FDT3.0 for HART®
Protocol Annex

Technical Specification

Version 1.00.00

FDT Group – Document No. <20-0003>

Publisher:

FDT Group AISBL
5 Industrieweg • 3001 Heverlee • Belgium

www.fdtgroup.org
info@fdtgroup.org

History

Rev.	Author	Change Description	Date
1.00.00	PG HART	Release version	2020-06-04

CONTENTS

Table of Figures

Table of Tables

1 Scope

1.1 General

This document provides information for integrating the HART® protocol into the Field Device Tool (FDT) specification version 3.0 [1].

This document neither contains the FDT3.0 specification nor modifies it. The FDT3.0 specification is available from the homepage of the FDT Group (www.fdtgroup.org).

This document also neither contains the complete specification for HART® protocol nor modifies them.

1.2 Intended audience

The intended audiences of this document are persons who are going to develop FDT3.0 products for HART.

2 Normative references

The following referenced documents are indispensable for the application 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 61784 (all parts), *Industrial communication networks – Profiles*

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

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

IEC 62453-41: *Field Device Tool (FDT) interface specification – Part 41: Object model integration profile – Common object model*

HART 7.6: *HART Communication Protocol Specification, FieldComm Group, June 2016*

FDT3.0, Technical Specification, Version 1.00, FDT Group, AISBL; March 2020

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 FDT3.0, IEC 62453-1, IEC 62453-2, IEC 62453-41, MSDN® and the following apply.

3.2 Abbreviations

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

API	Application Programming Interface
BTM	Block Type Manager
C8PSK	*Coherent 8-way Phase Shift Keying HART communication layer* as defined in HCF_SPEC-60, Revision 1.0
EDD	Electronic Device Description
DLL	Dynamic Link Library
DTM	Device Type Manager
FA	Frame Application
FDT	Field Device Tool

FSK	*Frequency Shift Keying HART communication layer* as defined in HCF_SPEC-54, Revision 8.1
GUI	Graphical User Interface
GUID	Globally Unique Identifier (a UUID)
HART	Highway Addressable Remote Transducer
IID	Interface ID
LCID	Locale ID
MSDN®	Microsoft Developer Network
WPF	Windows Presentation Foundation
XSL	Extensible Stylesheet Language
XSLT	XSL Transformations
UML	Unified Modeling Language

3.3 Conventions

3.3.1 Naming and referencing of datatypes

The conventions for naming and referencing of data types are explained in IEC 62453-2 clause A.1

3.3.2 Vocabulary for requirements

The following expressions are used when specifying requirements.

Wording	Indicates
“shall” or “Mandatory”	No exceptions allowed.
“should” or “recommended”	Strong recommendation. It may make sense in special exceptional cases to differ from the described behavior.
“can” or “optional”	Function or behavior may be provided, depending on defined conditions.

3.3.3 Use of UML

Figures in this document are using UML notation as defined in Annex I of the FDT3.0 specification.

4 Bus category

The type of HART protocol is identified by unique identifier in ProtocolId property within data type BusCategory as defined in Table 1.

Table 1 – Definition of BusCategory

ProtocolId value (ProtocolId)	**Buscategory Name**	**Display Name** (ProtocolName)	**Description**
036D1498-387B-11D4-86E1-00E0987270B9	HART_Basic	‘HART’	Support of HART protocol over FSK communication with basic functionality (includes earlier FDT versions)
98503B8F-0FFB-4EB7-BB67-F4D6BD16DB8D	HART_FSK	‘HART FSK’	Support of HART protocol over FSK communication with complete functionality
74D29D22-F752-40EF-A747-ACA72C791155	HART_Wireless	‘HART Wireless’	Support of WirelessHART protocol
58001A08-C178-4A59-A76B-9EF9111CB83D	HART_RS485	‘HART RS485’	Support of HART protocol over RS485 communication

ProtocolId value (ProtocolId)	Buscategory Name	Display Name (ProtocolName)	Description
EF708CB7-A2A1-42AF-890C-15CEB680CC12	HART_Infrared	'HART Infrared'	Support of HART protocol over Infrared communication
D122D172-F0C7-4B03-965B-512CD4C0871E	HART_IP	'HART IP'	Support of HART over IP protocol

The addressing modes depend on the type of the used HART protocol. Also additional addressing information might be necessary for some types of HART protocols. Table 2 shows the dependency of usable addressing modes and additional address information in dependency of the HART protocol in use.

Table 2 – Relation of Buscategories and supported features

Buscategory Name	Supported Addressing Modes	Address Data Type	Exposed Data	Comment
HART_Basic	ShortAddress	HartDevice Address	Not defined	An FDT3.0 DTM shall use this Id in addition to HART_FSK in order to provide compatibility with FDT2 DTMs Only single byte ManufacturerID and DeviceTypeID are supported.
HART_FSK	ShortAddress, ShortTag, LongTag		As described in chapter 5.3	A DTM may use more than one of these Ids if the device supports multiple physical connections e.g. WirelessHART and FSK
HART_RS485				
HART_Infrared				
HART_Wireless		HartDeviceWirelessAddress		
HART_IP		HartDeviceIpAddress		

Table 3 defines which PhysicalLayer can be used together with the BusCategory defined in Table 1.

Table 3 – Definition of PhysicalLayer

PhysicalLayer.Id value	PhysicalLayer name value	Description
BAB2091A-C0A7-4614-B9DE-FCC2709DCF5D	HART FSK Physical Layer	Support of HART FSK physical layer
B9F1A250-AC94-4487-8F25-A8F3F8F89DC5	WirelessHART Physical Layer	Support of WirelessHART physical layer
036D1591-387B-11D4-86E1-00E0987270B9	HART RS-485 Physical Layer	Support of HART devices using RS-485 communication
AE4119EF-B9FD-429c-B244-134DB182296A	HART Infrared Physical Layer	Support of HART devices using infrared communication
307dd808-c010-11db-90e7-0002b3ecdcbe	10BASET	HART Ethernet based Physical Layers
307dd808-c010-11db-90e7-0002b3ecdcbe	10BASET	
307dd809-c010-11db-90e7-0002b3ecdcbe	10BASETXHD	
307dd80a-c010-11db-90e7-0002b3ecdcbe	10BASETXFD	
307dd80b-c010-11db-90e7-0002b3ecdcbe	10BASEFLHD	
307dd80c-c010-11db-90e7-0002b3ecdcbe	10BASEFLFD	
307dd80d-c010-11db-90e7-0002b3ecdcbe	10BASEFXHD	
307dd80e-c010-11db-90e7-0002b3ecdcbe	10BASEFXFD	
307dd80f-c010-11db-90e7-0002b3ecdcbe	100BASETXHD	
307dd810-c010-11db-90e7-0002b3ecdcbe	100BASETXFD	

PhysicalLayer.Id value	PhysicalLayer name value	Description
307dd811-c010-11db-90e7-0002b3ecdcbe	100BASEFXHD	
307dd812-c010-11db-90e7-0002b3ecdcbe	100BASEFXFD	
307dd813-c010-11db-90e7-0002b3ecdcbe	100BASELX10	
307dd814-c010-11db-90e7-0002b3ecdcbe	100BASEPX10	
307dd815-c010-11db-90e7-0002b3ecdcbe	1000BASEXHD	
307dd816-c010-11db-90e7-0002b3ecdcbe	1000BASEXFD	
307dd817-c010-11db-90e7-0002b3ecdcbe	1000BASELXHD	
307dd818-c010-11db-90e7-0002b3ecdcbe	1000BASELXFD	
307dd819-c010-11db-90e7-0002b3ecdcbe	1000BASESXHD	
307dd81a-c010-11db-90e7-0002b3ecdcbe	1000BASESXFD	
307dd81b-c010-11db-90e7-0002b3ecdcbe	1000BASETHD	
307dd81c-c010-11db-90e7-0002b3ecdcbe	1000BASETFD	
307dd81d-c010-11db-90e7-0002b3ecdcbe	10GigBASEFX	

The significant information for topology planning is the BusCategory. The PhysicalLayer (which is provided in the BusInformation data type) shall be used only for additional information.

The DataLinkLayer property is not applicable for HART and shall be set to null.

5 Access to instance, device and process data

5.1 General

The HART protocol has semantics defined that allow in a wide range the identification of device variables and device parameters. Most of this semantic information is defined in the standard EDD import libraries.

This section describes how the semantic information defined with the HART protocol shall be used to export device data, instance data and process data.

5.2 IO signals provided by DTM

A DTM shall provide IO signal information of the device using the IProcessData interface

A HART device is connected to the process either via its analog channels or via digital information (e.g. burst mode). Analog channels are always related to a dynamic variable, as specified in [8] chapter 8 and therefore the description of an analog channel has to be accessed using the respective dynamic variable (e.g. the attributes of dynamic variable PV always describe the first analog channel).

HART distinguishes between three methods to access digital signals:

1. Access to analog value and assigned dynamic variables (Command #3)

   IO signals can be assigned to one of the four dynamic variables PV, SV, TV, and QV. Using the command #3 the analog value and the dynamic variables can be read without specific device knowledge.

2. Indexed access to device variables (Command #33)

   All device variable values and their units can be read using the related device variable code information in command #33.

3. Indexed access to device variables classification and status (Command #9)

   Command #9 is an extension of command #33. Beside of the value and unit also a classification and the variable status can be determined.

It is up to the command initiator to determine by means of the HART specification which commands to be used.

To provide all information required to access the output signal information the DTM has to provide the information shown in Table 4 within its HartIOSignalInfo.

Table 4 – Output signal info within IOSignalInfo / HartIOSignalInfo

Attribute	Description
Name	Name of the IO signal
Range	Reference to the variables providing range information
Unit	Reference to an enumeration variable describing the unit information
DeviceVariableAssignment	Constant enumeration value that can take following values - unassigned: The IO signal is not assigned to dynamic variable and can only be accessed using the indexed approach (reading device variables). - PV: IO signal assigned to the PV dynamic variable - SV: IO signal assigned to the SV dynamic variable - TV: IO signal assigned to the TV dynamic variable - QV: IO signal assigned to the QV dynamic variable
DeviceVariableCode	Constant that specifies the device specific variable code. Because of compatibility reasons to earlier FDT versions, this variable could be set to the value 252 that stands for “unknown”.

5.3 Data interfaces

5.3.1 General

Within HART, several command sets are defined. As part of the command set definition, HART provides a precise naming convention that is documented within the sources of the standard EDD libraries. The data provided by the access data interfaces should be named according to the HART naming convention. For backward compatibility, the semantic IDs as defined in previous versions of FDT (e.g. [2]) should also be provided by the DTM.

A DTM shall provide all device data that is related to the set of Universal Commands and should provide all device data related to Common Practice Commands. If the device supports additional command sets, like device family profiles, those data should also be exported using the naming convention as defined in the HART standard EDD include libraries and shown in Figure 1 and Figure 2.

5.3.2 Mapping HART data types to FDT data types

For a better usability of data provided by the data access interfaces IDeviceData and IInstanceData, all data from the device shall be converted into data types that are common to FDT. The mapping of basic data types is defined in Table 5.

