IEC/TR 62453-515

Edition 1.0 2009-08

TECHNICAL REPORT

colour inside

Field device tool (FDT) interface specification – Part 515: Communication implementation for common object model – IEC 61784 CPF 15

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

About IEC publications

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

- Catalogue of IEC publications: www.iec.ch/searchpub

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 It also gives information on projects, withdrawn and replaced publications.

- IEC Just Published: www.iec.ch/online_news/justpub

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

- Electropedia: www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.

- Customer Service Centre: www.iec.ch/webstore/custserv

If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:

Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

IEC/TR 62453-515

Edition 1.0 2009-08

TECHNICAL REPORT

Field device tool (FDT) interface specification – Part 515: Communication implementation for common object model – IEC 61784 CPF 15

INTERNATIONAL ELECTROTECHNICAL COMMISSION

PRICE CODE U

ICS 25.040.40; 35.100.05; 35.110

ISBN 2-8318-1058-0

CONTENTS

FOREWORD 3
INTRODUCTION 5
1 Scope 6
2 Normative references 6
3 Terms, definitions, symbols, abbreviated terms and conventions 7
3.1 Terms and definitions 7
3.2 Symbols and abbreviated terms 7
3.3 Conventions 7
3.3.1 Data type names and references to data types 7
3.3.2 Vocabulary for requirements 7
4 Bus category 7
5 Access to instance and device data 7
6 Protocol specific usage of general data types 7
7 Protocol specific common data types 8
8 Network management data types 8
8.1 General 8
8.2 Modbus device address - FDTModbusAddressSchema 8
9 Communication data types - FDTModbusCommunicationSchema 9
10 Channel parameter data types - FDTModbusChannelParameterSchema 17
11 Device identification 19
11.1 Device type identification data types - FDTModbusIdentSchema 19
11.2 Topology scan data types - DTMModbusDeviceSchema 20
11.3 Scan identification data types - FDTModbusScanIdentSchema 21
11.4 Device type identification data types - FDTModbusDeviceTypeIdentSchema 23
11.5 XSLT Transformation 24
Bibliography 30

Figure 1 – Part 515 of the IEC 62453 series 5

Table 1 – Protocol specific usage of general data types 8

INTERNATIONAL ELECTROTECHNICAL COMMISSION

FIELD DEVICE TOOL (FDT) INTERFACE SPECIFICATION –

Part 515: Communication implementation for common object model – IEC 61784 CPF 15

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with an IEC Publication.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

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

IEC/TR 62453-515, which is a technical report, has been prepared by subcommittee 65E: Devices and integration in enterprise systems, of IEC technical committee 65: Industrial-process measurement, control and automation:

This part, in conjunction with the other parts of the first edition of the IEC 62453 series cancels and replaces IEC/PAS 62453-1, IEC/PAS 62453-2, IEC/PAS 62453-3, IEC/PAS 62453-4 and IEC/PAS 62453-5 published in 2006, and constitutes a technical revision.

Each part of the IEC/TR 62453-5xy series is intended to be read in conjunction with its corresponding part in the IEC 62453-3xy series.

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

Enquiry draft	Report on voting
65E/71/DTR	65E/120/RVC

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

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

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

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

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

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

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

INTRODUCTION

This part of IEC 62453 is an interface specification for developers of FDT (Field Device Tool) components for function control and data access within a client/server architecture. The specification is a result of an analysis and design process to develop standard interfaces to facilitate the development of servers and clients by multiple vendors that need to interoperate seamlessly.

With the integration of fieldbusses into control systems, there are a few other tasks which need to be performed. In addition to fieldbus- and device-specific tools, there is a need to integrate these tools into higher-level system-wide planning- or engineering tools. In particular, for use in extensive and heterogeneous control systems, typically in the area of the process industry, the unambiguous definition of engineering interfaces that are easy to use for all those involved is of great importance.

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

Figure 1 shows how IEC/TR 62453-515 is aligned in the structure of IEC 62453 series.

