentation. The returned data will be converted into HTML format that can contain pictures and hyperlinks and can be printed.

4) Load/save data

It stores the instances which are related to the parameters by using the IPersistStreamInit interface or default IPersistPropertyBag interface. It doesn't define how the data is stored or what kind of data it stores but the DTM shall be able to rebuild their complete state when the FDT framework application sends such a request.

5) Start of the user interfaces

Graphical user interface is realized through the ActiveX technologies. ActiveX is an extension of COM technology. It defines how to integrate of the graphical user interface to applications. In the FDT, ActiveX control is displayed by the FDT framework application and connected to the DTM to exchange the data. Therefore, it can be seamlessly integrated into the user interface of the FDT frame applications and it can display the full functions of DTM.

6) Diagnosis functions

Such interface provides the diagnostic functions for the configuration of parameters. The basic diagnostic function which the FDT frame application uses to configure parameters is defined by the IDtmDiagnosis interface. Through the interface, the FDT frame application can find out whether the two DTM data sets are the same. On the contrary, IDtmOnlineDiagnosis defines a method which compares DTM data sets through the device parameter and device status can be determined by the interface.

7) Callback interface

It provides communication-related functions. The DTM defines IFdtCommunicationEvents and IFdtEvents as callback interface. Through IFdtCommunicationEvents interface, the DTM can get the response of the communication requests that are issued. Sometimes it is necessary for DTM to know the changed information of the environment. IFdtEvents can notify this information to a DTM, such that in multi-user environments the other DTM locks its data sets or a DTM changes the parameters which are provided by another DTM.

EDD elements can be divided into two categories: the description of basic parameters and its business logic. Description of the basic parameters includes the description of the device parameters and definition of device functions. Business logic includes equipment variables and the mapping of functional description and communication system while includes display and operation of the user interface. In order to achieve the conversion from EDD to FDT, EDD elements shall be mapped to the DTM. Therefore, these two categories can separately be mapped to the DTM.

5.2.2 Standardized requirements of the basic parameters description

The EDD contains the device description of the parameters. According to these parameters, the agent interpreter parses and the accordant results shall be given after it. The basic parameters are mapped for the FDT/DTM functionality and corresponding data.

The EDD contains the identification information, including the definition of manufacturer, device type and general information including version information of the equipment, such as: MANUFACTURER, DEVICE_TYPE, DEVICE_REVISION and EDD_REVISION. It defines the basic structural elements to describe the device parameters such as VARIABLE, RECORD, VARIABLE_LIST, BLOCK_A, METHOD and the effective organization of these data elements to describe the device attributes and define the function of the devices.

In the DTM, the default interface, that is IDtm and IdtmInformation interfaces, contains the basic features and interactive functions of equipment data in DTM. DTM general information, such as vendor, version number, and performance of DTM, can be accessed through IDtmInformation. Therefore, the description of the basic parameters of EDDL can be mapped to the corresponding interface in DTM, specifically as shown in Figure 3.

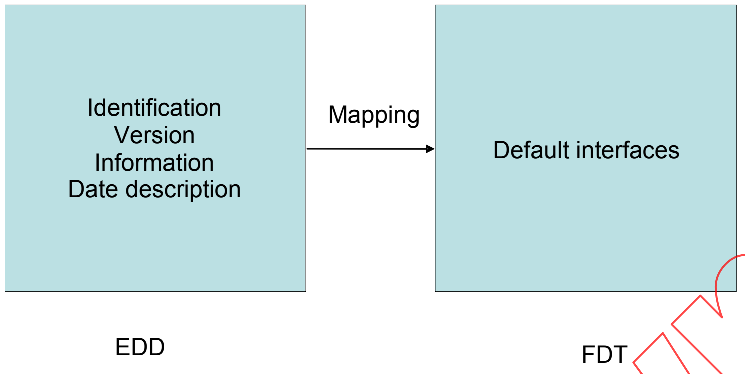

Figure 3 – Basic parameter description mapping

5.2.3 Standardized requirements of the fieldbus communication protocol mapping

The mapping of the fieldbus communication protocol includes the access method of the device parameters, the physical address, and so on. EDD realizes the device variables and the mapping between the description of functions and the communication system by defining the basic structural elements such as COMMAND, RESPONSE_CODE. In DTM, except for the default interface Device documentation, Load/save data, Diagnosis functions and Callback interface can be classified as communication protocol callback interface mapping interfaces. Therefore, the communication described modules of EDD can be mapped to these interfaces of DTM, see Figure 4.