Table 5 – Mapping of basic data types

HART Data types		FDT data type	IEC data type
Packed ASCII	(see [8] 5.1.1)	String	STRING
ISO Latin-1	(see [8] 5.1.2)	String	STRING
Dates	(see [8] 5.2)	DateTime	DateAndTime
Time	(see [8] 5.3)	ULong *(1/32 ms since midnight)*	ULINT *(1/32 ms since midnight)*
Single Precision Floating Point	(see [8] 5.4)	float	REAL
Double Precision Floating Point	(see [8] 5.4)	double	LREAL
1-4 Byte Unsigned Integer	(see [8] 5.5)	UInt	UDINT
5-8 Byte Unsigned Integer	(see [8] 5.5)	ULong	ULINT
1-4 Byte Signed Integer	(see [8] 5.6)	Int	DINT
5-8 Byte Signed Integer	(see [8] 5.6)	Long	LINT
1-4 Byte Enumerated	(see [8] 5.7.1)	UInt	UDINT

HART Data types		FDT data type	IEC data type
5-8 Byte Enumerated	(see [8] 5. 7.1)	ULong	ULINT
1-4 Byte Bit Fields	(see [8] 5.7.2)	UInt	UDINT
5-8 Byte Bit Fields	(see [8] 5.7.2)	ULong	ULINT

The access to device specific data has been standardized using structures with the standard EDD import libraries for HART. The structure uses ARRAY and COLLECTION constructs that shall be reused when exposing data within FDT in IDeviceData and IInstanceData interfaces with elements of type StructDataGroup.

When converting a COLLECTION into a StructDataGroup, the DataItems shall be identical to the COLLECTION members with:

DataItem Name = COLLECTION member identifier
DataItem Label = COLLECTION member label

When Converting an ARRAY into a StructDataGroup, the DataItem shall be identical to the ARRAY element with:

DataItem Name = ARRAY element index as string
DataItem Label = ARRAY element label

An example for such a structure is presented in Figure 1 and Figure 2.

5.3.3 SemanticInfo

The SemanticInfo for HART protocol related parameters is directly related to the protocol specification. The definition of the HART commands is the base for the parameter address information which shall be used in the properties ParameterReadAddress, ParameterWriteAddress and SemanticId of the SemanticInfo data type.

The syntax of the parameter address information is as follows:

CMD<x>[Q(<r>)]B<y>B<z>L<n>

The [Q(<r>)] portion only is required to define request data.

For description of the attributes, please view Table 6.

Table 6 – SemanticInfo attributes description

Attribute	Data Type	Description
<x>	decimal integer	command number in the range of 0 to 255
<r>	hex-string	request data bytes
<y>	decimal integer	index of the start byte in the response data section (start index = 0)
<z>	decimal integer	index of the start bit in the byte referenced by <y>
<n>	decimal integer	length of the value in bits

According to chapter 6.3 commands with extended command number have to be described using “CMD31Q<r>…” wherein the first two byte of the request data section have to contain the extended command number.

The property SemanticInfo.ApplicationDomain shall contain ‘FDT_HART’ for all parameters in Table 7 and Table 8.

5.3.4 Data exposure using IDeviceData and IInstanceData Interfaces

5.3.4.1 Export of basic device parameters

Using the HART data type mapping rules introduced in the preceding sections basic device parameters defined within the Universal and Common Practice Command sets can be exported.

Basic parameters that are not accessed using index information in the request part should for compatibility reasons provide additionally the SemanticInfo information specified in previous versions of FDT. In FDT3.0 variables are identified by their standard identifier in HART. In Table 7 all variables are listed that shall be exported in the related data interfaces when supported by the device. HART identifier shall be assigned to the property Data.Id. SemanticId shall be assigned to the properties SemanticInfo.SemnaticId, SemanticInfo.ReadParameterAddress and SemanticInfo.WriteParameterAddress.

Variables with identifiers starting with “PV.”, ”SV.”, ”TV.” and ”QV.” have to be seen as short cuts to variables that shall be available too using the structured exposure of device variables as defined in chapter 5.3.4.2.1

Table 7 – Basic Variables exported in IDeviceData and IInstanceData interfaces

Identifier	SemanticId	Description
device_type	CMD0B1B0L16	Expanded Device Type (see HCF Common Table 1, Device Type Codes and the Command Summary Specification, Section 6). Note: The information from CMD0 is also available via IHardwareIdentification interface.
request_preambles	CMD0B3B0L8	Minimum number of Preambles required for the request message from the Master to the Slave. This number includes the two preambles used in asynchronous Physical Layers (along with the Delimiter) to detect the start of message.
universal_revision	CMD0B4B0L8	HART Protocol Major Revision Number implemented by this device. For HART Revision 7, this value must be the number 7.
transmitter_revision	CMD0B5B0L8	Device Revision Level (refer to the HCF Command Summary Specification)
software_revision	CMD0B6B0L8	Software Revision Level of this device. Levels 254 and 255 are reserved.
hardware_revision	CMD0B7B3L5	(Most Significant 5 Bits) Hardware Revision Level of the electronics in this particular device. Does Not Necessarily Trace Individual Component Changes. Level 31 is Reserved.
physical_signaling_code	CMD0B7B0L3	(Least Significant 3 Bits) Physical Signaling Code (see HCF Common Table 10, Physical Signaling Codes)
device_flags	CMD0B8B0L8	Flags (see HCF Common Table 11, Flag Assignments)
device_id	CMD0B9B0L24	Device ID. This number must be different for every device manufactured with a given Device Type.
response_preambles	CMD0B12B0L8	Minimum number of preambles to be sent with the response message from the slave to the master.
max_num_device_variables	CMD0B13B0L8	Maximum Number of Device Variables. This indicates the last Device Variable code that a host application should expect to be found in the field device (e.g., when identifying the Device Variables using Command #54).
config_change_counter	CMD0B14B0L16	Configuration Change Counter
extended_fld_device_status	CMD0B16B0L8	Extended Field Device Status (refer to HCF Common Table 17, Extended Field Device Status)
manufacturer_id	CMD0B17B0L16	Manufacturer Identification Code (see HCF Common Table 8, Manufacturer Identification Codes)
private_label_distributor	CMD0B19B0L16	Private Label Distributor Code (see HCF Common Table 8, Manufacturer Identification Codes)
device_profile	CMD0B21B0L8	Device Profile (see HCF Common Table 57)
polling_address	CMD7B0B0L8	Polling Address of Device (refer to the Data Link Layer Specification)

Identifier	SemanticId	Description
loop_current_mode	CMD7B1B0L8	Loop Current Mode (refer to HCF Common Table 16, Loop Current Modes)
message	CMD12B0B0L192	Message
tag	CMD13B0B0L48	Tag
descriptor	CMD13B6B0L96	Descriptor
date	CMD13B18B0L24	Date Code
PV.SENSOR_SERIAL_NUMBER	CMD14B0B0L24	Transducer Serial Number
PV.DIGITAL_UNITS	CMD14B3B0L8	Transducer Limits and Minimum Span Units Code (refer to HCF Common Tables Specification)
PV.UPPER_SENSOR_LIMIT	CMD14B4B0L32	Upper Transducer Limit
PV.LOWER_SENSOR_LIMIT	CMD14B8B0L32	Lower Transducer Limit
PV.MINIMUM_SPAN	CMD14B12B0L32	Minimum Span
PV.ALARM_CODE	CMD15B0B0L8	PV Alarm Selection Code (see HCF Common Table 6, Alarm Selection Codes). The Alarm Selection Code indicates the action taken by the device under error conditions. For transmitters, the code indicates the action taken by the Loop Current. For Actuators, the action taken by the positioner is indicated.
PV.TRANSFER_FUNCTION	CMD15B1B0L8	PV Transfer Function Code (see HCF Common Table 3, Transfer Function Codes). The Transfer Function Code must return "0", Linear, if transfer functions are not supported by the device.
PV.RANGE_UNITS	CMD15B2B0L8	PV Upper and Lower Range Values Units Code (refer to HCF Common Tables Specification)
PV.UPPER_RANGE_VALUE	CMD15B3B0L32	PV Upper Range Value
PV.LOWER_RANGE_VALUE	CMD15B7B0L32	PV Lower Range Value
PV.DAMPING_VALUE	CMD15B11B0L32	PV Damping Value (units of seconds)
write_protect	CMD15B15B0L8	Write Protect Code (see HCF Common Table 7, Write Protect Codes). The Write Protect Code must return "251", None, when write protect is not implemented by a device.
PV.ANALOG_CHANNEL_FLAGS	CMD15B17B0L8	PV Analog Channel Flags (see HCF Common Table 26, Analog Channel Flags)
final_assembly_number	CMD16B0B0L24	Final Assembly Number
longTag	CMD20B0B0L256	Long Tag
PV.DIGITAL_UNITS	CMD1B0B0L8	Primary Variable Units (refer to HCF Common Tables Specification)
SV.DIGITAL_UNITS	CMD3B9B0L8	Secondary Variable Units Code (refer to HCF Common Tables Specification)
TV.DIGITAL_UNITS	CMD3B14B0L8	Tertiary Variable Units Code (refer to HCF Common Tables Specification)
QV.DIGITAL_UNITS	CMD3B19B0L8	Quaternary Variable Units Code (refer to HCF Common Tables Specification)
PV.CLASSIFICATION	CMD8B0B0L8	Primary Variable Classification (see HCF Common Table 21, Device Variable Classification Codes)
SV.CLASSIFICATION	CMD8B1B0L8	Secondary Variable Classification (see HCF Common Table 21, Device Variable Classification Codes)
TV.CLASSIFICATION	CMD8B2B0L8	Tertiary Variable Classification (see HCF Common Table 21, Device Variable Classification Codes)
QV.CLASSIFICATION	CMD8B3B0L8	Quaternary Variable Classification (see HCF Common Table 21, Device Variable Classification Codes)

Identifier	SemanticId	Description
lock_device_status_code	CMD76B0B0L8	Lock Status (see HCF Common Table 25, Lock Device Status)
last_clock_date,	CMD90B7B0L24	Date clock last set
last_clock_time	CMD90B10B0L32	Time clock last set
real_time_clock_flag	CMD90B14B0L8	RTC Flags (see HCF Common Table 42)

Table 7 contains data that can also be read via other interfaces. In this case the DTM shall take care about the consistency of the data. E.g. the variables which are related to CMD0 shall be identical to the HartDeviceScanInfo data, which can be read via the IHardwareScan interface.

Table 8 list all basic device variables that shall be exported only within IDeviceData interface if supported by the device.

Table 8 – Basic Variables exported only in IDeviceData interface

Identifier	SemanticId	Description
device_status	-/- *(empty)*	Device status information transferred with each reply. Due to the fact that device_status is accessible by standard means in each transaction, the semantic info is empty
PV.DIGITAL_VALUE	CMD1B1B0L32	Primary Variable
PV.ANALOG_VALUE	CMD2B0B0L32	Primary Variable Loop Current (units of milli amperes)
PV.PERCENT_RANGE	CMD2B4B0L32	Primary Variable Percent of Range (units of percent)
PV.DIGITAL_VALUE	CMD3B5B0L32	Primary Variable
SV.DIGITAL_VALUE	CMD3B10B0L32	Secondary Variable
TV.DIGITAL_VALUE	CMD3B15B0L32	Tertiary Variable
QV.DIGITAL_VALUE	CMD3B20B0L32	Quaternary Variable
current_date	CMD90B0B0L24	Current Date
current_time	CMD90B3B0L32	Current Time of Day

5.3.4.2 Export of Structural Information

5.3.4.2.1 Device Variables

In addition to the measurement values provided by dynamic variables, there exists information in the device that is related to the dynamic variable (e.g. range information, linearization functions). In HART several categories of such information are defined, and standardized means are available to document such relationships without device specific knowledge. This information shall be exported in the IDeviceData and IInstanceData interfaces using StructDataGroup elements as mentioned in section 5.3.2.

The DTM shall expose information in the structure as shown in Figure 1 and Figure 2.