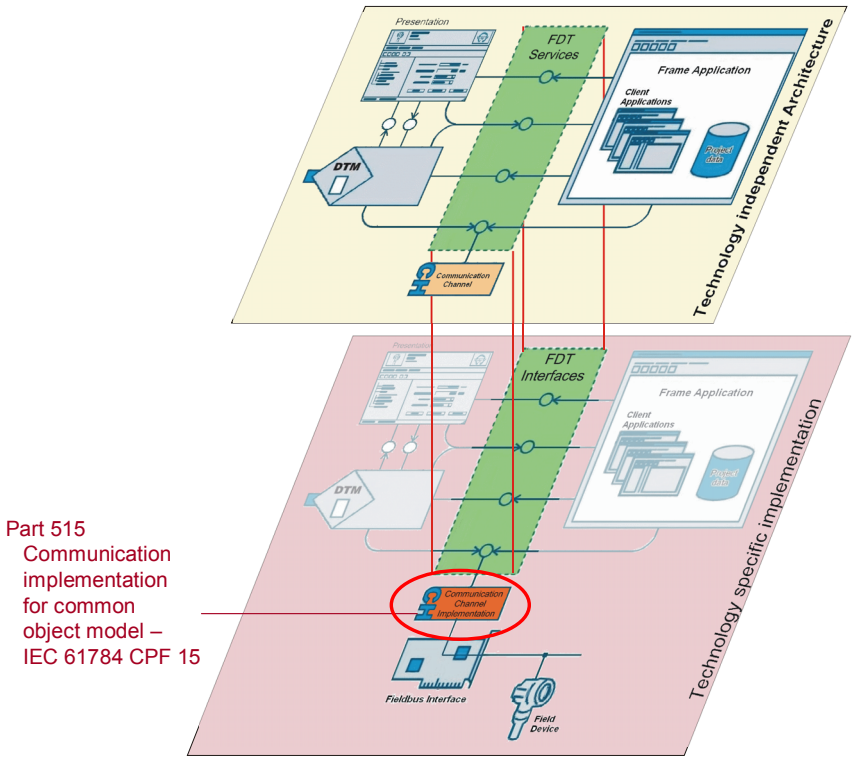

Figure 1 – Part 515 of the IEC 62453 series

FIELD DEVICE TOOL (FDT) INTERFACE SPECIFICATION –

Part 515: Communication implementation for common object model – IEC 61784 CPF 15

1 Scope

IEC/TR 62453-515, which is a technical report, provides information for integrating IEC 61784-2 CPF 15 (Modbus TCP®) and Modbus Serial Line®[1)] protocol support into FDT systems based on COM implementation. This part is to be used in conjunction with IEC/TR 62453-41.

NOTE This part of IEC 62453 only specifies the mapping of Modbus parameters to FDT data types. For restrictions of protocol specific parameters concerning allowed values and concerning limitations of arrays used in the definition of FDT data types, refer to IEC 61158-5-15 and the MODBUS Application Protocol Specification.

This part of IEC 62453 specifies communication and other services.

This specification neither contains the FDT specification nor modifies it.

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

IEC 61784-2, *Industrial communication networks – Profiles – Part 2: Additional fieldbus profiles for real-time networks based on ISO/IEC 8802-3*

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

IEC 62453-2:2009,*Field Device Tool (FDT) interface specification – Part 2: Concepts and detailed description*

IEC/TR 62453-41:2009, *Field Device Tool (FDT) interface specification – Part 41: Object model integration profile – Common object model*

IEC 62453-315:2009, *Field Device Tool (FDT) interface specification – Part 315: Communication profile integration – IEC 61784 CPF 15*

1) Modbus is the trademark of Schneider Automation Inc. It is registered in the United States of America. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trademark holder or any of its products. Compliance to this profile does not require use of the trademark Modbus. Use of the trademark Modbus requires permission from Schneider Automation Inc.

3 Terms, definitions, symbols, abbreviated terms and conventions

3.1 Terms and definitions

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

3.2 Symbols and abbreviated terms

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

CP	Communication Profile	[IEC 61784-1]
CPF	Communication Profile Family	[IEC 61784-1]

3.3 Conventions

3.3.1 Data type names and references to data types

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.

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 behavior.
Usage of “can’ or “Optional’	Function or behavior may be provided, depending on defined conditions.

4 Bus category

IEC 61784 CPF 15 protocol is identified by the attribute definition busCategory as specified in IEC 62453-315 (protocol identifiers).

5 Access to instance and device data

Used at methods:

- IDtmParameter::GetParameters()
- IDtmParameter::SetParameters()

These methods shall provide access to at least to all parameters defined in the IEC 62453-315.

6 Protocol specific usage of general data types

Table 1 shows how general data types are used with IEC 61784 CPF 15 devices.

Table 1 – Protocol specific usage of general data types

Attribute	Description for use
fdt:address	All these attributes of the FDTDatatype schema are used as defined in IEC 62453-315.
fdt:protocolId	
fdt:deviceTypeId	
fdt:deviceTypeInformation	
fdt:deviceTypeInformationPath	
fdt:manufacturerId	
fdt:semanticId fdt:applicationDomain	
fdt:tag	

7 Protocol specific common data types

This clause specifies the protocol specific common data types, which are used in the definition of other data types.

