IEC TR 62453-52-31

Edition 1.0 2017-06

TECHNICAL REPORT

colour inside

Field device tool (FDT) interface specification – Part 52-31: Communication implementation for common language infrastructure – IEC 61784 CP 3/1 and CP 3/2

IEC TR 62453-52-31:2017-06(en)

IEC TR 62453-52-31

Edition 1.0 2017-06

TECHNICAL REPORT

colour inside

Field device tool (FDT) interface specification – Part 52-31: Communication implementation for common language infrastructure – IEC 61784 CP 3/1 and CP 3/2

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 25.040.40; 35.100.05; 35.110

ISBN 978-2-8322-4335-0

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

____

FIELD DEVICE TOOL (FDT) INTERFACE SPECIFICATION –

Part 52-31: Communication implementation for common language infrastructure – IEC 61784 CP 3/1 and CP 3/2

FOREWORD

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 62453-52-31, which is a technical report, has been prepared by subcommittee 65E: Devices and integration in enterprise systems, of IEC technicall committee 65: Industrial-process measurement, control and automation.

Each part of the IEC 62453-52-xy series is intended to be read in conjunction with its corresponding part in the IEC 62453-3xy series. The corresponding part for this document is IEC 62453-303-1.

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

Enquiry draft	Report on voting
65E/440/DTR	65E/514/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 document has been drafted in accordance with the ISO/IEC Directives, Part 2.

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

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

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- withdrawn,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

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 is an interface specification for developers of Field Device Tool (FDT) 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 Device Type Manager (DTM), 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 this part of the IEC 62453-52-xy series is aligned in the structure of the IEC 62453 series.

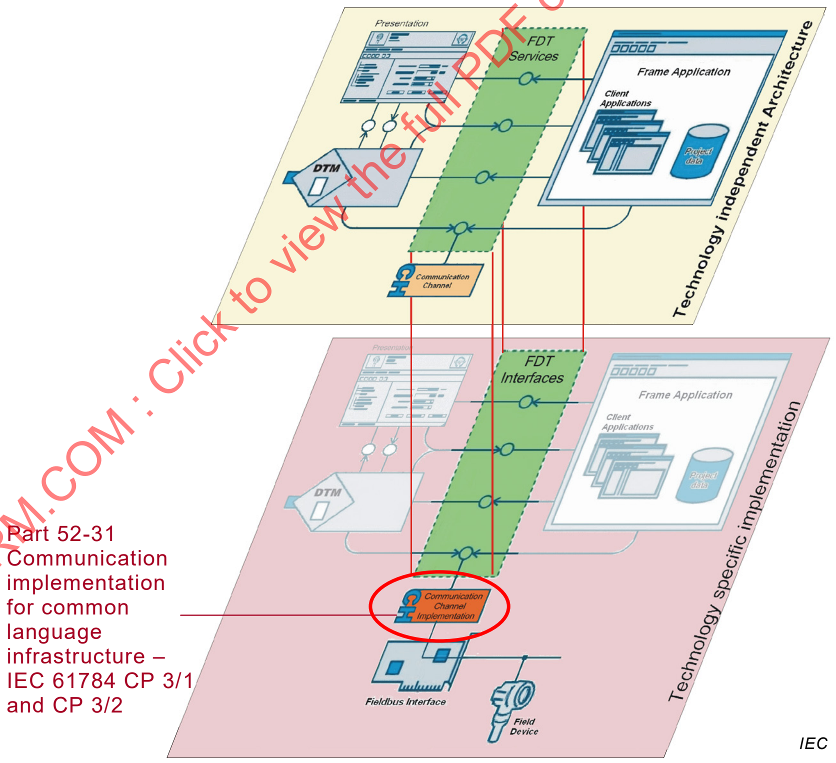

Presentation
FDT Services
Frame Application
Client Applications
Project data
DTM
Communication Channel
Technology independent Architecture
Presentation
FDT Interfaces
Frame Application
Client Applications
Project data
DTM
Part 52-31 Communication implementation for common language infrastructure – IEC 61784 CP 3/1 and CP 3/2
Communication Channel Implementation
Fieldbus Interface
Field Device
Technology specific implementation
IEC

Figure 1 – Part 52-31 of the IEC 62453 series

FIELD DEVICE TOOL (FDT) INTERFACE SPECIFICATION –

Part 52-31: Communication implementation for common language infrastructure – IEC 61784 CP 3/1 and CP 3/2

1 Scope

This part of the IEC 62453-52-xy series, which is a Technical Report, provides information for integrating the PROFIBUS[1] technology into the CLI-based implementation of FDT interface specification (IEC TR 62453-42).

This part of IEC 62453 specifies implementation of communication and other services based on IEC 62453-303-1.

This document neither contains the FDT specification nor modifies it.

2 Normative references

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

IEC 61131-3:2003, *Programmable controllers – Part 3: Programming languages*

IEC 61158 (all parts), *Industrial communication networks – Fieldbus specifications*

IEC 61158-6-3:2014, *Industrial communication networks – Fieldbus specifications – Part 6-3: Application layer protocol specification – Type 3 elements*

IEC 61784-1:2014, *Industrial communication networks – Profiles – Part 1: Fieldbus profiles*

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

IEC 62453-2:2016, *Field device tool (FDT) interface specification – Part 2: Concepts and detailed description*

IEC TR 62453-42:2016, *Field device tool (FDT) interface specification – Part 42: Object model integration profile – Common language infrastructure*

IEC 62453-303-1:2009, *Field device tool (FDT) interface specification – Part 303-1: Communication profile integration – IEC 61784 CP 3/1 and CP 3/2*
IEC 62453-303-1:2009/AMD1:2016

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

3 Terms, definitions, symbols, abbreviated terms and conventions

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 62453-1, IEC 62453-2, IEC TR 62453-42 and IEC 62453-303-1 apply.

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

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

3.2 Symbols and abbreviated terms

For the purposes of this document, the symbols and abbreviations given in IEC 62453-1, IEC 62453-2, IEC 62453-303-1, and IEC TR 62453-42 apply.

3.3 Conventions

3.3.1 Datatype names and references to datatypes

The conventions for naming and referencing of datatypes are explained in IEC TR 62453-2:2016, Clause A.1.

3.3.2 Vocabulary for requirements

The following expressions are used when specifying requirements.

Usage of “shall” or “mandatory”	No exceptions allowed.
Usage of “should” or “recommended”	Strong recommendation. It may make sense in special exceptional cases to differ from the described behaviour.
Usage of “conditional”	Function or behaviour shall be provided, depending on defined conditions.
Usage of “can” or “optional”	Function or behaviour may be provided, depending on defined conditions.

3.3.3 Use of UML

The figures in this document are using UML notation as defined in Annex A of IEC 62453-1:2016.

4 Bus category

IEC 61784 CP 3/1 and IEC 61784 CP 3/2 protocols are identified in the attribute ProtocolId of the BusCategory element by the identifiers, as specified in IEC 62453-303-1.

The supported PhysicalLayer are identified by the Identifier values as specified in IEC 62453-303-1.

The supported DataLinkLayer are identified by the Identifier values as specified in IEC 62453-303-1.

5 Access to instance and device data

5.1 General

The minimum set of data provided by a Device DTM shall be:

– All device parameters of the Physical Block and Out value of the Function Blocks shall be exposed via the data interfaces (PROFIBUS PA devices).

– All process values available for the device shall be modelled as ProcessData including the ranges and scaling, if applicable.

– All network configuration related parameters shall be exposed in NetworkData (see Clause 8).

5.2 IO signals provided by DTM

A DTM shall provide IO signal information for the device using the IProcessData interface. The IO signals describe datatype and address parameters of process data as detailed in Clause 10.

5.3 Data interfaces

5.3.1 General

Via the interfaces IDeviceData and IInstanceData all device specific parameters shall be exposed.

5.3.2 Mapping of PROFIBUS datatypes to FDT datatypes

PROFIBUS uses datatypes as specified in IEC 61158 for the transmission on the fieldbus. The FDT interfaces IDeviceData and IInstanceData use .NET datatypes, while PLC applications use datatypes defined in IEC 61131-3. Hence a mapping between these three type systems is defined in Table 1.

Table 1 – Mapping of datatypes

PROFIBUS datatype	FDT datatype	IEC 61131 datatype
Bit information		
Boolean	BooleanValue	BOOL
Unsigned8	BinaryBitArrayValue[8]	BYTE
Unsigned16	BinaryBitArrayValue[16]	WORD
Unsigned32	BinaryBitArrayValue[32]	DWORD
Numeric information with and without sign		
Integer8	SignedByteValue	SINT
Integer16	IntValue	INT
Integer32	LongValue	DINT
Unsigned8	ByteValue	USINT
Unsigned16	UIntValue	UINT
Unsigned32	ULongValue	UDINT
Float32	FloatValue	REAL
Float64	DoubleValue	LREAL
Printable characters (e.g. text)		
Visible String	StringValue	STRING
Unicode String	StringValue	WSTRING

PROFIBUS datatype	FDT datatype	IEC 61131 datatype
Time information		
TimeDifference without Date Indication	TimeSpanValue	TIME
Date	DateValue	DATE
Time Of Day without date indication	TimeValue	TIME_OF_DAY
Time of Day with date indication	DateTimeValue	DATE_AND_TIME
Combinations of basic datatypes		
Octet String	BinaryByteArrayValue	ARRAY
ARRAY	StructDataGroup	ARRAY
STRUCT OF	StructDataGroup	STRUCT

The FDT datatypes are used by the <Read> and <Write> methods in the interfaces IInstanceData and IDeviceData.

5.3.3 SemanticInfo

The identifier in SemanticId shall be unique and always reference the same element. This means the semantic information shall be the same whenever the same data is referenced. By using this attribute e.g. a Frame Application is able to get the information regarding the meaning and usage of a single data structure.

Table 2 – Usage of general datatypes

Attribute	Description for use
SemanticInfo.ReadParameterAddress SemanticInfo.WriteParameterAddress	For PROFIBUS, ReadParameterAddress and WriteParameterAddress are always identical. The address string shall be constructed according to the rules of the FDT SemanticId. PROFIBUS Parameter Address: The property 'Address' follows the different device models that are defined for PROFIBUS devices. FDT currently supports the following models: – PROFIBUS DP / DP-V1, – PROFIBUS PA, – PROFIdrive (greater or equal profile version 3) **PROFIBUS DP / DP-V1** The device model is based on devices that are composed of slots, whereas slots do not have to represent physical objects. The data that is contained in the slots are addressable via Indexes. This data may be variables or composed blocks of data. The Address property is APIxxSLOTyyINDEXzz xx – API yy – Slot zz – Index xx, yy, zz are based on decimal format without leading '0'

Attribute	Description for use
	PROFIBUS PA The device is represented by a device management structure and a number of blocks that provide different functionality (physical block, function block, transducer block). The blocks are mapped to slot addresses, but this mapping may vary depending on the device type. The Address property is APIxxSLOTyyINDEXzz xx – API yy – Slot zz – Index xx, yy, zz are based on decimal format without leading '0' **PROFIdrive** According to the PROFIdrive profile [5], a device (drive unit) may be composed by a number (1..many) of drive objects (DOs). The DOs may have different type. Each DO is uniquely identifiable and manages its own parameters. Each parameter can be uniquely identified by its number (PNU). Each DO has its own number space. A parameter may contain simple data or composed data (e.g. arrays). The data of the device are accessible via a parameter channel (normaly slot 0 index 47). The Address property is APIxxSLOTyyINDEXzz.DOdo-id.pnu xx – API yy – Slot zz – Index do-id – Drive Object ID pnu – ParameterNumber xx, yy, zz, do-id, pnu are based on decimal format without leading '0'
SemanticInfo.ApplicationDomain/ SemanticInfo.SemanticId	The SemanticIDs for PROFIBUS follow the different device models that are defined for PROFIBUS devices. FDT currently supports the following models: – PROFIBUS DP, – PROFIBUS PA, – PROFIdrive. **PROFIBUS DP / DP-V1** The ApplicationDomain is: FDT_PROFIBUS_DPV1 The device model is based on devices that are composed of slots, whereas slots do not have to represent physical objects. The data that is contained in the slots are addressable via Indexes. This data may be variables or composed blocks of data. The SemanticId for devices not based on a profile is directly based on the PROFIBUS address information: The SemanticId is: APIxx.SLOTyy.INDEXzz xx – AP yy – Slot zz – Index xx, yy, zz are based on decimal format without leading '0'

Attribute	Description for use
	PROFIBUS PA The ApplicationDomain is: FDT_PROFIBUS_PA The device is represented by a device management structure and a number of blocks that provide different functionality (physical block, function block, transducer block). The blocks are mapped to slot addresses, but this mapping may vary depending on the device type. Since the device model is based on blocks, the SemanticIds also are based on the block model. Within each block, the data is identifiable by names of parameters. The SemanticId for PROFIBUS profile related parameter follows the following rules: – the SemanticId shall be built based on the names defined in the profiles; – structured parameters shall be combined with a '.'; – spaces within the profile definition shall be exchanged with an underscore; – blocks shall be counted according to the Object Dictionary; – the block number shall be part of the SemanticId. The SemanticId is BlockType.BlockIndex.NameOfParameter.AttributeOfParameter EXAMPLE AnalogInputFB.3.OUT.Unit
(cont. SemanticInfo.ApplicationDomain/ SemanticInfo.SemanticId)	**PROFIdrive** The ApplicationDomain is: FDT_PROFIBUS_PROFIDRIVE According to the PROFIdrive profile, a device (drive unit) may be composed by a number (1..many) of drive objects (DOs). The DOs may have different types. Each DO is uniquely identifiable and manages its own parameters. Each parameter can be uniquely identified by its number (PNU). Each DO has its own number space. A parameter may contain simple data or composed data (e.g. arrays). The data of the device are accessible via a parameter channel (slot 0, index 47). The SemanticId is: DOdo-id.PNUpnu do-id – Drive Object ID pnu – ParameterNumber do-id, pnu are based on decimal format without leading '0' EXAMPLE DO3.PNU64

6 Protocol specific behaviour

6.1 PROFIBUS device model

The definition of Process Data Items for the description of I/O values shall be structured according to the PROFIBUS device model (see Figure 2).