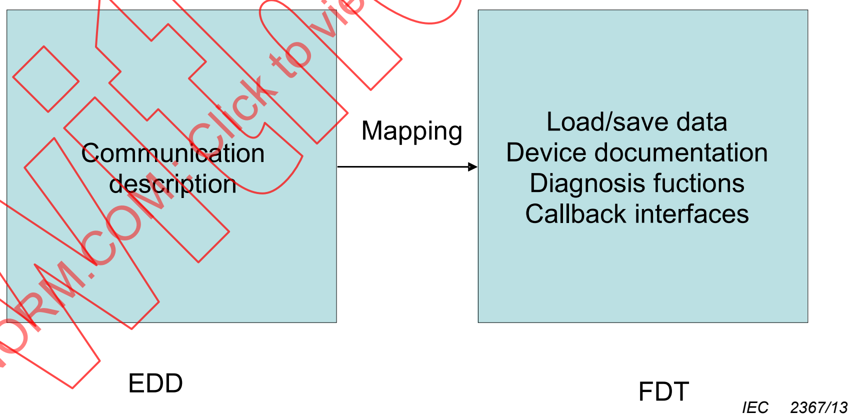

Figure 4 – Communication mapping

5.2.4 Standardized requirements of the device parameters visual description

The visual depiction of device parameters mainly includes a logical group of parameters (menu display) and the graphical display. EDD specifies data editing, display format and the description of the graphical interface by defining user interface elements such as EDIT_DISPLAY, MENU and provides a coherent description for the display of the application's data, editing format and visualization as shown in Figure 5.

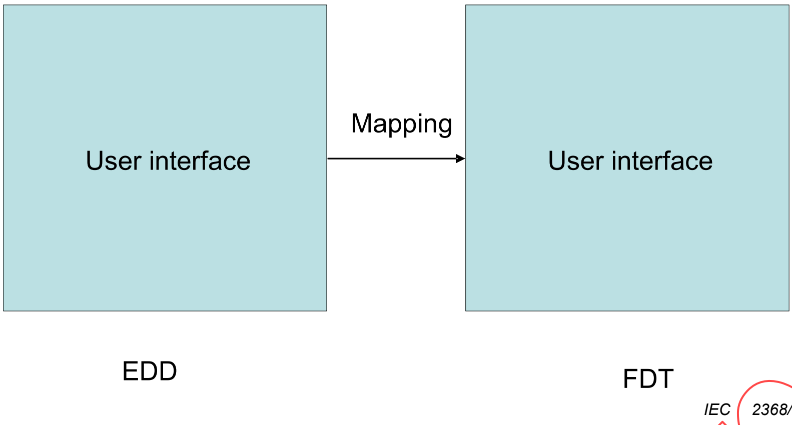

Figure 5 – User interface mapping

5.2.5 Standardized requirements of error handling

The agent interpreter may report errors in EDD when it works. When the errors occur, the agent interpreter should give specification and detailed error messages in order to track the source of the error or to take appropriate corrective measures. According to the different types of errors, there is a need to provide the corresponding error code, the details of the errors, and the presentation means of the error information.

1) The principles of error handling

   Error handling codes should have the following characteristics:

   a) User-friendly

      Error handling is for the user of the application. Therefore the error handling should be for customer service and to help users understand the problems in order to achieve restoration.

   b) Error handling is not interrupted (unless required to do so)

      Error handling should ensure that users could work as normal even if there are problems. It should be able to obtain the data that may generate an error and could process it and sometimes ignore certain low-level errors or maintain and continue processing on the condition that it guarantees the program does not crash.

   c) Comprehensive information

      When an error occurs, it shall provide useful comprehensive information. This information should be the information that the user first needs. If the error cannot be recovered well and the process shall be changed, then the error message shall indicate what to do next.

2) Three error types and their treatments

   a) Ignore the warnings

      The warning is defined as no errors or non-fatal errors. The warning should show if applicable that the warning signs will not prevent the interpreter from continuing to parse the area where the problem was detected. This could be done by displaying a pop up notification.

   b) Error

      For example, the document may be incomplete or the data from the parsing of the document may be lost, altered or wrong. The program should be able to recover from such errors rather than stop parsing and processing. The error messages should be recorded. This allows for better tracking of incidents and prevents the problem from happening again.

   c) Fatal error

The fatal error would interfere with the resolution process and prevent it from continuing. An example of a fatal error would be a document that lacks a good format. The application shall assume that the document is not available as long as it reports a fatal error and returns an error message. It should provide application-specific information for debugging, such as status codes and error pages.

5.3 The requirements of EDD document format

Currently, some EDD files have been compiled into a binary format. As this is not standardized, this document requires an EDD source file according to IEC 61804.

The EDD is written in plain text format. Text format is the most common supported storage format of the file and almost any kind of text editor supports it. So the use of text format greatly facilitates the writing of EDDL. However, due to limited processing power and storage space, some of the microprocessors cannot directly deal with the text format of EDDL. So a compiler first needs to compile an EDDL text file into a target binary file and then embed an agent interpreter in the configuration software or the device management software. In this way, the equipment providers only need to provide the EDDL binary file, because the binary file is not easily modified and will greatly enhance the safety of the equipment.