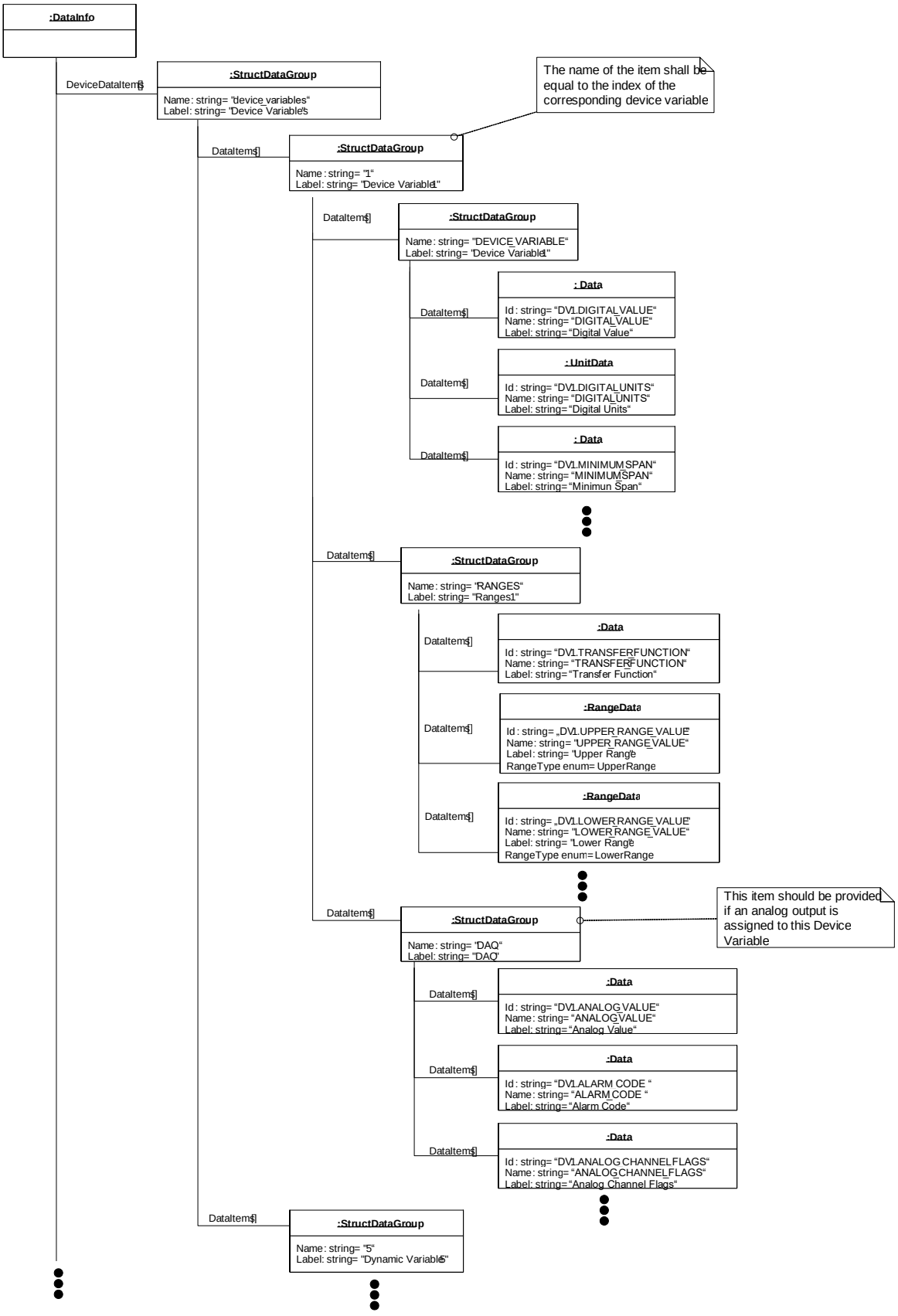


Figure 1 – Structural information for device variables

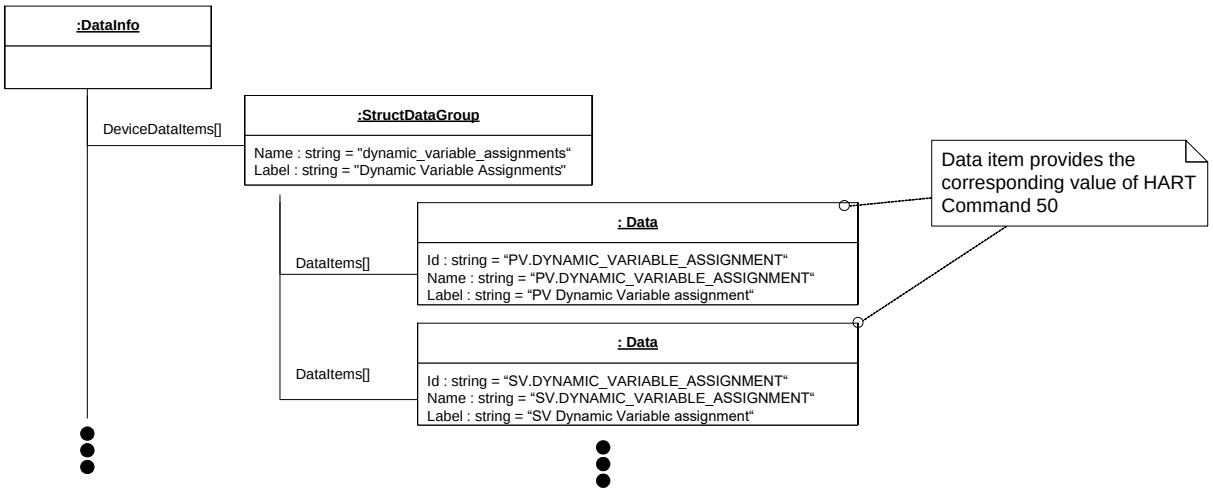


Figure 2 – Structural information for dynamic variables

Future changes to the structure shall be reflected similarly.

5.3.4.2.2 Device specific status

HART supports two types of status information.

device status	This status is communicated with the response of each command and contains some flags that describe the state of the device. This status is already exposed in IDeviceData with variable "device_status".
additional device status	This status information contains device specific information. Usually it is presented in HART command #48 when the "additional status available" flag is set in the device_status variable.

To expose additional device status information via the IDeviceData interface a new identifier is introduced here. The data is exposed with DataGroup "additional_device_status". Elements of this data group are references to variables that contain device specific status information like shown in Figure 3.

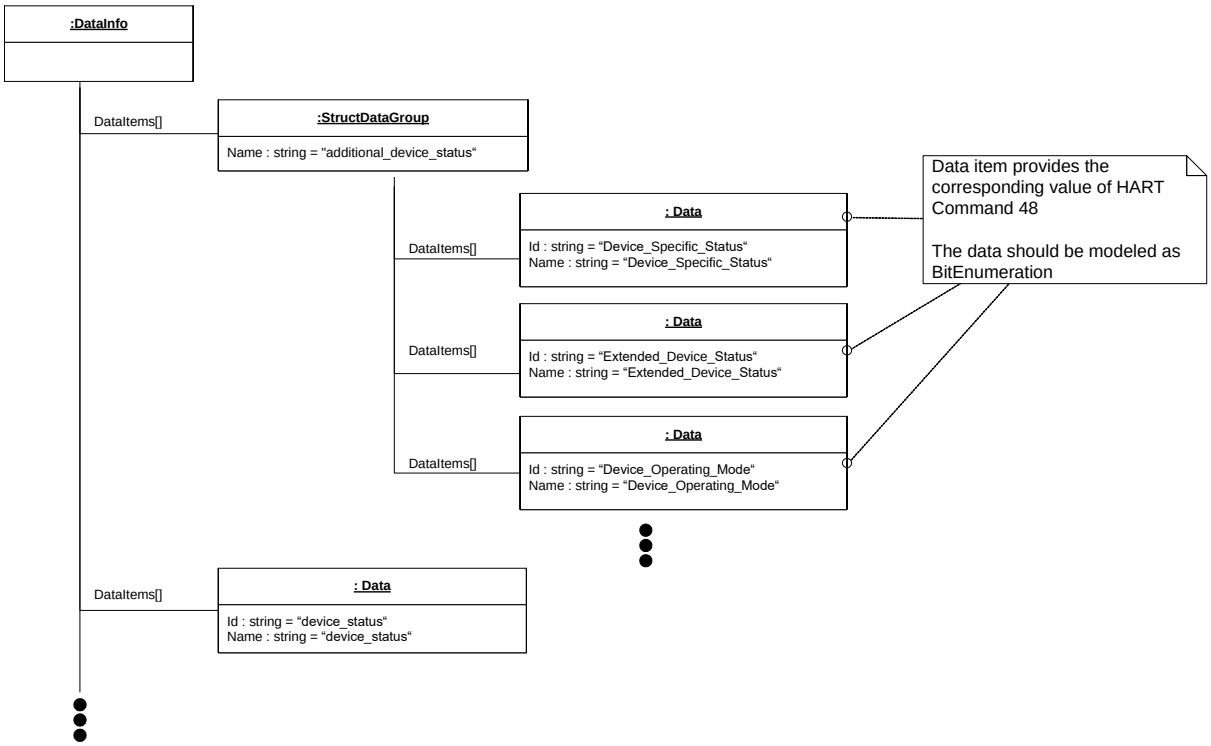


Figure 3 – Structural information for extended device status

SemanticInfo information shall be provided for all variables referenced in this group, allowing external tools to request status information without instantiation of the DTM.

5.3.4.2.3 Extended Burst and Event information

HART 7 introduced new features in burst information including event notification. Data Propagation using these new burst and event management can be configured with standard means as defined by HART. The content of burst messages (especially for sub devices) can be evaluated using the exported structural information as presented in the preceding sections.

Support of this new handling is not defined in this document. It is open to the DTM developer to export additional information with the means introduced above.

6 Protocol specific behavior

6.1 Device addressing

HART is a master/slave protocol that is not connection based. Transactions are always communicated using a device unique address information, a 5 byte long integer, the so called long address.

Device addressing in HART therefore is mainly to determine this long address.

There are currently three ways possible to determine the long address.

1. short address

   The short address is a number between 0 and 63 (for HART version 5 only 0 to 15). In the context of a direct connection to the device the short address is unique and allows to read the long address using command #0.

2. short tag

   With command #11 the long address information can be requested for a device with a specific tag. Such requests are especially used for installations with a huge amount of connected HART devices. All HART multiplexer devices and other HART communication structures have to support this command.

3. long tag

   HART Version 6 introduced the long tag, which at 32 characters has 24 more characters than the short tag, and can thus provide for a larger number of device label options. For devices with HART version less than 6 instead of long tag, message is used. With command #21 a similar method to determine the long address is possible. Command #21 is usually supported by highly modular devices or gateways.

A Device DTM is responsible to provide and store all information that is used for resolving the long address of a connected device. It has therefore to maintain the data for all three address resolving methods. The DTM responsible to connect to the communication hardware has to select the method and provide means for a user to input the address information.

6.2 Support of scanning

All Communication Channels supporting a HART protocol shall support the IScanning interface.

6.3 Extended Command Numbers

The HART command number is defined as a one byte unsigned integer. Beginning with HART Version 6, an extended command number format, using two bytes instead of one, was defined to allow for more than 255 command numbers. This format uses command number 31, which was previously reserved, to indicate that the request is using the extended command number format.

According to the specification in [10] section 8.1.2, extended command numbers are implemented with command #31 by using the extended command number as first two bytes in the request and response section.

In FDT all commands with extended command numbers shall be implemented using command #31.

6.4 Burst mode

A subscription of device initiated data transfer can be requested by calling ISubscription::<SubscriptionInitialization()>. The Communication Channel may detect if the device is already in Burst Mode. If HART Burst frames are received, the device is in Burst mode and BurstModeDetected property of HartSubscribeResponse is set to TRUE. This means that Device DTM will start to receive Burst messages via SubscriptionCallback mechanism. In case that no burst messages were received, BurstModeDetected property is set to FALSE. It is up to the Device DTM to set the device into Burst mode. Then the Device DTM may call ISubscription::<SubscriptionInitialization()> again in order to receive Burst messages.