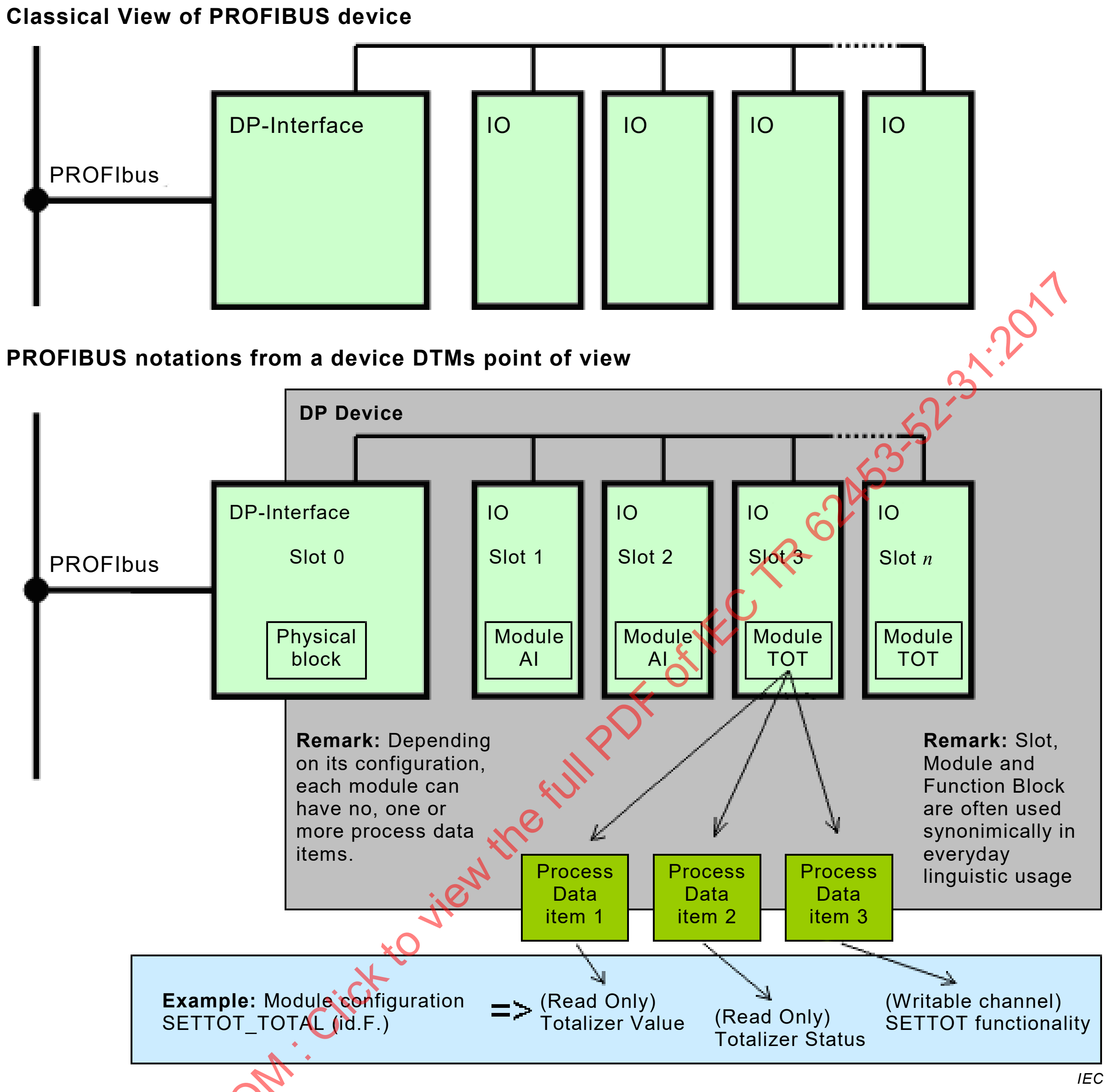

Classical View of PROFIBUS device
PROFIbus
DP-Interface
IO
IO
IO
IO
PROFIBUS notations from a device DTMs point of view
DP Device
PROFIbus
DP-Interface
Slot 0
Physical block
IO
Slot 1
Module AI
IO
Slot 2
Module AI
IO
Slot 3
Module TOT
IO
Slot n
Module TOT
Remark: Depending on its configuration, each module can have no, one or more process data items.
Remark: Slot, Module and Function Block are often used synonimically in everyday linguistic usage
Process Data item 1
Process Data item 2
Process Data item 3
Example: Module configuration SETTOT_TOTAL (id.F.)
=>
(Read Only) Totalizer Value
(Read Only) Totalizer Status
(Writable channel) SETTOT functionality
IEC

Figure 2 – FDT PROFIBUS Device Model

DTMs for PROFIBUS devices shall provide information about their I/O data to provide engineering systems knowledge to access such data without the use of the DTMs.

6.2 Configuration and parameterization of PROFIBUS devices

6.2.1 General

In a GSD-based configuration tool, the user defines the configuration and sets the appropriate parameters for the modules. The configuration tool creates the configuration string and the parameter string that are used to set up the slave properly.

With FDT the configuration and parameterization of the devices is no longer executed only by a central piece of software; it moved partly into the DTMs. A Device DTM is responsible for providing configuration and parameterization information for a PROFIBUS master that puts the PROFIBUS slaves in operation.

A Device DTM is used to adjust a field device to its specific application. Within PROFIBUS, there are three different aspects of adjustment:

– Communication parameterization: User Prm Data (used in the PROFIBUS service Set_Prm for setting up the cyclic communication and the specific behaviour of the device);
– Configuration data: Cfg Data (used in the PROFIBUS service Chk_Cfg for definition of the format and length of the input/output data that are transmitted within cyclic communication);
– Application parameterization: application specific parameters (transmitted via acyclic read/write PROFIBUS services).

The application parameterization transmitted via acyclic communication is not in the scope of this document. The parameter data transmitted for this purpose is device specific. Only the communication services that can be used by Device DTMs for performing such device specific parameterization are defined. Within this document the term parameterization represents communication parameterization (Set_Prm).

6.2.2 Monolithic DTM for a modular PROFIBUS device

A monolithic DTM is one single DTM that represents the complete device with its Bus Interface Module (BIM) and its I/O modules. In general, such a DTM offers a configuration user interface (presentation object) that allows definition of the used BIM and module types.

Not all PROFIBUS devices require a configuration user interface. That is why not all DTMs provide the configuration function (ApplicationID: Configuration). This is valid only for non-modular PROFIBUS devices if the User Prm Data cannot be changed.

The configuration dialog shall allow changing the data only in offline mode if the data set can be locked.

6.2.3 Composite DTM for a modular PROFIBUS device

Separate DTMs represent the BIM (Composite Device DTM) and the particular I/O modules (Module DTMs). The effort developing such a modular DTM is normally higher than in the case of a monolithic DTM, because:

– a private protocol shall be implemented between BIM and I/O modules to ensure that only a Module DTM can be added to the BIM DTM. This requires an own FDT protocol ID and the adaptation/creation of FDT communication datatypes.

Implementing a Modular DTM results in the following advantages:

– the project topology represents the device structure,
– the user is able to access module-related information directly as a function of the Module DTM,
– IEC 62453 defines a mechanism to identify DTMs. With these mechanisms it is possible to provide support for scanning the modules below the BIM and generate the topology automatically,
– supporting a new type of BIM or I/O module requires an additional DTM "only" and does not affect existing components. This may result in reduced test effort that can also simplify the certification process.

The configuration data to set up the PROFIBUS configuration of a modular PROFIBUS device shall be provided by the Device DTM representing the BIM. This configuration data may be generated from information of the instantiated Module DTMs and by using a configuration dialog.

Modular DTMs can be provided for modular devices (e.g. a plant operator may add/remove modules). Monolithic DTMs can be used to represent devices that show no modularity (e.g. PA devices).

6.3 Support for DP-V0 configuration

A PROFIBUS slave usually communicates cyclically via PROFIBUS DP-V0 with a class 1 master (DPM1). In addition to this the slave may support DP-V1 communication. This should be indicated by setting the SlaveFlagDpv1Slave property (see 6.5.2.2.2 Slave bus parameter set) to true.

A Gateway DTM for a PROFIBUS slave does not have to provide DP-V0 communication. An example is a remote I/O system with HART[2] modules. It may have a Gateway DTM that requires the DP-V1 protocol and provides the HART protocol. This enables HART Device DTMs to communicate with their devices via the Gateway DTM and via a Communication DTM for DP-V1. Following the specification the Gateway DTM delivers process data items for both protocols DP-V1 and HART. The ProtocolId is a member of NetworkDataInfo datatype.

The Process Data Info of a Device DTM shall contain data items for DP-V0 including all information to allow integration into the control system (e.g. Dpv0IOSignalInfo of the I/O value if available).

6.4 PROFIBUS slaves operating without a class 1 PROFIBUS master

In most cases, a PROFIBUS slave is configured and parameterized by a PROFIBUS class 1 master device. So a running master device in the network is required.

Some slaves (marked via SlaveFlagDpv1Slave) are able to allow acyclic communication without cyclic master to slave communication. Especially in the case of gateway functionality this allows the parameterization of field devices connected to them by using a class 2 master. So instrument specialists are able to work with field devices also in case the controller is not yet working.

If a master starts communication, these devices start to detect bus speed to react properly. This may take some time. A communication DTM or gateway DTM shall take this into account and adjust internal timeouts accordingly.

In the following, two examples for problems regarding detecting devices are described that a user may keep in mind when working with such devices.

EXAMPLE 1:
The user performs a network scan. The Communication DTM tries to read diagnostic data via a Dpv0ReadDiagnosis Request but does not receive a response. The device is not detected by the Communication DTM. This occurs mostly when the device has a low PROFIBUS address. The reason is that the device has not completed bus speed detection when it was asked for its diagnostic data. The workaround is to assign these devices a higher PROFIBUS address.

EXAMPLE 2:
The user tries to connect a field device linked to a gateway that supports DP-V1 without a running cyclic master. This can lead to an error message because the gateway device has not completed bus speed detection when it is asked for a connection. Consequently, the user has to try to connect again. This happens only in very rare situations.

6.5 PROFIBUS-related information of a slave DTM

6.5.1 General

The information used by a cyclic master device to set up the PROFIBUS network properly and allow cyclic communication between control system and slave devices shall be provided by a DTM in

[2] HART ® is the trade name of a product supplied by HART Communication Foundation. This information is given for convenience of users of this document and does not constitute an endorsement by IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

– PROFIBUS Network Data,
– GSD information,
– Process data items.

A DTM of a PROFIBUS slave shall deliver these parts of PROFIBUS-related information to get integrated into a Frame Application. In the next subclauses, a more detailed description is given on how to generate and how to provide this information. The actual information depends on the kind of DTM (see 6.2 Configuration and parameterization of PROFIBUS devices).

6.5.2 PROFIBUS Network Data (PND)

6.5.2.1 PND introduction

The PND of a single DTM instance describes the actual parameter and configuration data of the corresponding PROFIBUS slave. Each DTM representing a PROFIBUS slave device shall provide PROFIBUS Network Data. The PND is the PROFIBUS specific NetworkDataInfo datatype. This information is obtained by calling the service GetNetworkDataInfo.