8 Network management data types

8.1 General

The data types specified in this clause are used at following methods:

- IDtmParameter:GetParameters
- IDtmParameter:SetParameters

8.2 Modbus device address - FDTModbusAddressSchema

The address of a Modbus device is available at the element <BusInformation/UserdefinedBus>.

```
<?xml version="1.0"?>
<Schema name="FDTModbusAddressSchema" xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes" xmlns:fdt="x-schema:FDTDataTypesSchema.xml">

<!--  Address schema for Modbus protocol V1.0 -->
<!-- This additional schema describes the different methods of addressing for Modbus TCP and Modbus Serial Line
Devices  -->

    <AttributeType name = "schemaVersion" dt:type = "number" default = "1.0"/>

   <!--Definition of attributes for Modbus addressing-->

    <!-- Slave address for Modbus Serial Line -->
    <AttributeType name="slaveAddress" dt:type="ui1"/>
    <!-- IP address for Modbus TCP -->
    <AttributeType name="tcpAddress" dt:type="string"/>
   <!-- TCP Port for Modbus TCP (default 502) -->
    <AttributeType name="tcpPort" dt:type="ui2" default = "502"/>

    <ElementType name="ModbusTCP" content="mixed" model="closed">
        <attribute type="fdt:nodeId" required="no"/>
        <attribute type="tcpAddress" required="yes"/>
      <attribute type="tcpPort" required="no"/>
      <attribute type="slaveAddress" required="no"/>
    </ElementType>
```

```
    <ElementType name="ModbusSerial" content="mixed" model="closed">
        <attribute type="fdt:nodeId" required="no"/>
        <attribute type="slaveAddress" required="yes"/>
    </ElementType>
</Schema>
```