In order to unsubscribe, the Device DTM calls ISubscription::<SubscriptionTermination()>. Device DTM will not receive any more Burst information via SubscriptionCallback mechanism. The Communication Channel does not switch off the Burst Mode in the device. The Device DTM may switch Burst Mode on or off by using normal TransactionRequests (HART command #109). This is independent of the subscription.

The sequence is described in Figure 4.

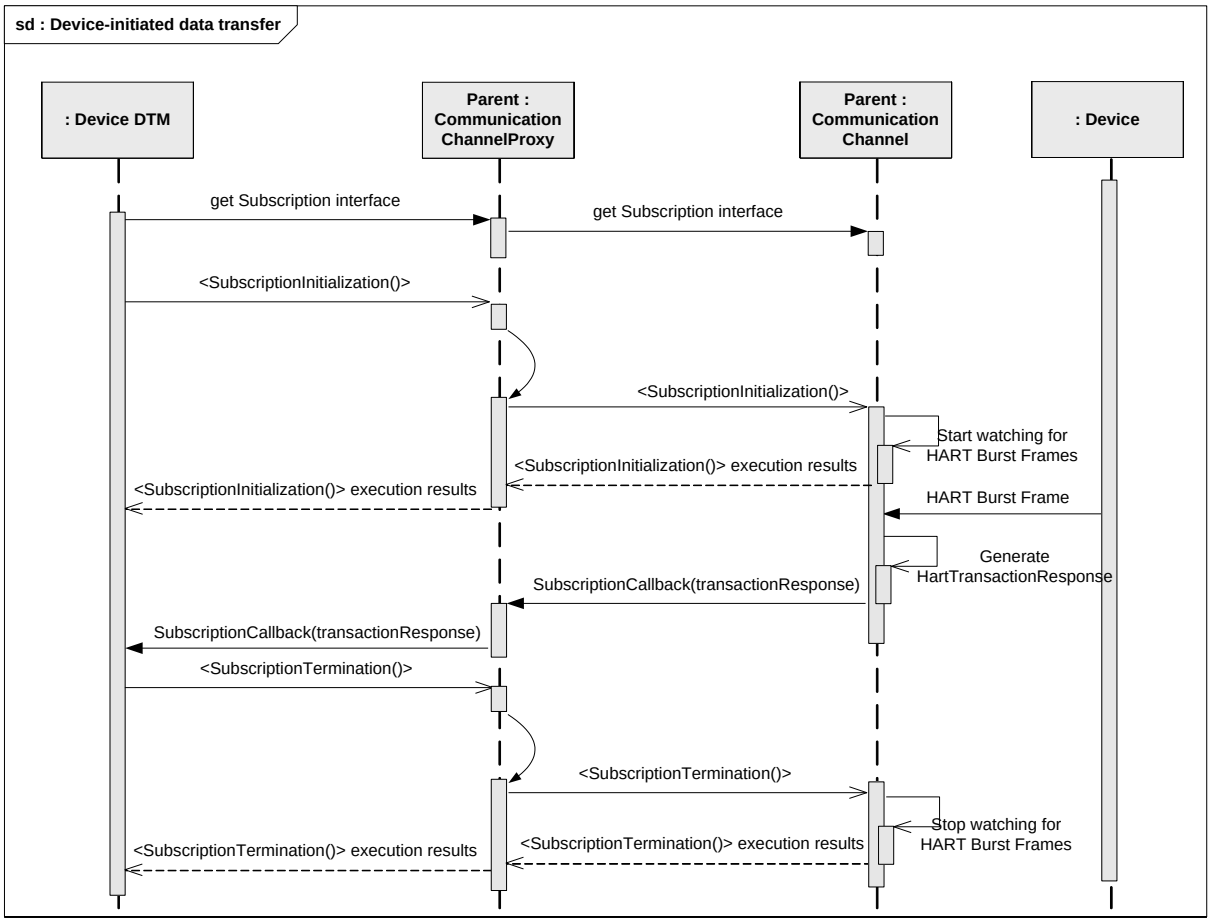


Figure 4 – Device-initiated data transfer with Burst Mode

6.5 Handling of communication failures and timeouts

HART is not a connection based protocol (especially when using physical layers FSK and C8PSK). The HART concept uses therefore a device specific handling of communication errors. The protocol defines a section in the response frame that can carry communication failure information.

In case a communication error (this also includes timeouts) occurs on the HART physical layers, no Abort message shall be sent to the Device DTM, but the transaction request shall be responded with a set of data that describes the communication error as defined in HART [8].

The Device DTM has in case of communication failures the responsibility to perform the error handling to recover from the communication failure.

Only in case of a connection-based communication break (e.g. Ethernet connection to a HART modem), the Communication DTM or Gateway DTM shall send an Abort signal to the Device DTM.

6.6 Buscategory HART_Basic

Buscategory HART_Basic is defined to allow serialization-compatibility with existing FDT2 DTMs (see [2]). An FDT2 DTM may use this buscategory in addition to the buscategories HART_FSK, HART_Wireless, HART_RS485, HART_Infrared and HART_IP (see [2]). An FDT3.0 DTM is not allowed to use buscategory HART_Basic.

6.7 HART communication structures with multiple gateways

With HART devices structures are possible that allow having multiple gateways in a communication chain. Typical examples for such a topology are wired HART devices connected to a wireless adapter communicating to a wireless gateway (see 6.9).

General concept of nested communication is that a device receives the command data that was generated by its DTM instance and that the DTM instance receives the response data of its device. Also required in nested communication is that the child DTM always is the active sender and therefore is not allowed to pass through communication send by its child DTM without encapsulation or transformation.

With command #77 HART defines a standard encapsulation technology to propagate communication through a network topology with means of nested communication. Each command request that was send by a sub device has to be encapsulated in a command #77 before sending to the parent device. When a response to a command #77 is returned the Gateway DTM shall unpack this command and send its contained command response data to the respective child DTM.

Depending on the implementation in the gateway, a command #77 might be restructured to another command structure. In this case the Gateway DTM has the responsibility to transform incoming command #77 requests from the child DTM to the gateway specific commands and also to restructure the resulting responses back again respectively to responses on the originally received command #77 request.

6.8 Handling of Delayed Responses

HART defines strict time constraints for responses to a request within a HART transaction. In case a device is unable to fulfill the time constraints it can initiate a Delayed Response (DR) sequence. The DR handling is well defined within the HART specification and does not need further introduction here, but some rules have to be defined that allow the DR handling within FDT's nested communication.

The two partners of a DR sequence are usually the host (FDT System) and the device. In such a case the responsibility to handle the DR responses from the device is located at the DTM that represents the device. The Network topology has to ensure that the DR responses are transferred correctly to the DTM that has to handle the DR. An example for such a delayed response handling is shown in Figure 5.

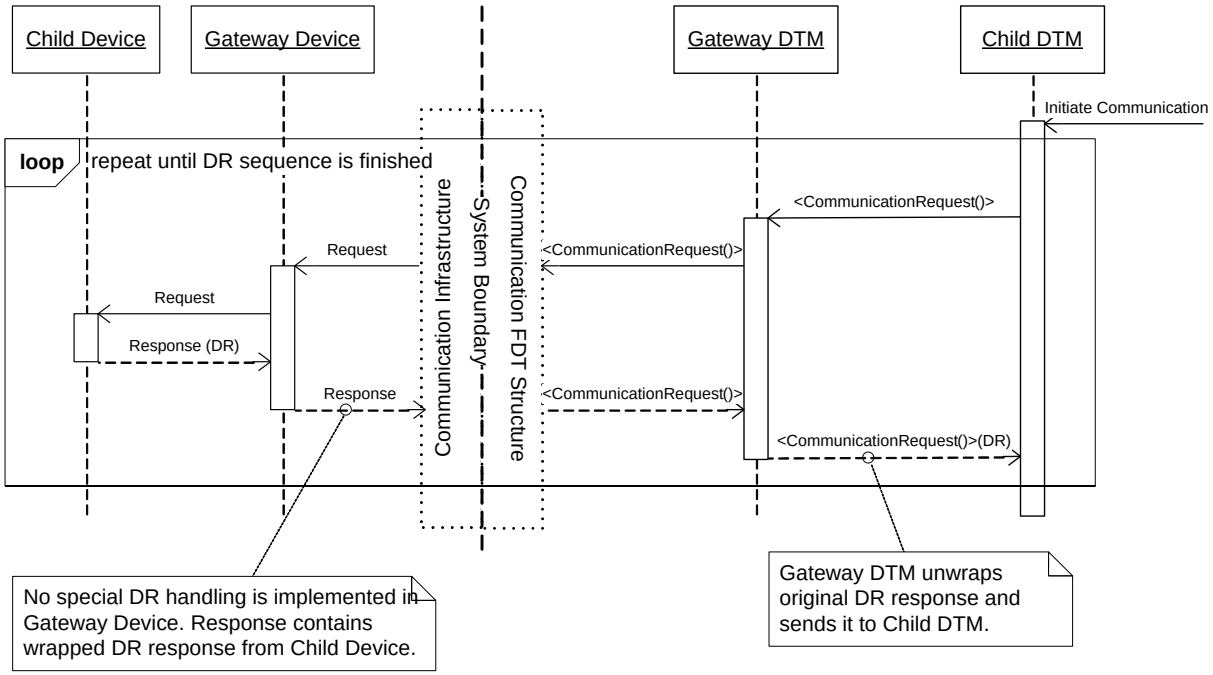


Figure 5 – Handling of Delayed Reponses (scenario 1)

It is also possible that the two partners of a DR sequence are both devices. For example, a gateway device (e.g. WirelessHART Gateway) might execute a delayed response sequence with a child device (e.g. WirelessHART Adapter). In this case the gateway takes over the responsibility to handle the DR responses of the child. The delayed responses will not reach the DTM that represents the child device, but because of the fact that the Gateway is unable to

respond directly, the gateway itself could answer with DR responses that in this case would have to be handled by the DTM representing the gateway. Usually the nested communication concept reflects the interaction between the devices. In the case described here this is not possible and the implementation has to follow the sequence shown in Figure 6.

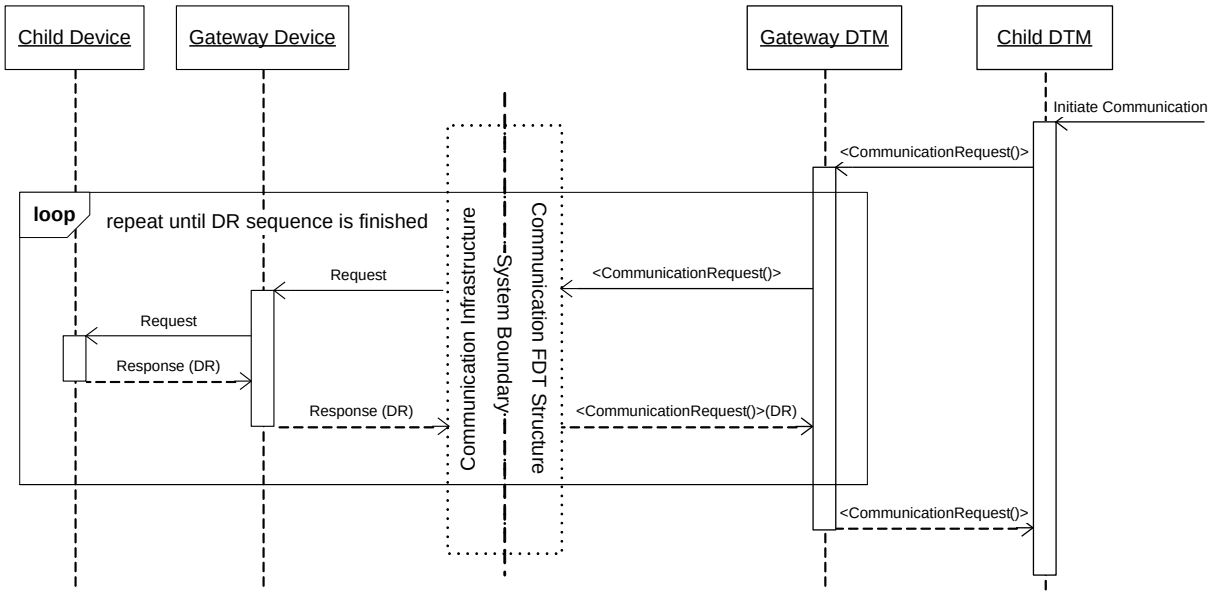


Figure 6 – Handling of Delayed Reponses (scenario 2)

A DR sequence might take a long time that might disturb the usage of the FDT Frame Application and that might block user interaction. There is no timeout time definition existing for DR sequences and neither the DTM itself nor any other DTM in the nested communication chain is capable to initiate a timeout that could recover the system. Timeout time in such a case is application dependent and shall be configurable by the user. When a DR sequence lasts an unreasonable amount of time it shall be an aim to involve the user. In user interface free environments configurable timeout mechanisms shall be implemented.