The PND includes information about the configuration and the parameters for initialization of the slave. The PND is provided by the DTM and is required in order to generate the bus master configuration.

The PND contains data which might be changed during master configuration. That means that the PND may be transferred back to the slave DTM by calling SetNetworkData. A Slave DTM shall accept the new information and recompute the configuration/internal parameters to match the new PND.

The Slave DTM shall check whether the new values are according to the capabilities of the device. The call to SetNetworkData shall be refused (by throwing an exception of type Fdt.FdtInvalidValueException) if the device cannot handle the new values.

6.5.2.2 Creating the PND

6.5.2.2.1 General

The PND may be generated from the GSD information of a PROFIBUS device. This subclause explains the meaning of the individual elements of the PND in detail. The explanations reference the PROFIBUS specification and use the GSD keywords.

6.5.2.2.2 Slave bus parameter set

All values are provided by the Slave DTM. It is the responsibility of the Slave DTM to be compatible with the Slave GSD. The Master DTM can overwrite some of these initial values sent by the Slave DTM if they depend on the capabilities of the master.

EXAMPLE:
Within the GSD file, it is stated that a device supports the Freeze Mode by the keyword “Freeze_Mode_supp”. The master sets the value “PrmDataFreezeMode” within the Slave Bus Parameter Set because only the master knows whether it supports this mode.

Table 3 explains which component is the source of the parameter values (“Information source ”). Some of the values can be changed by the system or by user interaction. If possible, the default values for the parameters are defined (“Default Value”).

Table 3 – PROFIBUS Network Information

Member Name	Type	Information source and meaning	Default Value
DeviceDescriptionReference	Document	Slave DTM Path reference to GSD file.	-
SlaveFlagExtraAlarmSap	Boolean	Slave DTM false – master shall acknowledge alarms via SAP51 true – master shall acknowledge alarms via SAP50	Extra_Alarm_SAP_supp (GSD)
SlaveFlagDpv1DataTypes	Boolean	Slave DTM false – DP slave CFG data of EN 50170 true – DP slave CFG data of DP-V1	DPV1_Data_Types (GSD)
SlaveFlagDpv1Slave	Boolean	Slave DTM false – slave does not support DP-V1 true – slave does support DP-V1	DPV1_Slave (GSD)
SlaveFlagPublisherSupport	Boolean	Slave DTM false – no publisher support true – DP slave supports publisher functionality	Publisher_supp (GSD)
SlaveFlagFailSafe	Boolean	Slave DTM false – DP slave receives zero data in CLEAR mode true – DP slave receives no data in CLEAR mode	Fail_Safe or Fail_Safe_required (GSD)
SlaveFlagNaToAbort	Boolean	Slave DTM false – no abort when NA occurs true – abort if NA (no response from FDL) occurs	0
SlaveFlagIgnoreAutoClear	Boolean	Slave DTM false – process the auto clear function true – ignore the auto clear function	0
MaxDiagDataLen	Byte	Slave DTM	Max_Diag_Data_Len (GSD)

Member Name	Type	Information source and meaning	Default Value
MaxAlarmLen	Byte	Length of the alarm structure, see IEC 61158-6-3:2014, Table 5 Slave DTM, Conditions: One of GSD keywords:- Diagnostic_Alarm_supp – Process_Alarm_supp – Pull_Plug_Alarm_supp – Status_Alarm_supp – Updata_Alarm_supp – Manufacturer_Specific_Alarm_supp OR – Alarm_Type_Mode_supp	Is calculated on the base of the different GSD values
MaxChannelDataLen	Byte	Slave DTM: This field defines how much data can be transferred between slave and master. In this case the maximum of these values shall be calculated and set by the slave DTM. Rule for calculation: Max_Data_Len or C1_Max_Data_Len plus 4 Bytes (Function Num, Slot_Number, Length)	Value within Slave GSD
DiagUpdateDelay	Byte	Slave DTM	Slave GSD [2] Diag_Update_Delay
AlarmMode	Byte	Slave DTM	Slave GSD [2] Alarm_Sequence_Mode_Count
C1ResponseTimeout	Word	Slave DTM	Slave GSD [2] C1_Response_Timeout
PrmDataWdOn	Boolean	Master defines that watchdog is used or not. If watchdog is enabled, the master also shall set WD_Fact_1 and WD_Fact_2	0
PrmDataFreezeMode	Boolean	Slave DTM shows with this bit that the feature is supported	Freeze_Mode_supp within GSD
PrmDataSyncMode	Boolean	Slave DTM shows with this bit that the feature is supported	Sync_Mode_supp within GSD
PrmDataLockReq	Boolean	Master DTM	0
PrmDataUnlockReq	Boolean	Master DTM	0

PrmDataWdFact1	Byte	Depending on PrmDataWdBase1ms
		
		Watchdog_Time = 10 ms * WD_Fact_1 * WD_Fact_2 OR
		Watchdog_Time = 1 ms * WD_Fact_1 * WD_Fact_2
			1

PrmDataWdFact2	Byte	Depending on PrmDataWdBase1ms
		
		Watchdog_Time = 10 ms * WD_Fact_1 * WD_Fact_2 OR
		Watchdog_Time = 1 ms * WD_Fact_1 * WD_Fact_2
			1

PrmDataMinTsdr	Byte	Slave DTM	11
PrmDataIdentNumber	Word	Slave DTM	Slave GSD [2] IDENT_NUMBER
PrmDataGroupIdent	Byte	Indicates to what groups the device belongs. It is set by the master.	0 (not assigned to any group)
PrmDataWdBase1ms	Boolean	See PrmDataWdFact1 and PrmDataWdFact2 false – watchdog time base is 10 ms true – watchdog time base is 1 ms	WD_Base_1ms_supp within GSD
PrmDataFailSafe	Boolean	Slave DTM shows with this bit that the feature is supported	Fail_safe within GSD

Member Name	Type	Information source and meaning	Default Value
PrmDataFailSafeRequired	Boolean	Slave DTM shows with this bit that the master is expected to support this feature true – master shall support fail safe mode false – fail safe mode is optional If set to true, then PrmDataFailSafe shall be set to true too.	Fail_Safe_required within GSD
PrmDataDpv1Enable	Boolean	Slave DTM	Slave GSD (If the GSD minimum include one of these setting: C1_Read_Write_supp = 1 or Diagnostic_Alarm_supp = 1 or Process_Alarm_supp = 1 or Pull_Plug_Alarm_supp = 1 or Status_Alarm_supp = 1 or Update_Alarm_supp = 1 or Manufacturer_Specific_Alarm_supp = 1 . The slave supports DP-V1 and then the PrmDataDpv1Enable should be true
PrmDataCheckCfgMode	Boolean	Slave DTM shows with this bit that the feature is supported	Check_Cfg_Mode within GSD
PrmDataUpdateAlarmRequired	Boolean	Slave DTM shows with this bit that the master is required to support this feature true – master shall support update alarm false – update alarm is optional If set to true, then PrmDataUpdateAlarm shall be set to true too.	Update_Alarm_required within GSD
PrmDataUpdateAlarm	Boolean	Slave DTM shows with this bit that the feature is supported	Update_Alarm_supp within GSD
PrmDataStatusAlarmRequired	Boolean	Slave DTM shows with this bit that the master is required to support this feature true – master shall support status alarm false – status alarm is optional If set to true, then PrmDataStatusAlarm shall be set to true too.	Status_Alarm_required within GSD
PrmDataStatusAlarm	Boolean	Slave DTM shows with this bit that the feature is supported	Status_Alarm_supp within GSD
PrmDataManufacturerSpecificAlarmRequired	Boolean	Slave DTM shows with this bit that the master is required to support this feature true – master shall support manufacturer specific alarm false – manufacturer specific alarm is optional If set to true, then PrmDataManufacturerSpecificAlarm shall be set to true too.	Manufacturer_Specific_Alarm_required within GSD

Member Name	Type	Information source and meaning	Default Value
PrmDataManufacturerSpecificAlarm	Boolean	Slave DTM shows with this bit that the feature is supported	Manufacturer_Specific_Alarm_supp within GSD
PrmDataDiagnosticAlarmRequired	Boolean	Slave DTM shows with this bit that the master is required to support this feature true – master shall support diagnostic alarm false – diagnostic alarm is optional If set to true, then PrmDataDiagnosticAlarm shall be set to true too.	Diagnostic_Alarm_required within GSD
PrmDataDiagnosticAlarm	Boolean	Slave DTM shows with this bit that the feature is supported	Diagnostic_Alarm_supp within GSD
PrmDataProcessAlarmRequired	Boolean	Slave DTM shows with this bit that the master is required to support this feature true – master shall support process alarm false – process alarm is optional If set to true, then PrmDataProcessAlarm shall be set to true too.	Process_Alarm_required within GSD
PrmDataProcessAlarm	Boolean	Slave DTM shows with this bit that the feature is supported	Process_Alarm_supp within GSD
PrmDataPullPlugAlarmRequired	Boolean	Slave DTM shows with this bit that the master is required to support this feature true – master shall support pull plug alarm false – pull plug alarm is optional If set to true, then PrmDataPullPlugAlarm shall be set to true too.	Pull_Plug_Alarm_required within Slave GSD
PrmDataPullPlugAlarm	Boolean	Slave DTM shows with this bit that the feature is supported	Pull_Plug_Alarm_supp within Slave GSD
PrmDataBlockStructure	Boolean	Slave DTM shows with this bit that the feature is supported	Prm_Block_Structure_supp within GSD
PrmDataBlockStructureRequired	Boolean	Slave DTM shows with this bit that the master is expected to support this feature true – master shall support block structure false – block structure is optional If set to true, then PrmDataBlockStructure shall be set to true too.	Prm_Block_Structure_req within GSD
PrmDataIsochronMode	Boolean	Slave DTM shows with this bit that the feature is supported	Isochron_Mode_supp within GSD

Member Name	Type	Information source and meaning	Default Value
PrmDataIsochronModeRequired	Boolean	Slave DTM shows with this bit that the master is expected to support this feature true – master shall support isochrone mode false – isochrone mode is optional If set to true, then PrmDataIsochronMode shall be set to true too.	Isochron_Mode_required within GSD
PrmDataPrmCmd	Boolean	Slave DTM shows with this bit that the feature is supported	PrmCmd_supp within GSD
PrmDataUsrPrmData	Byte Array[a]	Calculated by Slave DTM	Data within GSD
CfgData	Byte Array[a]	Slave DTM, Depending on Module Configuration	Data within Slave GSD (Module, EndModule)
AddTabData	Byte Array[a]	Address assignment Table (only for DP-V0 Masters) Calculated by the Communication DTM	
SlaveUserData	Byte Array[a]	Calculated by the Communication DTM	Data within GSD
ExtPrmData	Byte Array[a]	Slave DTM	Data within GSD
MaxModules	Word	Slave DTM	Data within GSD
MaxInputLen	Word	Slave DTM	Data within GSD
MaxOutputLen	Word	Slave DTM	Data within GSD
MaxDataLen	Word	Slave DTM	Data within GSD
CurrentInputLen	Word	Calculated by Slave DTM for current configuration	Data within GSD
CurrentOutputLen	Word	Calculated by Slave DTM for current configuration	Data within GSD

[a] If this data is not applicable to the device (i.e. the service is not supported), then the ByteArray has zero length.

6.5.2.3 Modification of the PND

6.5.2.3.1 Propagation of changes

The PND includes parameter and configuration data. The slave DTM or the Frame Application may modify the PND.

The system shall ensure that the Communication Channel representing the PROFIBUS master is aware of these changes. This is achieved by sending the event NetworkDataInfoChanged from the Slave DTM to report the change of the PND. All changes should be reported as soon as possible, but not before the changes are persistent. The Frame Application informs the Parent DTM via NetworkDataInfoChanged event that parameters of a child have been changed. Then the Communication DTM can get the new PND of the Slave DTM.

6.5.2.3.2 Conditions for changing the PND

According to IEC TR 62453-42, it is allowed to change the parameters of a DTM starting from “running” state (see IEC TR 62453-42:2016, 6.6).

The PND can be changed multiple times, but only if the DTM is in offline mode and if the data set can be locked.

If a Slave DTM wants to change parameters in online mode it shall use DP-V0 or DP-V1 transaction requests. If there is no way of changing the parameters by transaction requests, the DTM shall disable configuration and parameterization in the online state.

6.5.2.3.3 Parameter data

If the user changes User Parameters of the BIM or of one module (e.g. via user interface) and this affects the PND, consequently the DTM shall update the PND. In addition to this, it shall request a save and inform the Frame Application via NetworkDataInfoChanged. The Frame Application shall distribute the information to all relevant components.

6.5.2.3.4 Configuration data

Configuration data will change every time if the user adds/removes modules or changes the module type, etc.