An EDDL compiler will convert the text file into a binary format in order to directly deal with it through the processor and the problem is considered at the beginning of defining the EDDL. Therefore, defining the different elements maintains a good structure. EDDL can be converted into element tables (array of structures), such as a variable table, order table, method table, menu table and a variety of relational tables, and a number of auxiliary elements tables, etc., while simultaneously addressing well-designed rules. These forms can be packaged into a DD package. For the convenience of addressing, the maximum space of a package should be fixed (e.g. 64 kB) so that large amounts of data described in a text file EDDL can be packaged into the device description library.

5.4 Requirements for EDD storage management

There are different methods for converting EDD to DTM and each may have different management requirements. For example, if a DTM needs to load the EDD dynamically at run time, it should provide the EDD's document management capabilities. An EDD should be saved in the specified directory and save the file information, such as registration table (registry), profile and so on in a fixed position. After doing this, the DTM can find the EDD and load it at run time. In order to ensure consistent results every time, the contents of the EDD should not be modified by the DTM. After the success of the loading, the DTM shall release the reference of the EDD as soon as possible so that other entities can refer to the EDD file.

For example, it is possible to manage EDD by using a Windows®[2] registry, where each file corresponds to a registry key that contains the file name, device ID, version and other information or by generating additional XML files to manage the EDD according to the specifications of the DTM interface. The XML files shall be able to display the EDD correctly. The XML elements and attributes are generated to record the EDD information in this XML file. Although using the different means, the goal of the two different implementations is the same, i.e. the program should quickly access critical information devices in the run time without reading and parsing all the EDD files. The FDT container manages the device information via the device list in FDT technology, but sometimes the DTM needs all the EDD files as, for instance, in selecting a device that matches the sub-module. Clause 5 provides the corresponding requests of the EDD file storage management.

[2] Windows® is an example of a suitable product available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of this product.

5.5 The user interface style requirements of FDT/DTM converted from EDDL

5.5.1 Overview

The interface style requirements standardize the FDT/DTM interface changed from EDDL. The purpose of 5.5 is to provide a relatively uniform and friendly interface to the users. Based on 5.5, users may operate different devices from different vendors more efficiently, more safely and uniformly. This user interface is different from ordinary DTM user interface based on the EDD document management area. IEC/TR 62453-61 describes ordinary DTM user interface in detail.

5.5.2 Basic requirements of the FDT/DTM interface design

Principles for designing DTM user interfaces according to the FDT/DTM user's point of view:

1) Visibility of system status

   The system should always keep users informed about what is going on, through appropriate feedback within an acceptable time limit.

2) Match between system and user's logic

   The system should speak the users' language with words, phrases and concepts familiar to the user, rather than system-oriented terms.

3) Uniform standards

   The operations of different devices are completely unified. Users should not have to wonder whether different words, situations, or actions mean the same thing.

4) Flexibility and new user supports

   Enable users to tailor frequent actions and provide novice navigation to satisfy new users.

5) Error prevention and diagnosis

   A careful design which prevents problems from occurring in the first place is even better than good error messages. Error messages should be expressed in plain language, precisely indicating the problem, and constructively suggesting a solution.

6) Help and documentation

   Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focus on the user's task, list concrete steps to be carried out, and not be too large.

5.5.3 FDT/DTM user interface

The user interface of a DTM should be based on the Microsoft® Windows® Style Guide[3]. It is recommended to use Windows® common controls. Windows® common controls shall act in the way defined by Microsoft®. This means that changing the behavior of common controls such as buttons, combo boxes, edit controls, keyboard shortcuts, etc. is not allowed.

1) FDT/DTM user interface categories

   In general, a DTM user interface is divided into the following areas:

   - Identification Area: contains information about the device that is handled by the DTM.
   - EDD Management Area: manages the EDD and shows some information about the EDD.
   - Application Area: contains all necessary GUI elements for the selected function.
   - Navigation Area: contains a navigation tree consisting of all the parameters.
   - Action Area: contains buttons to initiate the user's choice.

3 Microsoft® Windows® Style Guide is an example of a suitable product available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of this product.

- Status Bar: contains global status information about the DTM and device.

2) Standard user interface of FDT/DTM

In the standard user interface of FDT/DTM, Identification Area, EDD Management Area, Navigation Area, Application Area, Action Area, Status Bar are the mandatory elements and Menu and Toolbar are the optional elements. The FDT/DTM standard user interface is shown in Table 3, and specific examples are shown in Appendix B.