To handle DR responses with a reliable interoperability, the following rules have to be fulfilled:

- The DTM of a device that might send DR responses has to handle the DR responses of the device.
- DR responses that are not handled by other devices have to be propagated to the DTM that represents the device that send the responses.
- A DTM shall be aware that it will not receive DR responses from the device, when the DR responses are handled by the parent device.
- A DTM that handles DR responses has to implement a user configurable timeout management that must allow the user to set a timeout.

6.9 Communication- and network structures in WirelessHART

6.9.1 Introduction

WirelessHART defines a rich and secure protocol between devices using 2.4GHz wireless technology. Host systems can access wireless devices using a wireless gateway and are not intended to interact with the WirelessHART network directly. Using a WirelessHART gateway a host system can communicate with any WirelessHART device using HART master/slave transactions, without requiring specific knowledge of the WirelessHART protocol.

HART specifies three standard types of WirelessHART network devices:

1. WirelessHART Gateway:

   This device connects a WirelessHART network to the outer world via HART or other protocols that allow data transfer with high baud rates. It is possible to have more than one WirelessHART Gateway active in a WirelessHART Network. The WirelessHART Gateways are responsible to manage the network directory and propagate information from and to the WirelessHART devices.

   For better readability in section 6.9 a WirelessHART Gateway is simply named Gateway.

2. WirelessHART Field Device:

   The WirelessHART Field Device is a device that can participate in a WirelessHART network.

   For better readability in section 6.9 a WirelessHART Field Device is simply named Field Device.

3. WirelessHART Adapter:

   The WirelessHART Adapter is a specialized WirelessHART Field Device that allows the connection of HART FSK and/or 4-20mA sub-devices to the Wireless network.

   For better readability in section 6.9 a WirelessHART Adapter is simply named Adapter and devices connected to an adapter are simply called Sub-Devices.

This section will focus on specialties of WirelessHART and define implementation rules within FDT that are required for interoperable nested communication.

6.9.2 Network topology

Adapters are special devices that connect other HART physical layers (usually HART FSK) with the WirelessHART network as shown in Figure 7.

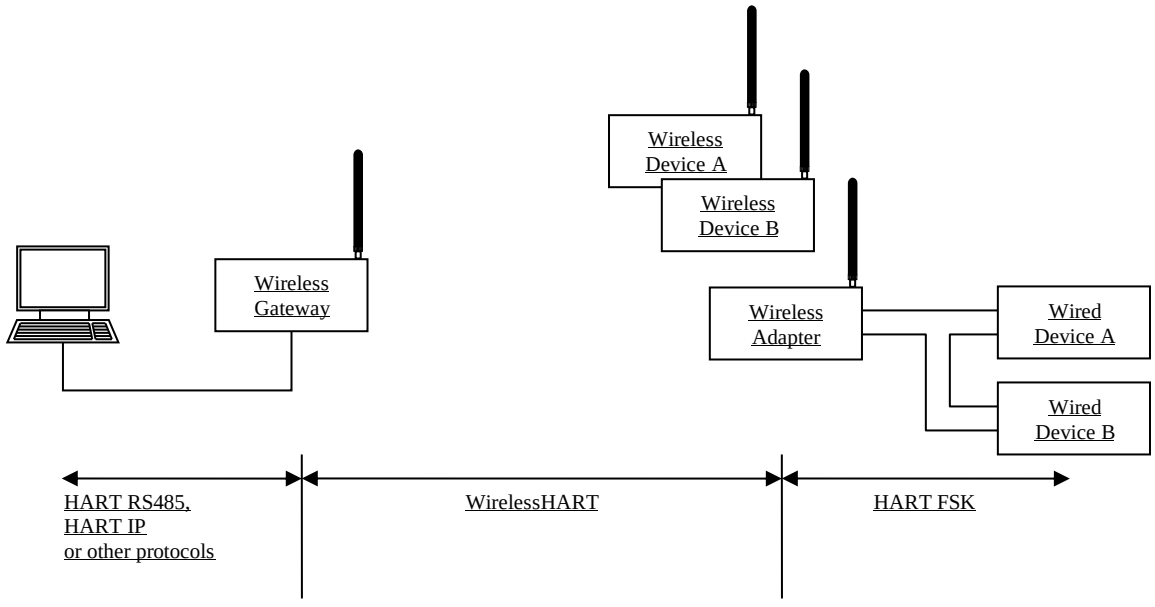


Figure 7 – FDT Frame Application connected to a WirelessHART Gateway

From the perspective of nested communication in FDT the Gateway and the Adapter are both gateway devices that shall be presented as such in the network topology of an FDT Frame Application. The resulting FDT topology is shown in Figure 8.

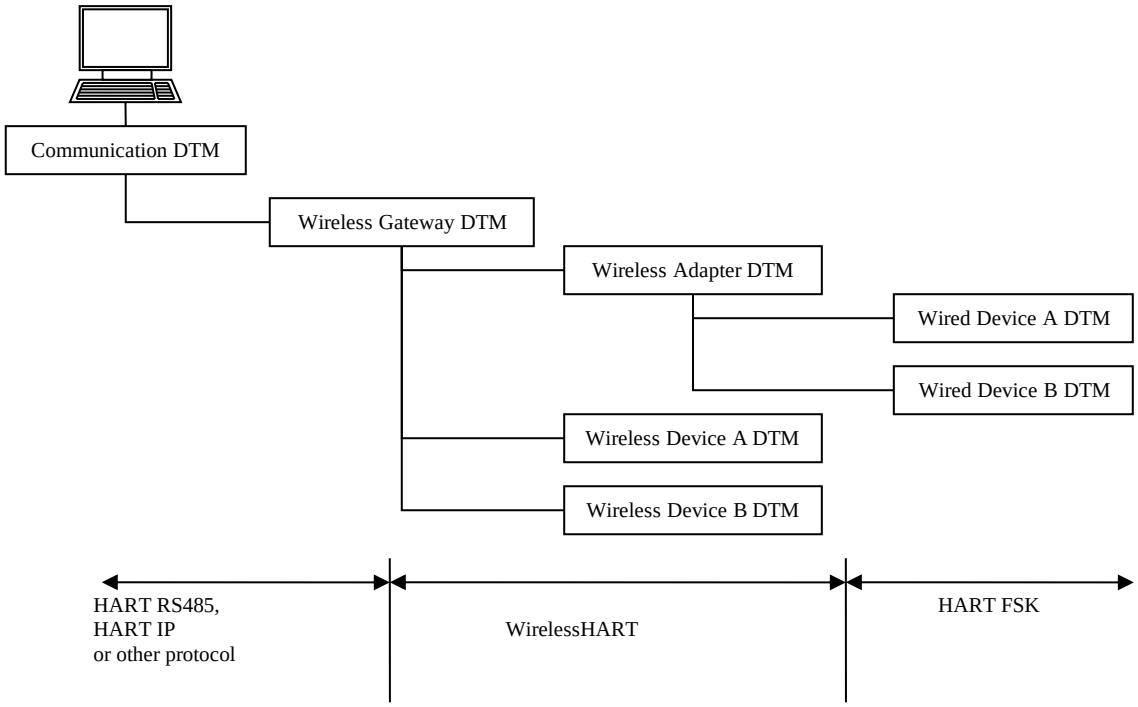


Figure 8 – FDT Topology of a WirelessHART Network

An Adapter interacts with the HART FSK loop like a common HART device. It is acting as a HART Master but can also be addressed with HART transactions from another Master. Especially in service use cases an FDT Frame Application might be connected to the HART FSK loop to directly access the Adapter. In this case the Adapter is connected to the FDT Frame Application as a usual device in a HART FSK multidrop and multimaster scenario like shown in Figure 9.

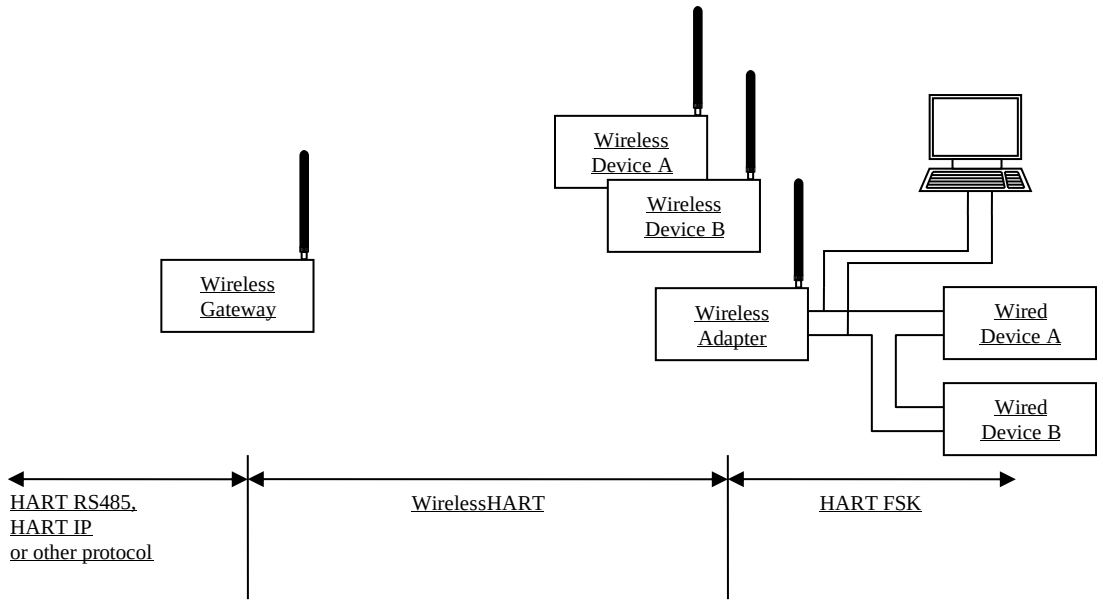


Figure 9 – FDT Frame Application connected to a WirelessHART adapter

In this use case the Network Topology in the FDT Frame shall be structured as shown in Figure 10.