In the case of the modular DTM, the BIM will be informed when the user adds or removes modules via the service <ChildAdded> and service <ChildRemoved>. Changes of the parameters in a module will be reported by service NetworkDataInfoChanged.

The monolithic DTM or the BIM DTM update their PND data, request a save and inform the Frame Application via NetworkDataInfoChanged.

Changes that affect the PND often take effect on the internal topology and the Process Data Items. This information shall be updated by the DTMs too.

The PND can be changed by the Slave DTM and by the Communication DTM. The Communication DTM shall not change the configuration data and the user parameters parts of the PND.

6.5.2.4 Special cases related to the PND

6.5.2.4.1 DP-V1 support

In the GSD file there are two flags regarding DP-V1. At first, the "DPV1_Slave" value: This means that the slave has the possibility to work as a DP-V1 slave. If this value exists and the value is "1", then the SlaveFlagDpv1Slave shall be set to true. For older systems, there should also be the possibility to work as a DP-V0 slave.

Only the Communication DTM knows that its master device is able to provide acyclic services.

After building the Slave Bus Parameter Set, the Communication DTM will receive the slave's initial PND. If the SlaveFlagDpv1Slave is set to true, the Master DTM shall set the highest bit in the first byte of Extended DP-V1 Status. Now the slave works as a DP-V1 slave.

6.5.2.4.2 Extended DP-V1 status

All the PrmData<...> bits are set by the Communication DTM for the master. A Slave DTM shall accept these settings and adapt its functionality if necessary.

6.5.3 GSD Information

6.5.3.1 General

The GSD information is not stored with single slave instances or in a global accessible file. It is provided by the DTM via the PND in the service GetNetworkDataInfo.

Some existing DCS use the GSD file directly to obtain information about the possible configuration and parameters of a DP Slave. This behaviour is not recommended for developing future DCS.

Besides, the information about modules and its parameters, a GSD file contains additional information about the slave, such as the supported baud rates.

This information is useful for a DCS system to configure an entire network according to the capabilities of different slaves.

In order to support DCS, a DTM of a PROFIBUS device shall provide a separate GSD file. It shall be referenced as a document of type TechnicalDocumentation in the DeviceDescriptionReference property of the PND.

6.5.3.2 GSD for gateway devices

6.5.3.2.1 Types of PROFIBUS gateway devices

There are two types of gateway devices.

– The visible gateway devices work as a PROFIBUS slave (with a PROFIBUS station address) and to the underlying network they act as a master. Slave devices behind such a visible gateway have a separate address space and are addressed by extended addressing from the PROFIBUS master.
– The transparent gateway devices just transform the PROFIBUS network to the underlying network. Slave devices behind an invisible gateway share the address space with the devices on the rest of the PROFIBUS segment. They are addressed via normal station address by the PROFIBUS master.

For both types of devices there is a need for special GSD files to support legacy DCS as mentioned before.

6.5.3.2.2 Visible gateway devices

Visible gateway devices are shipped without a GSD file. Instead they have a proprietary software suite that configures and parameterizes it or they are shipped with a tool that creates a GSD for parameterization software. The GSD tool creates a GSD for the gateway device depending on the underlying network configuration or bus settings (e.g. baud rates).

The DTMs for such visible gateway devices (Gateway DTM) should provide the functionality of the GSD tool. If the GSD is configuration-dependent, the DTM could call service GetNetworkDataInfo for each of its children, extract the GSD information and create configuration-dependent GSD information in the same way the tool does. After initialization of the DTM, it should deliver PND according to the linking device itself. Every time a child is added or removed, this leads to a change in the network data or process data or both of the Gateway DTM.

If the GSD depends on bus settings, a DTM’s configuration or parameterization dialog could be used to change bus settings. Based on these settings, updated GSD information can be inserted in the DTM information. Here again the DTM has to request a slave and raise a ProcessDataChanged event if process data was changed and a NetworkDataInfoChanged event if network data was changed.

6.5.3.2.3 Transparent gateway devices

There are transparent linking devices on the market (e.g. PROFIBUS FMS/DP and PROFIBUS PA) performing a baudrate transformation. This requires a special handling of the slave specific GSD files. There are tools available which are able to adapt existing GSD files according to the higher baudrate (so called ‘GSD Converter’).

The GSD information shall be delivered by the DTM for the device. In order to support this kind of linking devices, a slave DTM shall expose the GSD file on hard drive.

6.5.4 Process Data Items

A device can offer a number of process values depending on the actual configuration. Information about these values is provided via Process Data Items. The protocol specific classes for this purpose are described in Clause 10 Datatypes for process data information.

If the process data is also accessible as Device Data Info or there is a relation to a communication channel (for instance for gateway DTMs), these relations shall be made available as IOSignalRefs.

7 Protocol specific usage of general datatypes

7.1 General datatypes

IEC TR 62453-42 already defines a set of datatypes that can be used to identify a device and to provide device information. Table 4 shows how general datatypes are used with IEC 61784 CP 3/1 and CP 3/2 devices.

Table 4 – Protocol specific usage of general datatypes

Attribute	Description for use
Address	The station address of the PROFIBUS slave device. Shall be formatted as a decimal number without leading zeros when represented as string.
DeviceTypeId	The DeviceTypeId shall contain the Ident_Number of the supported physical device. The IDENT_NUMBER shall be represented in hexadecimal format with 4 hex digits, i.e. "0x0815.
HardwareRevision	The hardware revision of the physical device.
ManufacturerId	Manufacturer according to Profile specification. For example in PROFIBUS PA: Physical Block Index 10: DEVICE_MAN_ID
PhysicalLayer	See Clause 4
ProtocolId	See Clause 4

7.2 Protocol specific handling of the datatype STRING

PROFIBUS uses strings in the form of character arrays. These arrays usually have a fixed length. For interaction with FDT, the following rules shall apply:

– leading spaces are left trimmed;

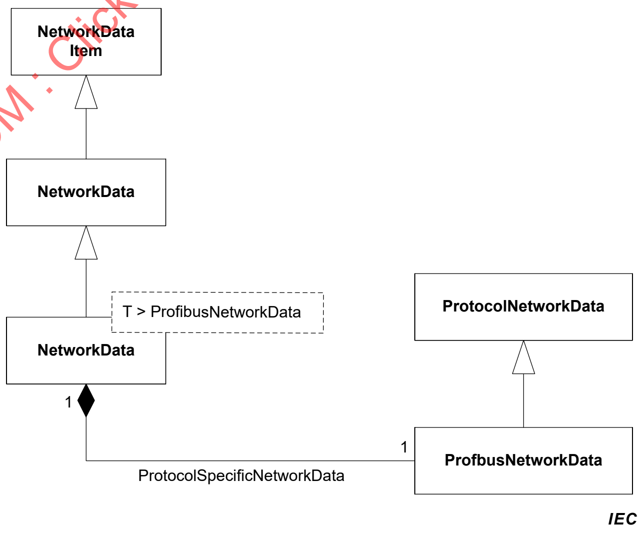
– arrays shall be filled with space characters (0x20);

– non-printable characters in VisibleStrings shall be replaced by '?'.

8 Network management datatypes

8.1 General

The data needed for management of the network are exposed by the Device DTM in the INetworkData interface (see Figure 3).



Used in:

INetworkData.GetNetworkDataInfo()

Figure 3 – ProfibusNetworkData

The properties of ProfibusNetworkData are described in Table 6.

The datatypes specified in this subclause are used in the following services:

– GetNetworkDataInfo service;

– SetNetworkData service.

The datatype ProfibusDeviceAddress is used for defining the network address of a device.

The protocol specific datatypes are based on definitions given in IEC 61784 and IEC 61158 (as described for each data type). Furthermore, they contain additional information about the device that is needed by systems to configure CPF 3 links and to establish communication between the CPF 3 master device and the CPF 3 slave devices.

8.2 Configuration

The configuration of the device itself is done with the aid of the DTM's GUI. Downloading the configuration into the slave device is performed via the CPF 3 master device. To do that and in order to set up the bus communication, the master needs information from the DTM as there is:

– GSD file
  The GSD information is type-specific information and not instance-specific (with the exception of certain gateway devices as described earlier). It is not stored with single slave instances or in a global accessible file.

The master device can use the general type-specific information from the slave's GSD information like bus timing parameters, supported baud rates, etc.

– CFG string (Cfg Data)
  The CFG string provides the instance-specific information about the current configuration of the device. It defines the structure of the data frames that will be transmitted on the PROFIBUS. This structure depends on the modules that are actually configured.

The DTM provides the CFG string within the property CfgData that is part of the PROFIBUS Network Data available via service GetNetworkDataInfo.

The master device uses this information to set up communication with the slave device.

8.3 Process Data Items

In case of CPF 3 protocols, a FDT Process Data Item is a representative for a single date or a process value that can be accessed from a Frame Application via the master device. The information available at services for I/O related information describes how to access a data item via a PROFIBUS DP-V1 command or how to address a data item within a PROFIBUS DP frame for cyclic I/O. Besides all mandatory elements (which include id, BitPosition and BitLength) it is highly recommended that the DP-V1 address information is provided. This information (DP-V1 Slot) is used by some frames to manage the PROFIBUS device module information.

8.4 Parameterization

There are two options to write parameters set from the DTM's GUI to the CPF 3 slave device in the field:

– User Parameters
  User Parameters are part of the PROFIBUS-DP Slave-Bus-Parameter-Set. They contain manufacturer-specific data to characterize the DP slave. The DTM stores the User Parameters in property PrmDataUsrPrmData of the PROFIBUS Network Data. The User Parameters are stored with the master device during PROFIBUS master configuration and are automatically sent to the slave during set up of bus communication.

  NOTE This process is PROFIBUS-specific. For details, see IEC 61158 series.

When changing User Parameters at runtime, the DTM shall use a DP-V0 connection and the appropriate DP-V0 commands for parameter exchange as described in the datatypes.

– Writing Parameters with DP-V1 services (MSAC2 primitives)
  The DTM may use DP-V1 transport services to send its parameters to the slave device. For that, it has to use a DP-V1 connection and the corresponding communication commands. During setup of communication, DP-V1 services are not sent automatically. The Frame Application or a DTM shall invoke a download of parameters via DP-V1.

For details on the different behaviour of slaves depending on the kind of parameterization, refer to the IEC 61158 series.

DP-V1 connections and communication commands can also be used to execute commands at the slave. For details on the use of DP-V1, see also the IEC 61158 series.

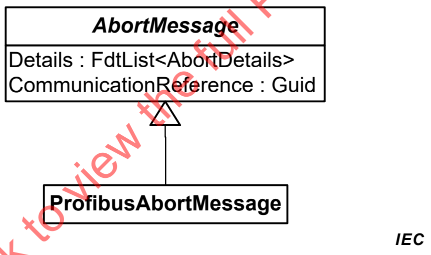
9 Communication datatypes

9.1 General

The datatypes contain the address information and the communication data required to execute the respective request or to transport the response information.

9.2 ProfibusAbortMessage

This is the PROFIBUS specific implementation of the abstract AbortMessage class (see Figure 4).


AbortMessage
Details : FdtList<AbortDetails>
CommunicationReference : Guid
ProfibusAbortMessage
IEC

Used in:

ICommunication.EndDisconnect()

Figure 4 – ProfibusAbortMessage

The properties of the ProfibusAbortMessage datatype are described in Table 5.

Table 5 – ProfibusAbortMessage datatype

Property	Description
CommunicationReference	Identifier for a communication link to a device.
Details	Details about the cause and source of the Abort.

9.3 DP-V0 Communication

9.3.1 General

Not all defined services are supported if the Master is not in cyclic data exchange with the slaves. In such cases, the following behaviour is expected:

If a Communication Channel receives a request that cannot be supported, the ErrorInformation property of the Transaction response shall be set to "NotSupportedFeature".

Depending on the bus master type and on the returned ConnectStatus, the following services are available (see Table 6).

Table 6 – Availability of services for Master Class 1 (C1)

	ConnectStatus		
Slave DTM Service Request	**MasterConnectedOnly**	**DeviceAtLifeList**	**DeviceInDataExchange**
Connect	✓	✓	✓
ReadUserParameter	O	O	O
WriteUserParameter	✓	✓	✓
ReadOutputData			✓
WriteOutputData			O
ReadInputData			✓
ReadDiagnosisData		✓	✓
ReadConfigurationData		✓	✓
NOTE ✓: the service is available, O: the service is optional and can be available, depending on the capabilities of the underlying master device.			

For Master Class 2 (C2), not all connect states are available, see Table 7 below.