Table 3 – Standard user interface of FDT/DTM

<table>
<tr><td>Identification Area</td><td>EDD Management Area</td></tr>
<tr><td colspan="2">Menu (optional)</td></tr>
<tr><td colspan="2">Toolbar (optional)</td></tr>
<tr><td>Navigation Area</td><td>Application Area</td></tr>
<tr><td colspan="2">Action Area</td></tr>
<tr><td colspan="2">Status Bar</td></tr>
</table>

5.5.4 FDT/DTM user interface areas

In general, the user shall be aware if modifications of parameters are applied into the data set of the DTM or into the device. Also the behavior of the GUI may vary depending on whether the GUI supports block mode or direct mode.

1) Identification Area

This area contains information about the device that is handled by the DTM (see Table 4).

Table 4 – Contents of Identification Area

Height	Width	Contents	Availability
Maximal 3 text lines	Variable	Device picture (left side) recommended	Recommended
		Company logo of the device FDT/DTM (right side)	Recommended
		Between picture and logo:	
		1. Device name, version information of DTM, version information of FDT that supports the DTM	Mandatory
		2. Description	Recommended
		3. DTM specific line	Optional

2) EDD Management Area

Click on the icon button for loading the EDD, then choose the path to load the EDD. Click on the icon button showing the EDD information; a window will pop up that includes some information defined in the EDD, such as the device's type number, device's type name, detailed description of the hardware and software versions, and so on. Without the EDD chosen, the DTM cannot run, it just has a GUI.

3) Menu

This optional area shall be used only in special cases. The Menu shall contain only menu entries which are directly related to the Application Area. That means that if application A is opened within the Application Area the Menu shall not contain menu entries of application B.

4) Toolbar

This optional area shall be used only in special cases. The Toolbar shall contain only elements which are directly related to the Application Area. Also a [Help] button could be part of the toolbar.

5) Navigation Area

The Navigation Area contains a navigation tree. This tree provides an overview of the whole parameter set. Within the tree the parameter set shall be grouped to appropriate parameter groups. When selecting a navigation tree entry (leaf or branch) the corresponding parameter group shall be displayed in the Application Area.

6) Application Area

The content of this area depends on the selected application.

7) Action Area

The content of this area contains two situations:

a) Application without changeable values

The Action Area contains only a [Close] button. Activation of this button closes the DTM user interface.

b) Application with changeable values

The area contains the [Ok] [Cancel] [Apply] buttons.

The button texts shall be adapted to the configured language according to the texts used in standard Microsoft® Windows® application. The current data source is currently shown in the status bar. The specific application is shown in Table 5.

Table 5 – Contents of Action Area[4]

Button	Data source	Behavior
【OK】		Values changed in the user interface will be applied on the instance data set. The user interface will be closed.
【OK】		All changed values will be transmitted to the device. The user interface will be closed.
【OK】	 	Applications contain values from different data sources. Each value of the application itself shall be uniquely assigned to one data source by an appropriate icon. The user interface will be closed.
【Cancel】	All states	In general, changed values will NOT be applied. Application will be closed.
【Apply】		Same behavior as [OK], but application will NOT be closed.
【Apply】		Same behavior as [OK], but application will NOT be closed.
【Apply】	 	Same behavior as [OK], but application will NOT be closed.

8) Status Bar

The Status Bar contains global status information about the DTM and the device. The elements of the Status Bar are listed in Table 6.

[4] The icons of Tables 5 to Table 10 are taken from IEC/TR 62453-61.

Table 6 – Elements of the Status Bar

Height	Width	Contents	Icon	Availability
1 text line	Variable	DTM connection state	See Table 7- Connection states of DTM	Mandatory
		Communication in progress		Mandatory
		Data source	See Table 8- Data source states	Mandatory
		State of instance data set	See Table 9 - States of the instance data set	Mandatory
		Changes directly made on the device. Changes have only an impact on the device and not on the instance data set		Mandatory
		Direct mode active		Mandatory
		Device diagnosis status	See Table 10 - Device diagnostic states	Optional
		OEM login or user role	'No user logged in' or name of the user who is logged in	Optional
		Progress bar		Optional

In all of the Status Bar elements, the corresponding states of DTM are shown in Table 7. Different connection states of DTM are visible, according to the DTM's correlative status icons in the Status Bar.

Table 7 – Connection states of DTM

Icon	Meaning	DTM state	Availability
	Connecting	Going online	Optional
	Connected	Online	Mandatory
	Disconnecting	Going offline	Optional
	Disconnected	Offline	Mandatory
	Disconnected-disturbed	Communication set after an error condition	Mandatory
Empty (no icon)	Unknown	Going online or going offline	Mandatory

In all of the Status Bar elements, the corresponding data source states are shown in Table 8. Different data source states and behaviors are easy to obtain, according to the correlative icons of data source states in the Status Bar.