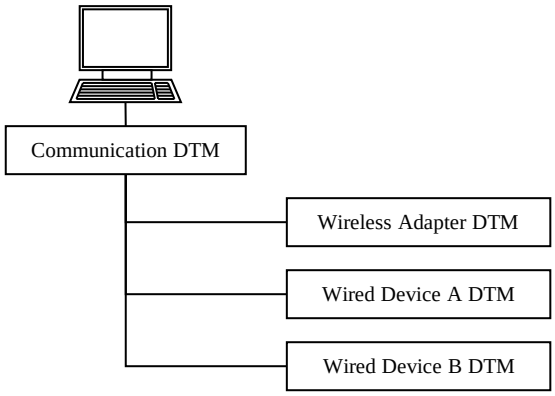


Figure 10 – FDT Topology when directly connected to a WirelessHART Adapter

Figure 8 and Figure 10 show that an Adapter DTM has to implement gateway functions when it is used in a WirelessHART FDT environment and on the other hand has to behave like a simple device, when used in a HART FSK environment. In FDT3.0 a DTM is always informed about changes in communication type. Using the information about the current connection type the Adapter DTM has to implement the respective specific structural behavior.

As a summary of the above section the following rules shall be implemented:

If the Adapter is connected via WirelessHART, it:

- has to interact as a Gateway DTM.
- has to handle communication to the connected Sub-Devices (as specified in section 6.6) that are attached in the topology as child DTMs.
- has to handle DR transactions as described in section 6.8.

If the Adapter is connected via HART FSK, it:

- has to deny attachment of child DTMs.
- has to deny connection to child DTMs.

If an instance of an Adapter DTM is moved from a WirelessHART Communication Channel to a HART FSK Communication Channel, it:

- has to keep all instances of the child DTMs untouched.
- has to allow moving Child DTMs away from its node.

6.10 Transparent Gateways

6.10.1 Introduction

HART supports various types of physical layers like HART FSK, HART over Ethernet, HART over 485, HART over Infrared etc. With this, it is also possible that a HART Network contains a Gateway to connect nodes supporting different physical layers. One example is the Wireless Adapter which allows connecting HART FSK devices in a WirelessHART network.

Another hypothetical example is a HART Infrared Adapter, which can connect a HART FSK device in a HART Infrared Network.

It is possible that some adapters can be transparent, i.e. they do not act as HART Nodes on the network. They just act as physical layer adapter, impersonating the device they are connected to. Though currently there are no Transparent Gateways in the market, HART protocol does not restrict the possibility of such Transparent Gateways/Converters in the network.

FDT3.0, however, does not support such transparent Gateways/Adapters.

With the definition of specific protocol Ids for each of the HART types, it is not possible to create a topology unless DTMs are available for each of the participating nodes in the HART network.

The following scenarios explain the behavior of such Gateways in a FDT Topology.

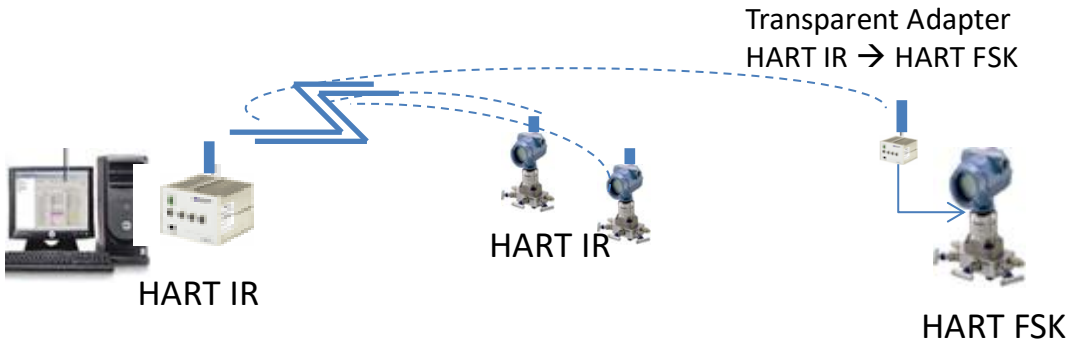


Figure 11 – Scenario with Transparent Adapters

6.10.2 Scenario 1: Manual Topology Creation

During manual topology creation, the HART Infrared Communication DTM only supports HART IR DTMs to be added to the topology. In this case, to add a HART FSK DTM to the Topology, there must be a Gateway DTM (HART Infrared to HART FSK DTM), though, the Adapter is transparent.

It is mandatory in FDT3.0, that every participating node in the physical network shall have a DTM available. Without this, it is not possible to create a valid topology.

6.10.3 Scenario 2: Topology Scan and Add

In case of Scan Topology, the Communication DTM may not detect the adapter (as it is transparent), and hence the Scan result will not return the Adapter. Instead, the scan result contains the HART FSK device. However, the HART FSK device can't be added to the topology due to different Protocol Ids.

In such scenario, unless the adapter acts as a HART node in the network, it is not possible to create the network topology in FDT3.0.

6.10.4 Conclusion

The FDT3.0 specification does not foresee transparent adapters as such, that allow general transformation from one protocol type to another protocol type.

In FDT3.0, transparent gateways can only be handled correctly, if they are supported/foreseen by the DTM for the parent device.

7 Protocol specific usage of general FDT data types

The FDT3.0 Specification already defines a set of data types that can be used to identify a device and to provide device information. Table 9 describes how these data types are used with HART.

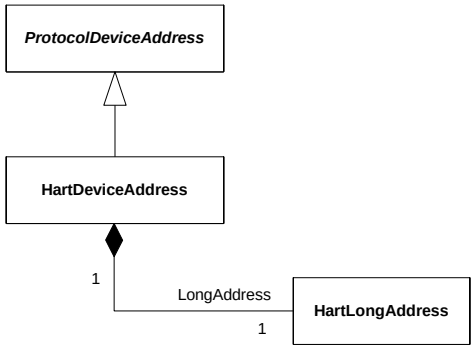
Table 9 – Usage of general data types

Data type / Property	Description for use in HART
ProtocolId	See chapter 4
PhysicalLayer.Id	See chapter 4
ApplicationDomain/ SemanticId	The application Domain is FDT_HART The SemanticId shall be build as described in chapter 5.3.3.

8 Protocol specific common data types

8.1 HartDeviceAddress

Table 10 describes the data type for addressing information used with protocol ID HART_Basic, HART_FSK, HART_Wireless, HART_RS485 and HART_Infrared.



Used in:

INetworkData::GetAddressInfo()
INetworkData::SetDeviceAddress()
ICommunication::BeginConnect()
ICommunication::EndConnect()
IScanning::EndScanRequest()

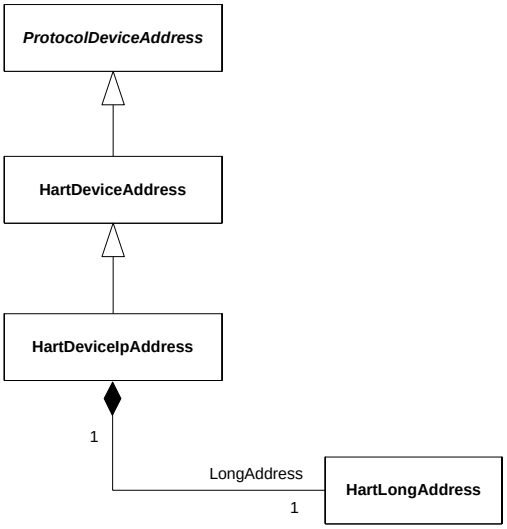
Figure 12 – HartDeviceAddress

Table 10 – HartDeviceAddress data type

Property	Description
ShortAddress	This property is intended to store the short address (see CMD#0) information.
AddressingMode	With this property the parent DTM defines which address property shall be used for the connection. The possible values of this enumeration are: • ShortAddress • ShortTag • LongTag • LongAddress
LongTag	This property is intended to store the LongTag (see CMD#20) information. Not to be synchronized with NetworkDataInfo.Label.
ShortTag	This property has store the Tag (see CMD#13). Not to be synchronized with NetworkDataInfo.Label.
LongAddress	Long frame address information according to the HART specification (see Table 16)

8.2 HartDeviceIpAddress

The following Table describes the data type derived from HartDeviceAddress with additional address information for HART TCP and UDP devices (Used with Protocol ID HART_IP).



Used in:

INetworkData::GetAddressInfo()
INetworkData::SetDeviceAddress()
ICommunication::BeginConnect()
ICommunication::EndConnect()
IScanning::EndScanRequest()

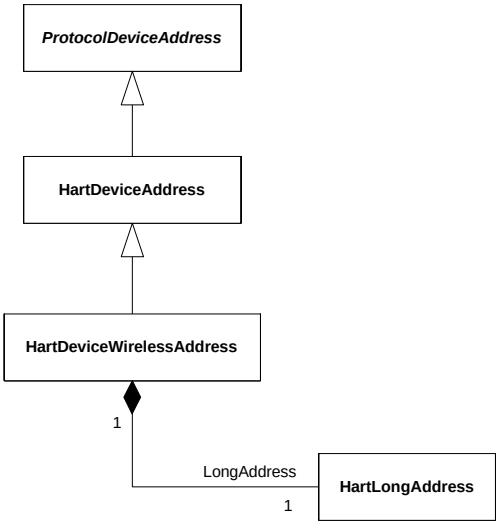
Figure 13 – HartDeviceIpAddress

Table 11 – HartDeviceIpAddress data type

Property	Description
ShortAddress	(Inherited from HartDeviceAddress)
AddressingMode	(Inherited from HartDeviceAddress)
LongTag	(Inherited from HartDeviceAddress)
ShortTag	(Inherited from HartDeviceAddress)
IpAddress	This property stores a host name or an IP address conformant to IPv4 or IPv6 standard.
Port	[Optional] This property stores the port number.
LongAddress	Long frame address information according to the HART specification (see Table 16)

8.3 HartDeviceWirelessAddress

The following Table describes the data type derived from HartDeviceAddress with additional address information for HART Wireless devices (Used with Protocol ID HART_Wireless).



Used in:

INetworkData::GetAddressInfo()
INetworkData::SetDeviceAddress()
ICommunication::BeginConnect()
ICommunication::EndConnect()
IScanning::EndScanRequest()

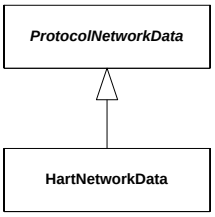
Figure 14 – HartDeviceWirelessAddress

Table 12 – HartDeviceWirelessAddress data type

Property	Description
ShortAddress	(Inherited from HartDeviceAddress)
AddressingMode	(Inherited from HartDeviceAddress)
LongTag	(Inherited from HartDeviceAddress)
ShortTag	(Inherited from HartDeviceAddress)
NetworkID	[Optional] This property stores the Network ID for a HART Wireless network.
LongAddress	Long frame address information according to the HART specification (see Table 16)

9 Network management

The data needed for management of the network are exposed by the Device DTM in the INetworkData interface.



Used in:

INetworkData::GetNetworkDataInfo()

Figure 15 – HartNetworkData

The following Table 13 describes the HART specific network data. This data is read only.

Table 13 – HartNetworkData data type

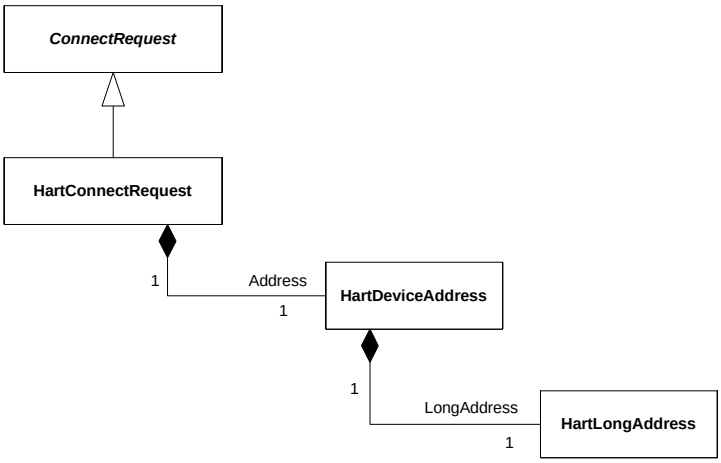
Attribute	Description
HartRevision	This property will be used as a informative constant that documents the HART major revision as communicated in CMD#0.
PollingAddressRange	This property describes the range of the polling address.