Table 7 – Availability of services for Master Class 2 (C2)

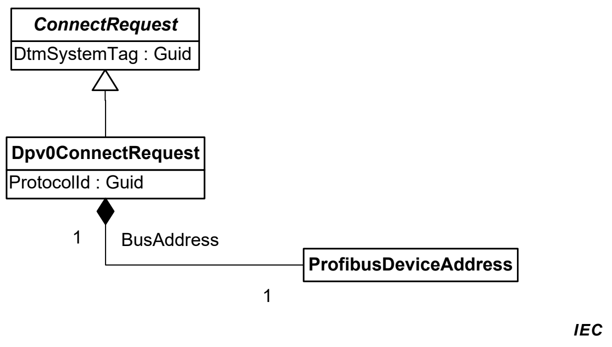
	ConnectStatus	
Slave DTM Action	**DeviceAtLifeList (no DP-V1 connection to device)**	**DeviceInDataExchange (DP-V1 connection to device)**
Connect	✓	✓
ReadUserParameter		
WriteUserParameter	O	O
ReadOutputData		O
WriteOutputData		
ReadInputData		✓
ReadDiagnosisData	✓	✓
ReadConfigurationData	O	O
NOTE ✓: the service is available, O: the service is optional and can be available, depending on the capabilities of the underlying master device.		

If the Master Class 2 communication component supports DP-V1 and DP-V0 and has established a DP-V1 connection to the device, a call to service Connect for DP-V0 shall return the status “DeviceInDataExchange” even if the device is not in status DataExchange.

If no DP-V1 connection is established, the Master Class 2 communication component shall verify the availability of the device (at least by service LifeList) prior to returning the result.

9.3.2 Dpv0ConnectRequest

This is the PROFIBUS DP-V0 specific implementation of the abstract class ConnectRequest (see Figure 5).



Used in:

ICommunication.BeginConnect()

Figure 5 – Dpv0ConnectRequest

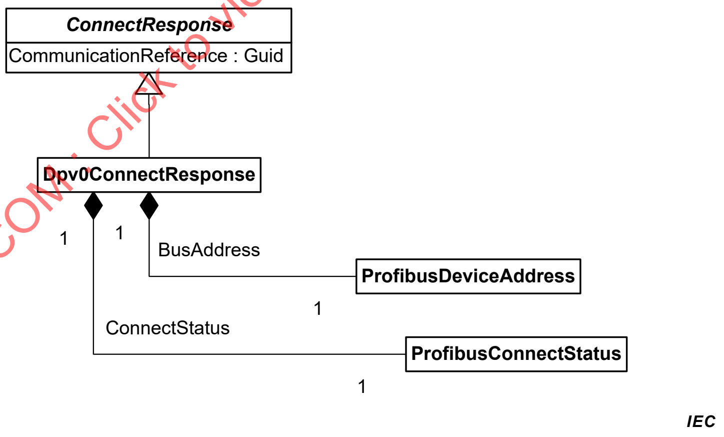
The properties of the Dpv0ConnectRequest datatype are described in Table 8.

Table 8 – Dpv0ConnectRequest datatype

Property	Description
BusAddress	Station address information according to the PROFIBUS specification.
ProtocolId	Unique identifier of the PROFIBUS DP-V0 protocol.
DtmSystemTag	Unique identification of the DTM in the Frame Application.

9.3.3 Dpv0ConnectResponse

This is the PROFIBUS DP-V0 specific implementation of the abstract class ConnectResponse (see Figure 6).



Used in:

ICommunication.EndConnect()

Figure 6 – Dpv0ConnectResponse

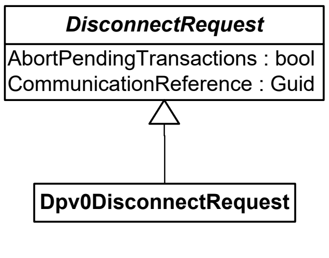
The properties of the Dpv0ConnectResponse datatype are described in Table 9.

Table 9 – Dpv0ConnectResponse datatype

Property	Description
BusAddress	Address information according to the PROFIBUS specification.
CommunicationReference	Identifier for a communication link to a device.
ConnectStatus	Describes the connection status established by the communication component.

9.3.4 Dpv0DisconnectRequest

This is the PROFIBUS DP-V0 specific implementation of the abstract class DisconnectRequest (see Figure 7).


DisconnectRequest
AbortPendingTransactions : bool
CommunicationReference : Guid
Dpv0DisconnectRequest
IEC

Used in:

ICommunication.BeginDisconnect()

Figure 7 – Dpv0DisconnectRequest

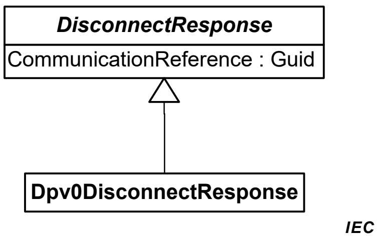
The properties of the Dpv0DisconnectRequest datatype are described in Table 10.

Table 10 – Dpv0DisconnectRequest datatype

Property	Description
CommunicationReference	Identifier for a communication link to a device.
AbortPendingTransactions	Indicates whether pending transactions shall be aborted.

9.3.5 Dpv0DisconnectResponse

This is the PROFIBUS DP-V0 specific implementation of the abstract class DisconnectResponse (see Figure 8).



Used in:

ICommunication.EndDisconnect()

Figure 8 – Dpv0DisconnectResponse

The properties of the Dpv0DisconnectResponse datatype are described in Table 11.

Table 11 – Dpv0DisconnectResponse datatype

Property	Description
CommunicationReference	Identifier for a communication link to a device.

9.3.6 Dpv0TransactionRequest

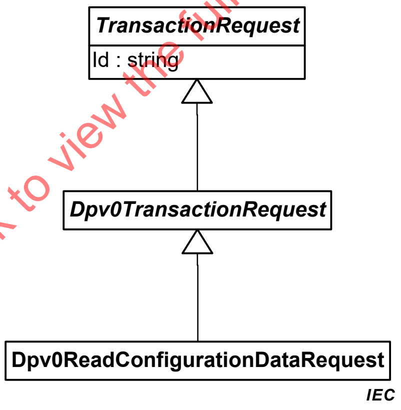
9.3.6.1 General

The Dpv0TransactionRequest is the base class for PROFIBUS DPV0 transaction requests. The following classes inherit from this class:

- Dpv0ReadConfigurationDataRequest
- Dpv0ReadDiagnosisDataRequest
- Dpv0ReadInputDataRequest
- Dpv0ReadOutputDataRequest
- Dpv0ReadUserParameterRequest
- Dpv0WriteOutputDataRequest
- Dpv0WriteUserParameterRequest

9.3.6.2 Dpv0ReadConfigurationDataRequest

This is the request for reading the Cfg Data from the device, derived from the base class Dpv0TransactionRequest (see Figure 9).



Used in:

ICommunication.BeginCommunicationRequest()

Figure 9 – Dpv0ReadConfigurationDataRequest

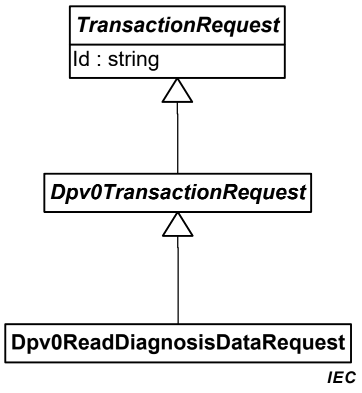
The properties of the Dpv0ReadConfigurationDataRequest datatype are described in Table 12.

Table 12 – Dpv0ReadConfigurationDataRequest datatype

Property	Description
Id	[Optional] Identifier for a single Transaction Request.

9.3.6.3 Dpv0ReadDiagnosisDataRequest

This is the request for reading the device diagnosis data, derived from the base class Dpv0TransactionRequest (see Figure 10).



Used in:

ICommunication.BeginCommunicationRequest()

Figure 10 – Dpv0ReadDiagnosisDataRequest

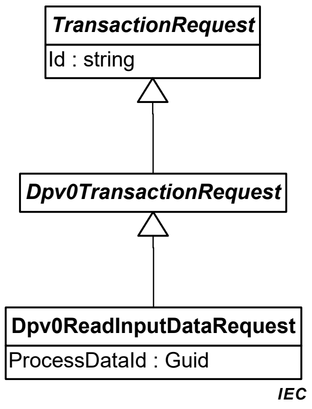
The properties of the Dpv0ReadDiagnosisDataRequest datatype are described in Table 13.

Table 13 – Dpv0ReadDiagnosisDataRequest datatype

Property	Description
Id	[Optional] Identifier for a single Transaction Request.

9.3.6.4 Dpv0ReadInputDataRequest

This is the request for reading the device input data, derived from the base class Dpv0TransactionRequest (see Figure 11).



Used in:

ICommunication.BeginCommunicationRequest()

Figure 11 – Dpv0ReadInputDataRequest

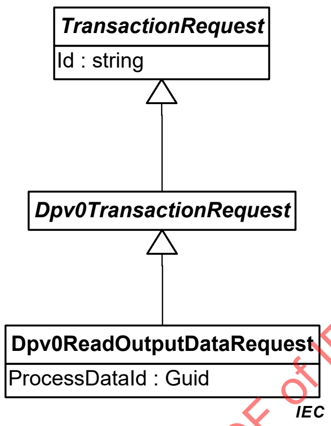
The properties of the Dpv0ReadInputDataRequest datatype are described in Table 14.

Table 14 – Dpv0ReadInputDataRequest datatype

Property	Description
ProcessDataId	Reference to IO Signal defining the data properties.
Id	[Optional] Identifier for a single Transaction Request.

9.3.6.5 Dpv0ReadOutputDataRequest

This is the request for reading the device output data, derived from the base class Dpv0TransactionRequest (see Figure 12).



Used in:

ICommunication.BeginCommunicationRequest()

Figure 12 – Dpv0ReadOutputDataRequest

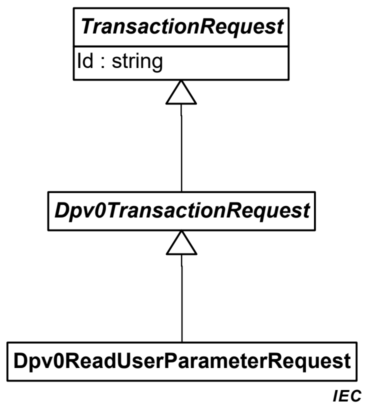
The properties of the Dpv0ReadOutputDataRequest datatype are described in Table 15.

Table 15 – Dpv0ReadOutputDataRequest datatype

Property	Description
ProcessDataId	Reference to IO Signal defining the data properties.
Id	[Optional] Identifier for a single Transaction Request.

9.3.6.6 Dpv0ReadUserParameterRequest

This is the request for reading the User Prm Data from the device, derived from the base class Dpv0TransactionRequest (see Figure 13).



Used in:

ICommunication.BeginCommunicationRequest()

Figure 13 – Dpv0ReadUserParameterRequest

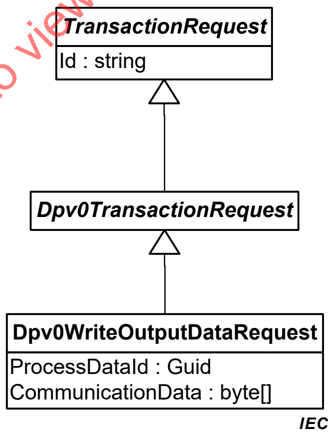
The properties of the Dpv0ReadUserParameterRequest datatype are described in Table 16.

Table 16 – Dpv0ReadUserParameterRequest datatype

Property	Description
Id	[Optional] Identifier for a single Transaction Request.

9.3.6.7 Dpv0WriteOutputDataRequest

This is the request for writing the device output data, derived from the base class Dpv0TransactionRequest (see Figure 14).



Used in:

ICommunication.BeginCommunicationRequest()

Figure 14 – Dpv0WriteOutputDataRequest

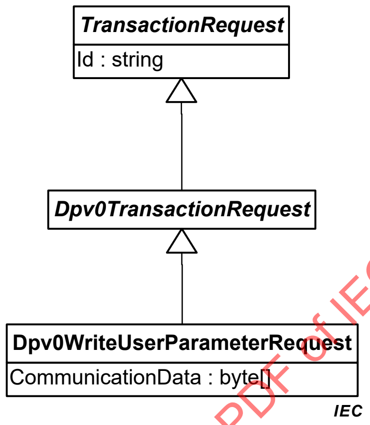
The properties of the Dpv0WriteOutputDataRequest datatype are described in Table 17.

Table 17 – Dpv0WriteOutputDataRequest datatype

Property	Description
ProcessDataId	Reference to IO Signal defining the data properties.
CommunicationData	Array of bytes to be written.
Id	[Optional] Identifier for a single Transaction Request.

9.3.6.8 Dpv0WriteUserParameterRequest

This is the request for writing the User Prm Data, derived from the base class Dpv0TransactionRequest (see Figure 15).