10 Communication data types

10.1 Introduction

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

10.2 HartConnectRequest



Used in:

ICommunication::BeginConnect()

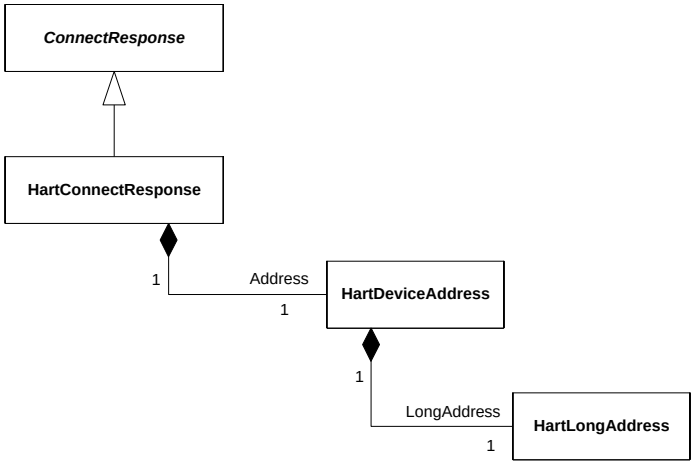
Figure 16 – HartConnectRequest

The properties of the data type are described in Table 15

Table 14 – HartConnectRequest datatype

Property	Description
Address	In order to establish a connection, the DTM has to provide address information of the HART Device. (see Table 10)
ProtocolId	Unique identifier of the HART protocol.
SystemTag	Unique identification of DTM within the Frame Application.
PreambleCount	[Optional] Provides a hint for the communication component about the number of preambles, required by the device type.
PrimaryMaster	[Optional] Provides a hint for a communication component that a DTM requires communication as primary or secondary master.
LongAddress	Long frame address information according to the HART specification (see Table 16)

10.3 HartConnectResponse



Used in:

ICommunication::EndConnect ()

Figure 17 – HartConnectResponse

The properties of the data type are described in Table 15

Table 15 – HartConnectResponse datatype

Property	Description
PreambleCount	Contains the information about the currently used preambleCount.
PrimaryMaster	Contains the information about the current state of the master
Address	Address information including ShortAddress, ShortTag, LongTag (see Table 10)
LongAddress	Long frame address information according to the HART specification (see Table 16)
CommunicationReference	Identifier for a communication link to a device.

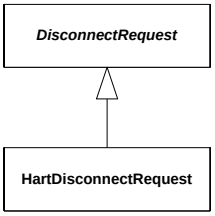
10.4 HartLongAddress

The properties of the data type are described in Table 16

Table 16 – HartLongAddress data type

Property	Description
Address1	1st byte of long address
Address2	2nd byte of long address
Address3	3rd byte of long address
Address4	4th byte of long address
Address5	5th byte of long address

10.5 HartDisconnectRequest



Used in:

ICommunication::BeginDisconnect()

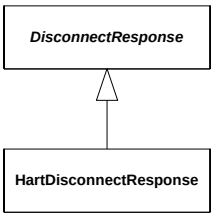
Figure 18 – HartDisconnectRequest

The properties of the data type are described in Table 17

Table 17 – HartDisconnectRequest data type

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

10.6 HartDisconnectResponse



Used in:

ICommunication::EndDisconnect()

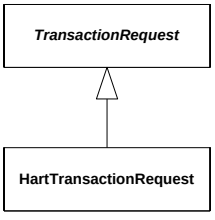
Figure 19 – HartDisconnectResponse

The properties of the data type are described in Table 18

Table 18 – HartDisconnectResponse data type

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

10.7 HartTransactionRequest



Used in:

ICommunication::BeginCommunicationRequest ()

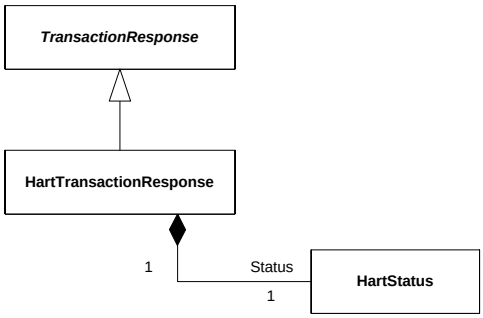
Figure 20 – HartTransactionRequest

The properties of the data type are described in Table 19

Table 19 – HartTransactionRequest data type

Property	Description
CommandNumber	HART command number
CommunicationData	Communication data to be transferred to the device
Id	[Optional] Identifier for a single Transaction Request.

10.8 HartTransactionResponse



Used in:

ICommunication::EndCommunicationRequest ()

Figure 21 – HartTransactionResponse

The properties of the data type are described in Table 20

Table 20 – HartTransactionResponse data type

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.
CommandNumber	HART command number
Status	Status information according to the HART specification
CommunicationData	Communication data received from the device

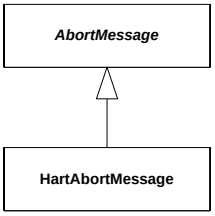
10.9 HartStatus

The properties of the data type are described in Table 21

Table 21 – HartStatus data type

Property	Description
DeviceStatus	Second status byte returned in HART command responses
CommandStatus	[Optional] Command Response Code, computed according HART specification from the first status byte returned in HART command responses
CommunicationStatus	[Optional] Communication error code, computed according HART specification from the first status byte returned in HART command responses

10.10 HartAbortMessage



Used in:

AbortCallback ()

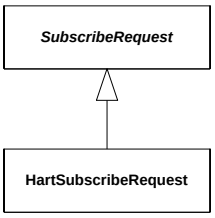
Figure 22 – HartAbortMessage

The properties of the data type are described in Table 22

Table 22 – HartAbortMessage data type

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

10.11 HartSubscribeRequest



Used in:

ISubscription::BeginSubscriptionInitialization ()

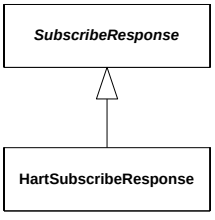
Figure 23 – HartSubscribeRequest

The properties of the data type are described in Table 23

Table 23 – HartSubscribeRequest datatype

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

10.12 HartSubscribeResponse



Used in:

ISubscription::EndSubscriptionInitialization ()

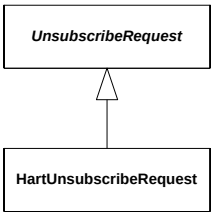
Figure 24 – HartSubscribeResponse

The properties of the data type are described in Table 24

Table 24 – HartSubscribeResponse data type

Property	Description
BurstModeDetected	Indicates whether the Communication Channel has detected that the device is already in burst mode.
CommunicationReference	Identifier for a communication link to a device.

10.13 HartUnsubscribeRequest



Used in:

ISubscription::BeginSubscriptionTermination ()

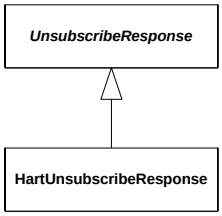
Figure 25 – HartUnsubscribeRequest

The properties of the data type are described in Table 25

Table 25 – HartUnsubscribeRequest data type

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

10.14 HartUnsubscribeResponse



Used in:

ISubscription::EndSubscriptionTermination ()

Figure 26 – HartUnsubscribeResponse

The properties of the data type are described in Table 26

Table 26 – HartUnsubscribeResponse data type

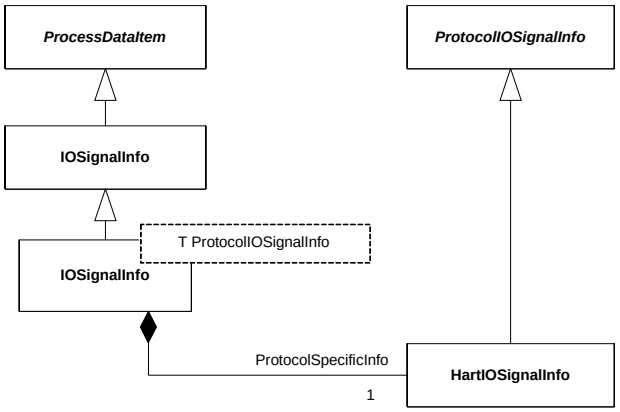
Property	Description
CommunicationReference	Identifier for a communication link to a device.

11 Data types for process data information

11.1 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.

11.2 HartIOSignalInfo



Used in:

IProcessData::EndGetProcessData ()

Figure 27 – HartIOSignalInfo

Table 27 shows which properties of IOSignalInfo data type should be provided or requires HART specific usage.

Table 27 – Usage of IOSignalInfo data type

Property	Description
SignalType	SignalType is always Input
SemanticInfos	See Table 8 for assigned variables
DeviceDataRef	Reference to the related variable, exposed in IDeviceData
UnitInfoRef	Reference to the variables providing range information
RangeInfoRefs	Reference to an enumeration variable describing the unit information

The properties of the HartIOSignalInfo data type are described in Table 28

Table 28 – HartIOSignalInfo data type

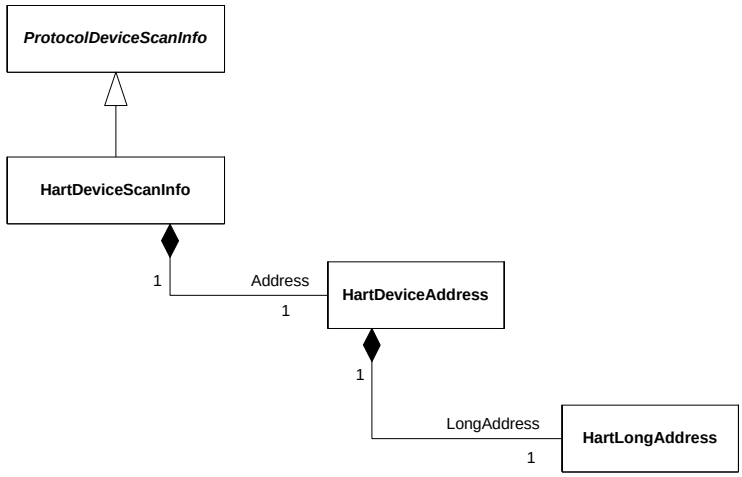
Property	Description
DeviceVariableAssignment	Constant enumeration value that can take following values - unassigned: The dynamic variable is not assigned to process variable and can only be accessed using the indexed approach. - PV: Variable is assigned to the PV process variable - SV: Variable is assigned to the SV process variable - TV: Variable is assigned to the TV process variable - QV: Variable is assigned to the QV process variable
DeviceVariableCode	Constant that specifies the device specific variable code (see additionally standard HCF table 34). Because of compatibility reasons to earlier FDT versions, this variable could be set to the value 252 that stands for “unknown”.

12 Device identification

12.1 General

This chapter defines identification relevant protocol specific data types.

12.2 HartDeviceScanInfo data type



Used in:

IDtmScanning::EndScanRequest ()

Figure 28 – HartDeviceScanInfo

The properties of the data type are described in Table 29.

Table 29 – HartDeviceScanInfo data type

Property	Description
Address	Address information of the scanned device.
HartMajorRevisionNumber	HART protocol revision.
ManufacturerIdentificationCode	Manufacturer specific code registered at the HART Foundation.
DeviceTypeCode	Device type specific code registered at the HART Foundation.
SoftwareRevision	Revision of the device embedded software.
HardwareRevisionLevel	Hardware revision of the device.
PhysicalSignalingCode	Physical signaling code.
DeviceId	Individual serial number of the device.
DeviceRevisionLevel	Device Command Revision Level supported by the device.
DeviceFlags	Device Flags.
ScannedPhysicalLayer	Determines the physical medium for which the scan information is provided.
ManufacturerSpecificExtension	[Optional] Can be used by DTM for a 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.