Used in:

ICommunication.BeginCommunicationRequest()

Figure 15 – Dpv0WriteUserParameterRequest

The properties of the Dpv0WriteUserParameterRequest datatype are described in Table 18.

Table 18 – Dpv0WriteUserParameterRequest datatype

Property	Description
CommunicationData	Array of bytes to be written.
Id	[Optional] Identifier for a single Transaction Request.

9.3.7 Dpv0TransactionResponse

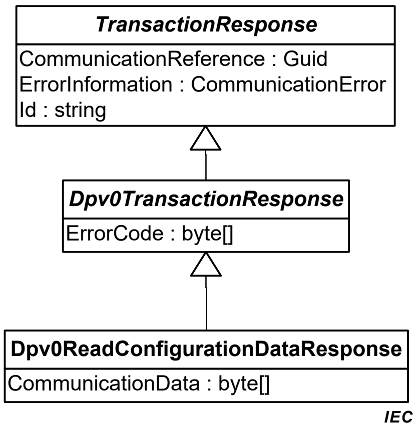
9.3.7.1 General

This is the base class for PROFIBUS DP-V0 transaction responses containing the common property ErrorCode (see 9.5). The following classes inherit from this class:

- Dpv0ReadConfigurationDataResponse
- Dpv0ReadDiagnosisDataResponse
- Dpv0ReadInputDataResponse
- Dpv0ReadOutputDataResponse
- Dpv0ReadUserParameterResponse
- Dpv0WriteOutputDataResponse
- Dpv0WriteUserParameterResponse

9.3.7.2 Dpv0ReadConfigurationDataResponse

This is the response for reading the Cfg Data (see Figure 16).


TransactionResponse
CommunicationReference : Guid
ErrorInformation : CommunicationError
Id : string
Dpv0TransactionResponse
ErrorCode : byte[]
Dpv0ReadConfigurationDataResponse
CommunicationData : byte[]
IEC

Used in:

ICommunication.EndCommunicationRequest()

Figure 16 – Dpv0ReadConfigurationDataResponse

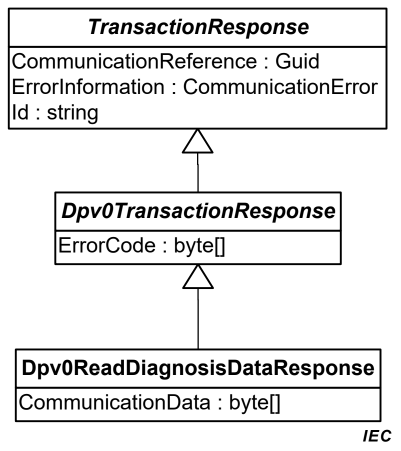
The properties of the Dpv0ReadConfigurationDataResponse datatype are described in Table 19.

Table 19 – Dpv0ReadConfigurationDataResponse datatype

Property	Description
CommunicationReference	Identifier for a communication link to a device.
ErrorInformation	[Optional] Description of a fieldbus protocol independent error occurred during communication.
Id	[Optional] Identifier of the corresponding Transaction Request.
CommunicationData	Communication data received from the device.
ErrorCode	The result of the service call.

9.3.7.3 Dpv0ReadDiagnosisDataResponse

This is the response for reading the diagnosis data (see Figure 17).


TransactionResponse
CommunicationReference : Guid
ErrorInformation : CommunicationError
Id : string
Dpv0TransactionResponse
ErrorCode : byte[]
Dpv0ReadDiagnosisDataResponse
CommunicationData : byte[]
IEC

Used in:

ICommunication.EndCommunicationRequest()

Figure 17 – Dpv0ReadDiagnosisDataResponse

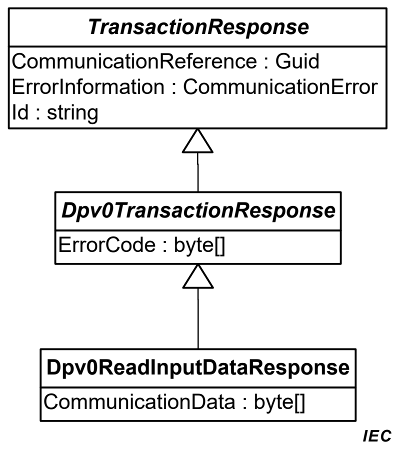
The properties of the Dpv0ReadDiagnosisDataResponse datatype are described in Table 20.

Table 20 – Dpv0ReadDiagnosisDataResponse datatype

Property	Description
CommunicationReference	Identifier for a communication link to a device.
ErrorInformation	[Optional] Description of a fieldbus protocol independent error occurred during communication.
Id	[Optional] Identifier of the corresponding Transaction Request.
CommunicationData	Communication data received from the device.
ErrorCode	The result of the service call.

9.3.7.4 Dpv0ReadInputDataResponse

This is the response for reading the input data (see Figure 18).


TransactionResponse
CommunicationReference : Guid
ErrorInformation : CommunicationError
Id : string
Dpv0TransactionResponse
ErrorCode : byte[]
Dpv0ReadInputDataResponse
CommunicationData : byte[]
IEC

Used in:

ICommunication.EndCommunicationRequest()

Figure 18 – Dpv0ReadInputDataResponse

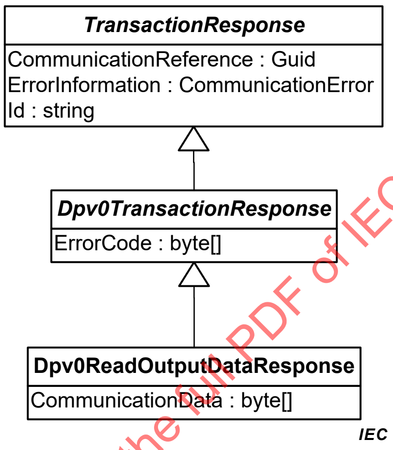
The properties of the Dpv0ReadInputDataResponse datatype are described in Table 21.

Table 21 – Dpv0ReadInputDataResponse datatype

Property	Description
CommunicationReference	Identifier for a communication link to a device.
ErrorInformation	[Optional] Description of a fieldbus protocol independent error occurred during communication.
Id	[Optional] Identifier of the corresponding Transaction Request.
CommunicationData	Communication data received from the device.
ErrorCode	The result of the service call.

9.3.7.5 Dpv0ReadOutputDataResponse

This is the response for reading the output data (see Figure 19).


TransactionResponse
CommunicationReference : Guid
ErrorInformation : CommunicationError
Id : string
Dpv0TransactionResponse
ErrorCode : byte[]
Dpv0ReadOutputDataResponse
CommunicationData : byte[]
IEC

Used in:

ICommunication.EndCommunicationRequest()

Figure 19 – Dpv0ReadOutputDataResponse

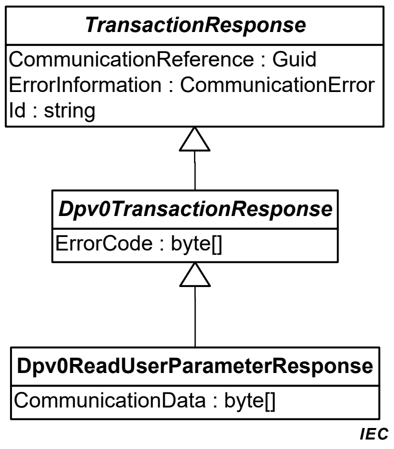
The properties of the Dpv0ReadOutputDataResponse datatype are described in Table 22.

Table 22 – Dpv0ReadOutputDataResponse datatype

Property	Description
CommunicationReference	Identifier for a communication link to a device.
ErrorInformation	[Optional] Description of a fieldbus protocol independent error occurred during communication.
Id	[Optional] Identifier of the corresponding Transaction Request.
CommunicationData	Communication data received from the device.
ErrorCode	The result of the service call.

9.3.7.6 Dpv0ReadUserParameterResponse

This is the response for reading the Usr Prm Data (see Figure 20).


TransactionResponse
CommunicationReference : Guid
ErrorInformation : CommunicationError
Id : string
Dpv0TransactionResponse
ErrorCode : byte[]
Dpv0ReadUserParameterResponse
CommunicationData : byte[]
IEC

Used in:

ICommunication.EndCommunicationRequest()

Figure 20 – Dpv0ReadUserParameterResponse

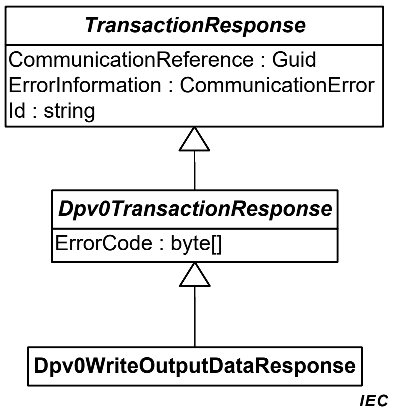
The properties of the Dpv0ReadUserParameterResponse datatype are described in Table 23.

Table 23 – Dpv0ReadUserParameterResponse datatype

Property	Description
CommunicationReference	Identifier for a communication link to a device.
ErrorInformation	[Optional] Description of a fieldbus protocol independent error occurred during communication.
Id	[Optional] Identifier of the corresponding Transaction Request.
CommunicationData	Communication data received from the device.
ErrorCode	The result of the service call.

9.3.7.7 Dpv0WriteOutputDataResponse

This is the response for writing the output data (see Figure 21).


TransactionResponse
CommunicationReference : Guid
ErrorInformation : CommunicationError
Id : string
Dpv0TransactionResponse
ErrorCode : byte[]
Dpv0WriteOutputDataResponse
IEC

Used in:

ICommunication.EndCommunicationRequest()

Figure 21 – Dpv0WriteOutputDataResponse

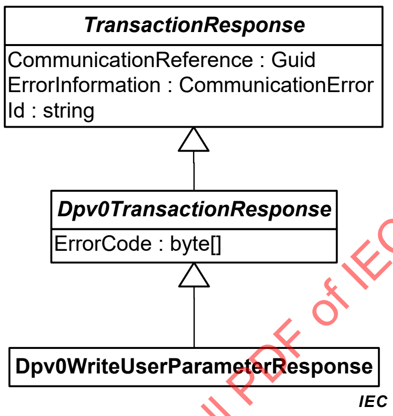
The properties of the Dpv0WriteOutputDataResponse datatype are described in Table 24.

Table 24 – Dpv0WriteOutputDataResponse datatype

Property	Description
CommunicationReference	Identifier for a communication link to a device.
ErrorInformation	[Optional] Description of a fieldbus protocol independent error occurred during communication.
Id	[Optional] Identifier of the corresponding Transaction Request.
ErrorCode	The result of the service call.

9.3.7.8 Dpv0WriteUserParameterResponse

This is the response for writing the Usr Prm Data (see Figure 22).



TransactionResponse
CommunicationReference : Guid
ErrorInformation : CommunicationError
Id : string
Dpv0TransactionResponse
ErrorCode : byte[]
Dpv0WriteUserParameterResponse
IEC

Used in:

ICommunication.EndCommunicationRequest()

Figure 22 – Dpv0WriteUserParameterResponse

The properties of the Dpv0WriteUserParameterResponse datatype are described in Table 25.

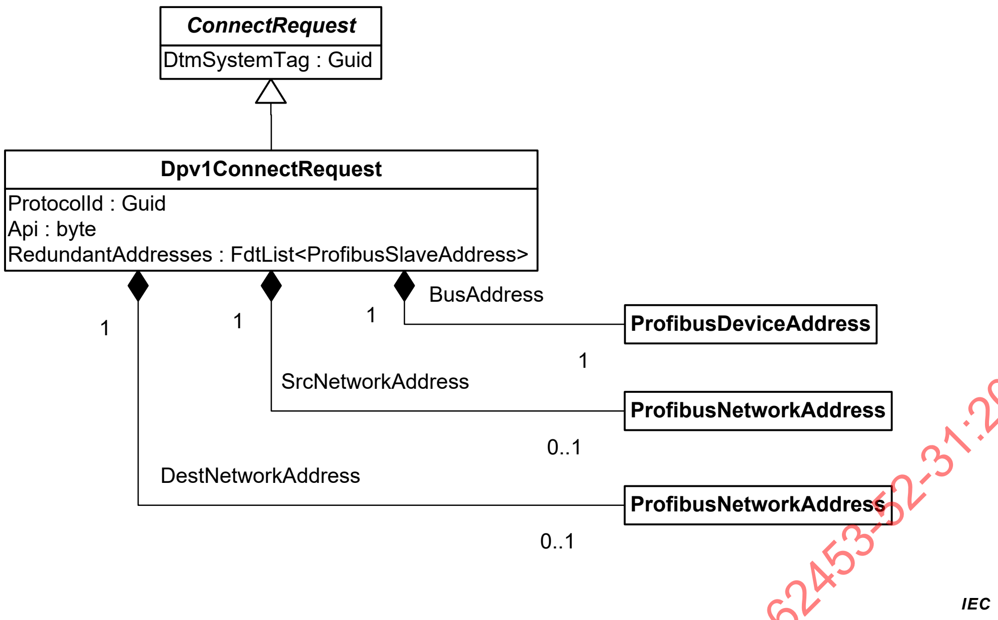
Table 25 – Dpv0WriteUserParameterResponse datatype

Property	Description
CommunicationReference	Identifier for a communication link to a device.
ErrorInformation	[Optional] Description of a fieldbus protocol independent error occurred during communication.
Id	[Optional] Identifier of the corresponding Transaction Request.
ErrorCode	The result of the service call.

9.4 DP-V1 Communication

9.4.1 Dpv1ConnectRequest

This is the PROFIBUS DP-V1 specific implementation of the abstract class ConnectRequest (see Figure 23).



Used in:

ICommunication.BeginConnect()

Figure 23 – Dpv1ConnectRequest

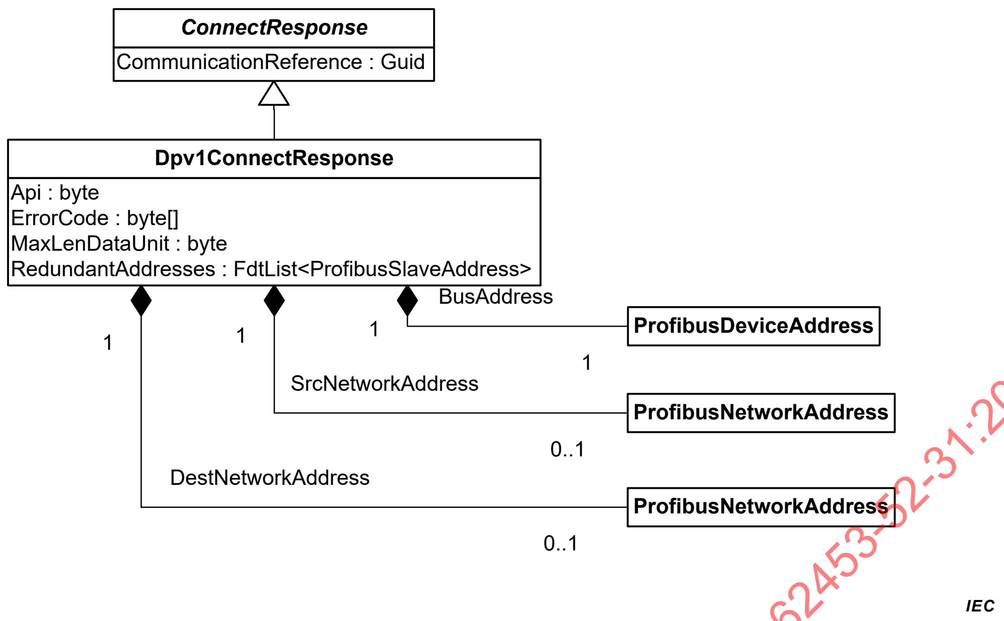
The properties of the Dpv1ConnectRequest datatype are described in Table 26.

Table 26 – Dpv1ConnectRequest datatype

Property	Description
CommunicationReference	Identifier for a communication link to a device.
ProtocolId	Unique identifier of the PROFIBUS DP-V1 protocol.
DtmSystemTag	Unique identification of the DTM in the Frame Application.
Api	Additional address information. If the device needs special values for the DPV1-Initiate, the DeviceDTM is responsible for providing the values in the ConnectRequest.
BusAddress	Address information according to the PROFIBUS specification.
SrcNetworkAddress	[optional] Describes the extended address of the source (required for inter-network addressing).
DestNetworkAddress	[optional] Describes the extended address of the destination (required for inter-network addressing).
RedundantAddresses	[optional] Within a connect request, a DTM of a PROFIBUS redundant slave can provide additional redundant slave addresses. The BusAddress property is to be used as the preferred address. The addresses in the RedundantAddresses property should be used in the order of the list if an alternative address is used to connect to the redundant slave.

9.4.2 Dpv1ConnectResponse

This is the PROFIBUS DP-V1 specific implementation of the abstract class ConnectResponse (see Figure 24).



Used in:

ICommunication.EndConnect()

Figure 24 – Dpv1ConnectResponse

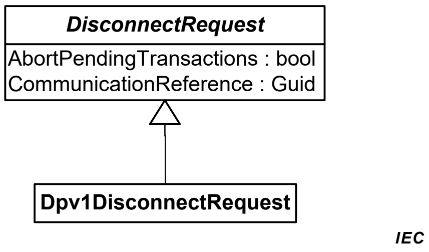
The properties of the Dpv1ConnectResponse datatype are described in Table 27.

Table 27 – Dpv1ConnectResponse datatype

Property	Description
CommunicationReference	Identifier for a communication link to a device.
Api	Additional addressing information. If the device needs special values for the DPV1-Initiate, the DeviceDTM is responsible for providing the values in the ConnectRequest.
BusAddress	Address information according to the PROFIBUS specification.
MaxLenDataUnit	[optional] Describes the amount of data, which can be transferred via the established connection. If this property is not available, no special length restriction is announced. Each communication component within the chain of interfaces concerning nested communication could introduce this property. Each communication component should change the contents of the property based on the following rule: The new value is the minimum of the current value and the restriction of its own implementation. If a communication component has no restriction, it should hand over the given value. If a communication component is able to reuse an established connection concerning a new connect request, it should take into account the data length determined for the existing connection.
SrcNetworkAddress	[optional] Describes the extended address of the source.
DestNetworkAddress	[optional] Describes the extended address of the destination.
RedundantAddresses	[optional] Within a connect request, a DTM of a PROFIBUS redundant slave can provide additional redundant slave addresses. The BusAddress property is to be used as the preferred address.The addresses in the RedundantAddresses property should be used in the order of the list if an alternative address is used to connect to the redundant slave.
ErrorCode	The result of the service call.

9.4.3 Dpv1DisconnectRequest

This is the PROFIBUS DP-V1 specific implementation of the abstract class DisconnectRequest (see Figure 25).


DisconnectRequest
AbortPendingTransactions : bool
CommunicationReference : Guid
Dpv1DisconnectRequest
IEC

Used in:

ICommunication.BeginDisconnect()

Figure 25 – Dpv1DisconnectRequest

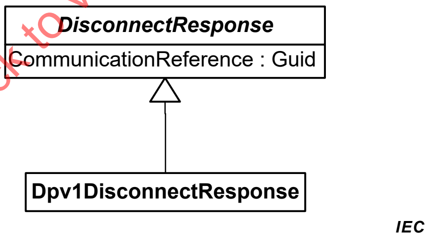
The properties of the Dpv1DisconnectRequest datatype are described in Table 28.

Table 28 – Dpv1DisconnectRequest datatype

Property	Description
CommunicationReference	Identifier for a communication link to a device.
AbortPendingTransactions	Indicates whether pending transactions shall be aborted.

9.4.4 Dpv1DisconnectResponse

This is the PROFIBUS DP-V1 specific implementation of the abstract class DisconnectResponse (see Figure 26).



Used in:

ICommunication.EndDisconnect()

Figure 26 – Dpv1DisconnectResponse

The properties of the Dpv1DisconnectResponse datatype are described in Table 29.

Table 29 – Dpv1DisconnectResponse datatype

Property	Description
CommunicationReference	Identifier for a communication link to a device.

9.4.5 Dpv1TransactionRequest

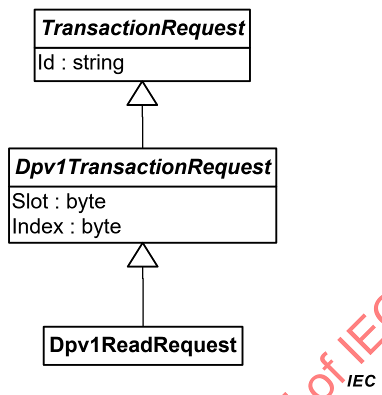
9.4.5.1 General

This is the base class for PROFIBUS DP-V1 transaction requests containing the common properties slot and index. The following classes inherit from this class:

Dpv1ReadRequest
Dpv1WriteRequest

9.4.5.2 Dpv1ReadRequest

This is the request for reading data from the device (see Figure 27).



Used in:

ICommunication.BeginCommunicationRequest()

Figure 27 – Dpv1ReadRequest

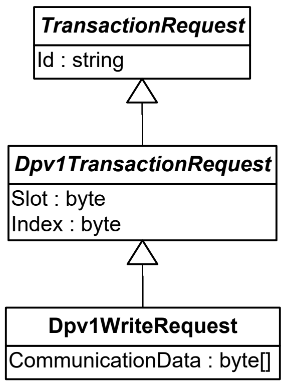
The properties of the Dpv1ReadRequest datatype are described in Table 30.

Table 30 – Dpv1ReadRequest datatype

Property	Description
Slot	Address information according to the PROFIBUS specification.
Index	Address information according to the PROFIBUS specification.
Id	[Optional] Identifier for a single Transaction Request.

9.4.5.3 Dpv1WriteRequest

This is the request for writing data to the device (see Figure 28).



Used in:

ICommunication.BeginCommunicationRequest ()

Figure 28 – Dpv1WriteRequest

The properties of the Dpv1WriteRequest datatype are described in Table 31.

Table 31 – Dpv1WriteRequest datatype

Property	Description
Slot	Address information according to the PROFIBUS specification.
Index	Address information according to the PROFIBUS specification.
CommunicationData	Array of bytes to be written.
Id	[Optional] Identifier for a single Transaction Request.

9.4.6 Dpv1TransactionResponse

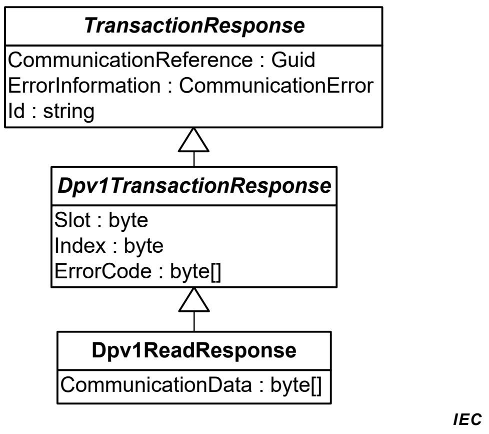
9.4.6.1 General

This is the base class for PROFIBUS DP-V1 transaction responses containing the common properties slot, index and error code. The following classes inherit from this class:

Dpv1ReadResponse
Dpv1WriteResponse

9.4.6.2 Dpv1ReadResponse

This is the response for reading data from the device (see Figure 29).


TransactionResponse
CommunicationReference : Guid
ErrorInformation : CommunicationError
Id : string
Dpv1TransactionResponse
Slot : byte
Index : byte
ErrorCode : byte[]
Dpv1ReadResponse
CommunicationData : byte[]
IEC

Used in:

ICommunication.EndCommunicationRequest ()

Figure 29 – Dpv1ReadResponse

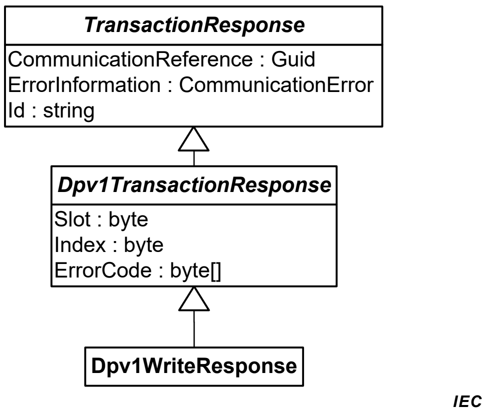
The properties of the Dpv1ReadResponse datatype are described in Table 32.

Table 32 – Dpv1ReadResponse datatype

Property	Description
CommunicationReference	Identifier for a communication link to a device.
ErrorInformation	[Optional] Description of a fieldbus protocol independent error occurred during communication.
Id	[Optional] Identifier of the corresponding Transaction Request.
Slot	Address information according to the PROFIBUS specification.
Index	Address information according to the PROFIBUS specification.
CommunicationData	Array of bytes read from device.
ErrorCode	The result of the service call (see 9.5).

9.4.6.3 Dpv1WriteResponse

This is the response for writing data to the device (see Figure 30).



Used in:

ICommunication.EndCommunicationRequest ()

Figure 30 – Dpv1WriteResponse

The properties of the Dpv1WriteResponse datatype are described in Table 33.

Table 33 – Dpv1WriteResponse datatype

Property	Description
CommunicationReference	Identifier for a communication link to a device.
ErrorInformation	[Optional] Description of a fieldbus protocol independent error occurred during communication.
Id	[Optional] Identifier of the corresponding Transaction Request.
Slot	Address information according to the PROFIBUS specification.
Index	Address information according to the PROFIBUS specification.
ErrorCode	The result of the service call (see 9.5).