The following Table 30 defines the semantics of HartDeviceScanInfo properties and how this information is mapped to predefined properties of DeviceScanInfo. The Communication Channel will read these values from the device and writes them to the properties of HartDeviceScanInfo.

Table 30 – Protocol specific mapping of scan information

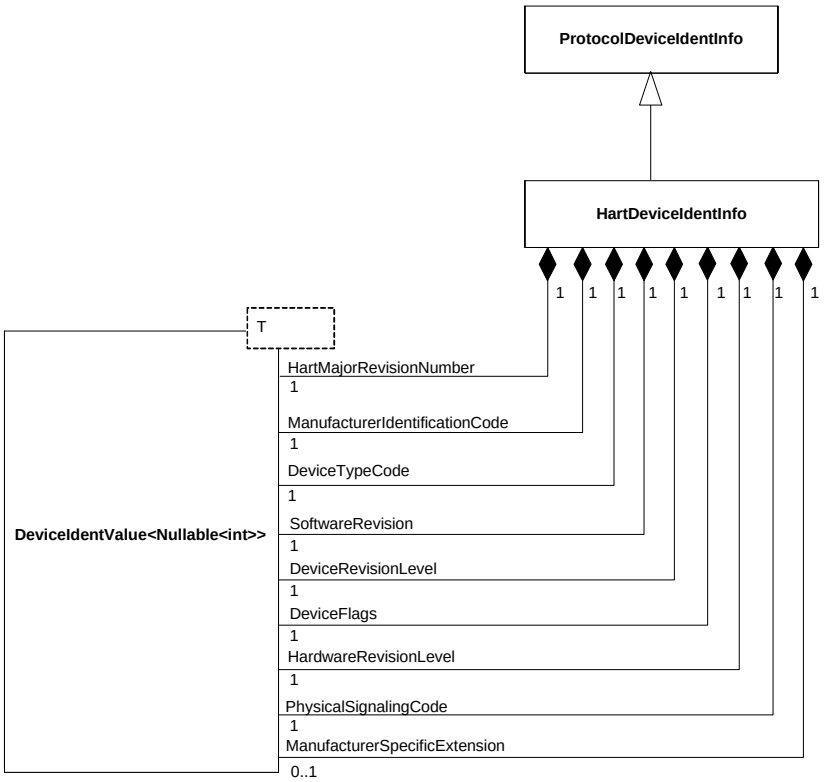
HartDeviceScanInfo property name	Data type of HartDevice-ScanInfo property	Mapped to property name in DeviceScanInfo	Data Request in physical device	Protocol Specific Name	HART Data Format	Specific Reference
Address.ShortAddress	int	Address.Address	To scan for HART devices CommChannel has to poll possible address range (HART 5: [0-15], HART 6: [0-63]) by calling Cmd #0. If Cmd #0 response is available, a physical device is connected to this address. Cmd #0 response does not contain short address value. If master using short address for polling receives a response, it can assume that short address of device is the same as used in polling request. In addition to this, polling address can be read from HART 6 device with Cmd #7.	Polling Address	Unsigned 8	[3] Chapter 6.8 Command #7 Read Loop Configuration.
Address.ShortTag	string	Tag (long or short tag depending on addressing rules)	Command #13 Bytes 0 - 5	Tag	6 Bytes or 8 Packed ASCII characters	[3] HART 6: HCF_Spec_127 – Universal Command Specification –
Address.LongTag	string	Tag (long or short tag depending on addressing rules)	HART6 or 7: Command #20 HART5: Command #12 Bytes 0 - 23	Long Tag Message	32 Bytes char 24 Bytes or 32 Packed ASCII characters	[3] Chapter 6.20 - Command #20 Read Long Tag [3] Chapter 6.12 - Command #12 Read Message
ProtocolId	GUID	ProtocolId	CommChannel has to pass Hart. ProtocolId in this attribute.			

HartDeviceScanInfo property name	Data type of HartDevice-ScanInfo property	Mapped to property name in DeviceScanInfo	Data Request in physical device	Protocol Specific Name	HART Data Format	Specific Reference
ScannedPhysicalLayer	PhysicalLayer	PhysicalLayer	CommChannel returns the physical layer, via which the scanned device is connected.			
ManufacturerIdentificationCode	int	ManufacturerId	HART7: Command #0 Byte 17+18. HART<7: Command #0 Byte 1.	Manufacturer Identification Code	16 bit unsigned enumerator	[3] Manufacturer Identification Codes
DeviceTypeCode	int	DeviceTypeId	HART 7: Command #0 Byte 1+2. HART<7: Command #0 Byte 2.	Device Type Code	16 bit unsigned enumerator	[3] Chapter 6.1 Command #0 Read Unique Identifier
SoftwareRevision	int	SoftwareRevision	Command 0 Byte 6.	Software Revision	8 bit unsigned integer	[3] Chapter 6.1 Command #0 Read Unique Identifier
HardwareRevisionLevel	int	HardwareRevision	Command #0 Byte 7 (Most significant 5 bits).	Hardware Revision	5 bit unsigned integer	[3] Chapter 6.1 Command #0 Read Unique Identifier
DeviceId	int	SerialNumber	Command #0 Bytes 9 - 11 (Device Identification Number).	Device Identification Number	Unsigned 24	[3] Chapter 6.1 Command #0 Read Unique Identifier
DeviceProfile	int	ProtocolIdentificationProfile	HART7: Command #0 Byte 21. HART<7: n/a	Device Profile Codes	8 bit unsigned enumerator	[3] Chapter 6.1 Command #0 Read Unique Identifier
PhysicalSignalingCode	int	Mapped to DeviceScanValues listed in ProtocolSpecificProperties with corresponding name as in HartDeviceScanInfo	Command #0 Byte 7 (Least significant 3 bits).	Physical Signaling Code	3 bit unsigned enumerator	[3] Chapter 6.1 Command #0 Read Unique Identifier
HartMajorRevisionNumber	int		Command #0 Byte 4.	HART Protocol Major Revision Number (e.g. 7 for HART 7)	8 bit unsigned integer	[3] Chapter 6.1 Command #0 Read Unique Identifier

HartDeviceScanInfo property name	Data type of HartDevice-ScanInfo property	Mapped to property name in DeviceScanInfo	Data Request in physical device	Protocol Specific Name	HART Data Format	Specific Reference
DeviceRevisionLevel	int		Command #0 Byte 5.	Device Revision Level	8 bit unsigned integer	Device Revision Level
DeviceFlags	int		Command #0 Byte 8.	Device Flags	Bit value according Flag Assignment table.	Bit value according Flag Assignment table.

12.3 HartDeviceIdentInfo data type

HartDeviceIdentInfo provides HART specific identifications of supported device types returned by GetDeviceIdentInfo(). For DTM assignment after fieldbus-scanning, the Frame Application can check in a protocol independent way if the identification of a scanned device type (DeviceScanInfo) matches to the supported DeviceIdentInfo.



Used in:

IDtmInformation::GetDeviceIdentInfo()

Figure 29 – HartDeviceIdentInfo

The properties of the data type are described in Table 31

Table 31 – HartDeviceIdentInfo data type

Property	Description
HartMajorRevisionNumber	HART protocol revision.
ManufacturerIdentificationCode	Manufacturer specific code registered at the HART Foundation.
DeviceTypeCode	Device type specific code registered at the HART Foundation.
SoftwareRevision	Revision of the device embedded software.
HardwareRevisionLevel	Hardware revision of the device.
PhysicalSignalingCode	Physical signaling code.
DeviceRevisionLevel	Device Command Revision Level supported by the device.
DeviceFlags	Device Flags.
ManufacturerSpecificExtension	[Optional] Can be used by DTM for a 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.

12.4 Mapping of information source

The following Table 32 defines the semantics of HartDeviceIdentInfo properties and how this information is mapped to predefined properties of DeviceIdentInfo.

Table 32 – Protocol specific mapping of identity information

HartDeviceIdentInfo property name	Data type of DeviceIdentValue in property in HartDeviceIdentInfo	Mapped to property in DeviceIdentInfo	Data Request in physical device	Protocol Specific Name	HART Data Format	Specific Reference
ProtocolId	Guid	ProtocolId	Device DTM shall pass Hart. ProtocolId in this attribute.			
ManufacturerIdentification Code	int	ManufacturerId	HART7: Command #0 Byte 17+18. HART<7: Command #0 Byte 1.	Manufacturer Identification Code	16 bit unsigned enumerator	[5] Manufacturer Identification Codes
DeviceTypeCode	int	DeviceTypeId	HART7: Command #0 Byte 1+2. HART<7: Command #0 Byte 2.	Device Type Code	16 bit unsigned enumerator	[3] Chapter 6.1 Command #0 Read Unique Identifier
SoftwareRevision	int	SoftwareRevision	Command #0 Byte 6.	Software Revision	8 bit unsigned integer	[3] Chapter 6.1 Command #0 Read Unique Identifier
HardwareRevisionLevel	int	HardwareRevision	Command #0 Byte 7 (Most significant 5 bits)	Hardware Revision	5 bit unsigned integer	[3] Chapter 6.1 Command #0 Read Unique Identifier
DeviceProfile	int	ProtocolIdentificationProfile	HART7: Command #0 Byte 21. HART<7: n/a	Device Profile Codes	8 bit unsigned enumerator	[3] Chapter 6.1 Command #0 Read Unique Identifier
PhysicalSignalingCode	int	Mapped to DeviceIdentValues listed in ProtocolSpecificProperties with corresponding name as in HartDeviceIdentInfo	Command #0 Byte 7 (Least significant 3 bits).	physical_signaling_code	3 bit unsigned enumerator	[3] Chapter 6.1 Command #0 Read Unique Identifier
HartMajorRevisionNumber	int		HART Command #0 Byte 4.	HART Protocol Major Revision Number (e.g. 7 for HART 7)	8 bit unsigned integer	[3] Chapter 6.1 Command #0 Read Unique Identifier

HartDeviceIdentInfo property name	Data type of DeviceIdentValue in property in HartDeviceIdentInfo	Mapped to property in DeviceIdentInfo	Data Request in physical device	Protocol Specific Name	HART Data Format	Specific Reference
DeviceRevisionLevel	int		HART Command #0 Byte 6.	transmitter_revision	8 bit unsigned integer	[3] Chapter 6.1 Command #0 Read Unique Identifier
DeviceFlags	int		HART Command #0 Byte 8.	device_flags	Bit value according Flag Assignment table.	[3] Chapter 6.1 Command #0 Read Unique Identifier

The HartDeviceIndentInfo properties may have either a single value which must exactly match to the supported device or a range of matching values may be defined in regular expressions.

Bibliography

[1] FDT 3.0 Specification V1.00, Order No.: 20-0001, FDT Group AISBL, 2020-06-04

[2] FDT2.1 for HART, Order No: 0002-0016-000, FDT Group AISBL, March 2018

[3] HCF Universal Command Specification, version 7.1, May 2008, HCF_SPEC-127

[4] HCF Common Practice Command Specification, version 9.1, May 2008, HCF_SPEC-151

[5] HCF Common Tables, version 20.3, November 2010, HCF_SPEC-183

[6] HCF Device Description Language Specification, version 12.1, October 2008, HCF_SPEC-500

[7] HCF DD Integrated Development Environment, version 4.1, July 2010, HCF_KIT-225

[8] HCF Command Summary Specification, version 9.0, July 2007, HCF_SPEC-99

[9] HCF HART Communication Protocol Specification, version 7.3, May 2011, HCF_SPEC-13

[10] HCF Network Management Specification, version 2.0, June 2012, HCF_SPEC-085