9.5 Error information provided by Communication Channel

In every transaction response datatype of FDT PROFIBUS specification a property ‘ErrorCode’ is provided. According to PROFIBUS, the error code is standardized to consist of 3 bytes, where each byte carries a meaning.

Since the error code is exchanged between different DTMs (e.g. Communication-DTM and Device-DTM) and since the receiver of the error code will try to understand the error code, the provider shall use the standard format:

standard length 3 bytes;
if the device provides error codes, these error codes are provided (and not local error codes from the Master).
If no error occurred, the property ‘ErrorCode’ shall be filled with 3 zero bytes.

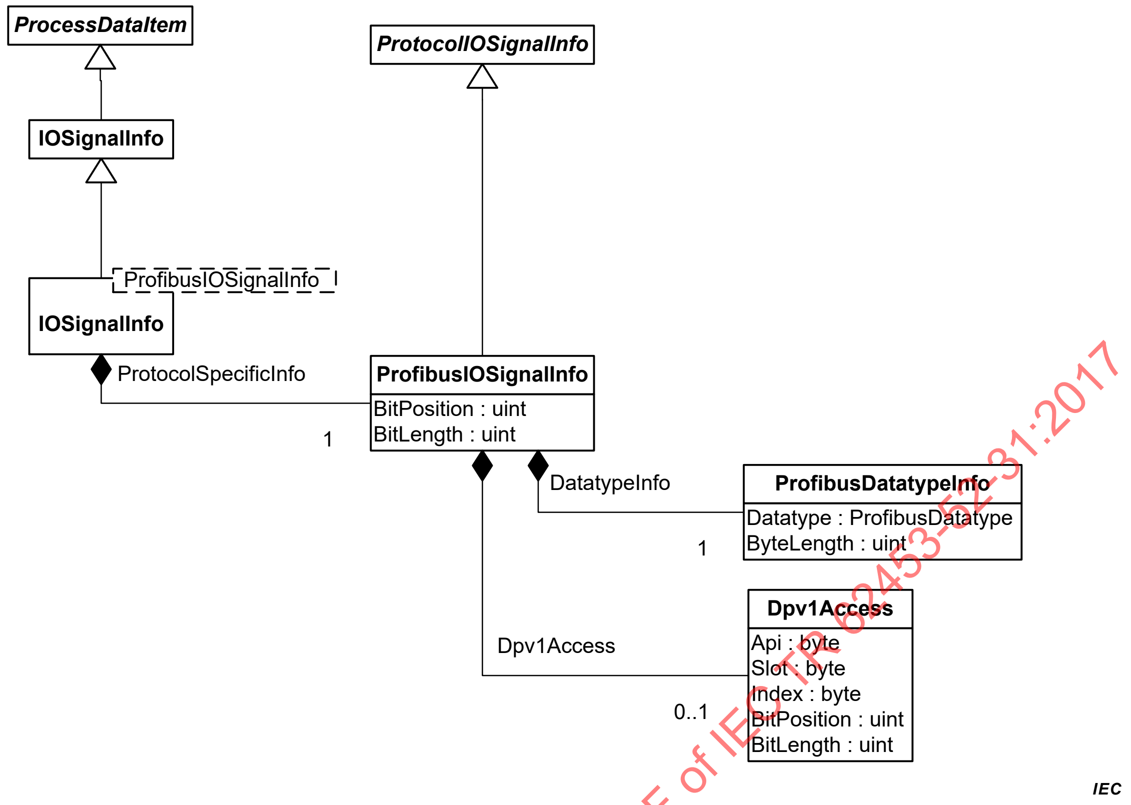
10 Datatypes for process data information

10.1 General

The process data information of a DTM represents the “Device Variables”, available on that device. A Process Control System (i.e. some external system, which monitors values on a device) can query the DTM's process data information via the IProcessData interface. The process data describes the process values such that an external system can use the information to access and interpret the values from the device during normal device runtime. The external system might not use FDT to access the values.

10.2 ProfibusIOSignalInfo

This is the PROFIBUS specific implementation of the abstract class ProtocolIOSignalInfo (see Figure 31).



Used in:

IProcessData.<ProcessData>()

IProcessData.SetIOSignalInfo()

Figure 31 – ProfibusIOSignalInfo

The properties of the ProfibusIOSignalInfo datatype are described in Table 34.

Table 34 – ProfibusIOSignalInfo datatype

Property	Description
DatatypeInfo	The datatype of the IO signal.
BitPosition	The position in the addressed data stream.
BitLength	The length of the data.
Dpv1Access	[optional] Describes how to access the IO data with DP-V1 protocol.
Api	The API value to access the value with DP-V1 protocol.
Slot	The slot part of the DP-V1 data address.
Index	The index part of the DP-V1 data address.
BitPosition	[optional] The position in the addressed data stream. If omitted, 0 shall be assumed.
BitLength	[optional] The length of the data. When omitted, the default length of the datatype shall be assumed.

11 Device identification

11.1 General

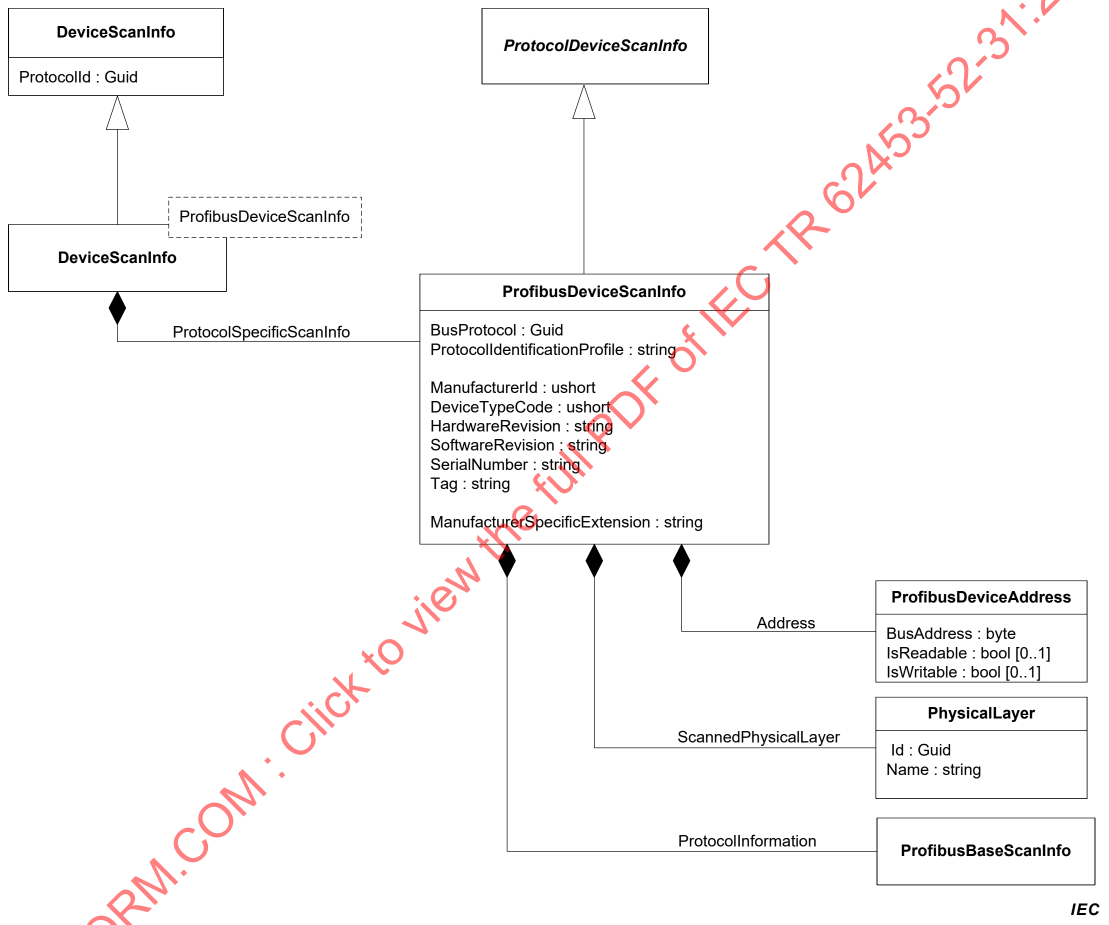
This clause defines identification relevant protocol specific datatypes.

A PROFIBUS scan may detect different device types: I&M devices, PROFIBUS PA devices or pure DP devices. Depending on the detected device type, not all properties of ProfibusDeviceScanInfo or ProfibusDeviceIdentInfo are available and will be filled with default values. The ProtocolIdentificationProfile property of the DeviceScanInfo or DeviceIdentInfo instances shall be set to either "DP", "PA", “IM-PA” or "IM" to indicate the identification type for the device.

11.2 ProfibusDeviceScanInfo datatype

11.2.1 General

This is the PROFIBUS specific implementation of the abstract class ProtocolDeviceScanInfo (see Figure 32).



Used in:

IDtmScanning.EndScanRequest()

Figure 32 – ProfibusDeviceScanInfo

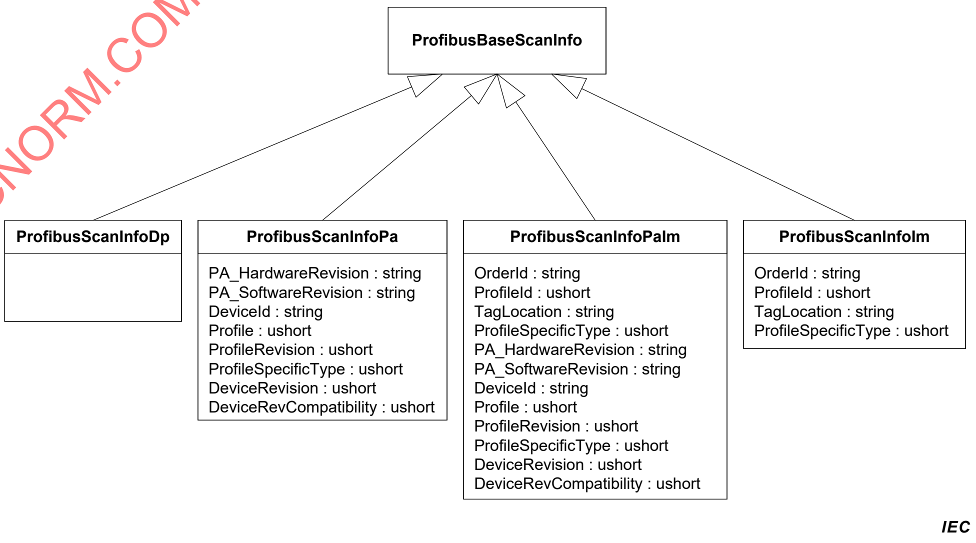
The properties of the ProfibusDeviceScanInfo datatype are described in Table 35. Protocol specific properties will be stored as key-value pairs in the property DeviceScanInfo.ProtocolSpecificProperties. Device specific properties will be stored as key-value pairs in the property DeviceScanInfo.DeviceSpecificProperties.

Table 35 – ProfibusDeviceScanInfo datatype

Property	Description
Address	The bus address of the device.
BusProtocol	Can be set to either DP-V0 or DP-V1. This information is provided by the Communication Channel (based on the ScanRequest)
ProtocolIdentificationProfile	Indicates the identification type for the device ("DP", "PA", "IM-PA" or "IM").
ScannedPhysicalLayer	Information about the physical layer that was scanned. This information is provided by the Communication Channel (based on knowledge of the fieldbus)
ManufacturerId	Manufacturer identification number. Available for PA and I&M only.
DeviceTypeId	The IDENT_NUMBER of the device.
HardwareRevision	The hardware version revision of the device. Available for PA and I&M only.
SoftwareRevision	The software version revision of the device. Available for PA and I&M only.
SerialNumber	The serial number of the specific device. Available for PA and I&M only.
Tag	Identifying tag for a device. Available for PA and I&M only.
ProtocolSpecificProperties	
ProtocolInformation	Profile specific information (provided by derived datatypes). The information of this member is mapped into DeviceScanInfo.ProtocolSpecificProperties.
DeviceSpecificProperties	
ManufacturerSpecificExtension	Can be used by DTM for vendor specific device identification information, e.g. by combining a number of device parameter values into one string value. This can be used to identify a specific device variant.

11.2.2 Datatypes derived from ProfibusBaseScanInfo

This is the profile specific implementation of the abstract class ProfibusBaseScanInfo (see Figure 33).



Used in:

IDtmScanning.EndScanRequest()

Figure 33 – Datatypes derived from ProfibusBaseScanInfo