9 Communication data types - FDTModbusCommunicationSchema

The data types specified in this clause are used with the methods of IFdtCommunication.

```
<?xml version = "1.0" encoding = "UTF-8"?>

<!--FDT communication schema for Modbus protocol V1.0-->
<!--This schema describes all Modbus services which are defined in the Modbus Application Protocol Specification
V1.1a from 4 June 2004 -->
<!--Due to the ongoing standardisation process in the CIA Group to specify the encapsulation of the CanOpen
protocol in Modbus -->
<!--it was abstained to describe the Modbus service 0x2B/0x0D  -->

<Schema name = "FDTModbusCommunicationSchema"
    xmlns = "urn:schemas-microsoft-com:xml-data"
     xmlns:dt = "urn:schemas-microsoft-com:datatypes"
    xmlns:fdt = "x-schema:FDTDataTypesSchema.xml"
    xmlns:mb = "x-schema:FDTModbusAddressSchema.xml">

    <AttributeType name = "schemaVersion" dt:type = "number" default = "1.0"/>
    <AttributeType name = "communicationReference" dt:type = "uuid"/>


    <!--Modbus protocol parameters-->

    <!--ModbusException response-->
    <AttributeType name = "modbusService" dt:type = "enumeration" dt:values = "ReadCoils ReadDiscreteInputs
ReadHoldingRegisters ReadInputRegisters WriteSingleCoil WriteSingleRegister ReadExceptionStatus Diagnostics
GetCommEventCounter GetCommEventLog WriteMultipleCoils WriteMultipleRegisters ReportSlaveID
ReadFileRecord WriteFileRecord MaskWriteRegister ReadWriteRegisters ReadFifoQueue
EncapsulatedInterfaceTransport ReadDeviceIdentification PrivateModbus"/>
    <AttributeType name = "modbusExceptionCode" dt:type = "bin.hex" dt:minLength = "1"/>

    <!--Read/Write Data attributes-->
    <AttributeType name = "outputAddress" dt:type = "ui2"/>
    <AttributeType name = "startAddress" dt:type = "ui2"/>
    <AttributeType name = "quantity" dt:type = "ui2"/>
    <AttributeType name = "multipleCoilValues" dt:type = "string" dt:minLength = "1" dt:maxLength = "2000"/>
    <AttributeType name = "discreteInputsStatus" dt:type = "string" dt:minLength = "1" dt:maxLength = "2000"/>
    <AttributeType name = "registerValues" dt:type = "bin.hex" dt:minLength = "2" dt:maxLength = "250"/>
    <AttributeType name = "singleCoilValue" dt:type = "boolean"/>
    <AttributeType name = "singleRegister" dt:type = "bin.hex" dt:maxLength = "2" dt:minLength = "2"/>
    <!-- Diagnostic services attributes-->
    <AttributeType name = "exceptionStatus" dt:type = "bin.hex" dt:maxLength = "1" dt:minLength = "1"/>
    <AttributeType name = "commStatus" dt:type = "bin.hex" dt:maxLength = "2" dt:minLength = "2"/>
    <AttributeType name = "diagnosticsSubFct" dt:type = "bin.hex" dt:maxLength = "2" dt:minLength = "2"/>
    <AttributeType name = "diagnosticsData" dt:type = "bin.hex" dt:maxLength = "250" dt:minLength = "2"/>
    <AttributeType name = "data" dt:type = "bin.hex"/>
    <AttributeType name = "eventCount" dt:type = "bin.hex" dt:maxLength = "2" dt:minLength = "2"/>
    <AttributeType name = "messageCount" dt:type = "bin.hex" dt:maxLength = "2" dt:minLength = "2"/>
    <AttributeType name = "events" dt:type = "bin.hex"/>
    <!-- Read/Write File Record attributes-->
    <AttributeType name = "referenceType" dt:type = "bin.hex" dt:maxLength = "1" dt:minLength = "1"/>
    <AttributeType name = "fileNumber" dt:type = "bin.hex" dt:maxLength = "2" dt:minLength = "2"/>
    <AttributeType name = "recordNumber" dt:type = "bin.hex" dt:maxLength = "2" dt:minLength = "2"/>
    <AttributeType name = "recordData" dt:type = "bin.hex" dt:minLength = "2"/>
    <!-- Mask Write Register attributes-->
    <AttributeType name = "referenceAddress" dt:type = "ui2"/>
    <AttributeType name = "andMask" dt:type = "bin.hex" dt:maxLength = "2" dt:minLength = "2"/>
    <AttributeType name = "orMask" dt:type = "bin.hex" dt:maxLength = "2" dt:minLength = "2"/>
    <!-- Read/Write Registers attributes-->
    <AttributeType name = "readStartAddress" dt:type = "ui2"/>
```

```
 <AttributeType name = "readRegisterValues" dt:type = "bin.hex" dt:minLength = "2" dt:maxLength = "250"/>
 <AttributeType name = "readQuantity" dt:type = "ui2"/>
 <AttributeType name = "writeStartAddress" dt:type = "ui2"/>
 <AttributeType name = "writeRegisterValues" dt:type = "bin.hex" dt:minLength = "2" dt:maxLength = "242"/>
 <!-- Read FIFO Queue attributes -->
 <AttributeType name = "fifoPointerAddress" dt:type = "ui2"/>
 <AttributeType name = "fifoRegisterValues" dt:type = "bin.hex" dt:minLength = "2" dt:maxLength = "62"/>
 <!-- Encapsulated Interface Transport attributes -->
 <AttributeType name = "meiType" dt:type = "bin.hex" dt:maxLength = "1" dt:minLength = "1"/>
 <AttributeType name = "meiData" dt:type = "bin.hex" dt:minLength = "1" dt:maxLength = "251"/>
 <!-- Device identification attributes-->
 <AttributeType name = "readDeviceIdCode" dt:type = "ui1"/>
 <AttributeType name = "objectId" dt:type = "bin.hex" dt:minLength = "1" dt:maxLength =" 1"/>
 <AttributeType name = "conformityLevel" dt:type = "bin.hex" dt:maxLength = "1" dt:minLength = "1"/>
 <AttributeType name = "moreFollows" dt:type = "boolean"/>
 <AttributeType name = "nextObjectId" dt:type = "bin.hex" dt:minLength = "1" dt:maxLength =" 1"/>
 <AttributeType name = "numberOfObjects" dt:type = "ui1"/>
 <AttributeType name = "objectValue" dt:type = "bin.hex" dt:minLength = "1" dt:maxLength = "244"/>
 <!-- Private Request attributes-->
 <AttributeType name = "privateRequest" dt:type = "bin.hex"/>
 <AttributeType name = "privateResponse" dt:type = "bin.hex"/>


 <ElementType name = "ConnectRequest" content = "eltOnly"  model = "closed">
<attribute type="fdt:systemTag" required="yes"/>
     <group order = "one" maxOccurs="1" minOccurs="1">
         <element type = "mb:ModbusSerial"/>
         <element type = "mb:ModbusTCP"/>
     </group>
 </ElementType>

 <ElementType name = "ConnectResponse" content = "empty" model = "closed">
     <attribute type = "communicationReference" required = "yes"/>
 </ElementType>

 <ElementType name = "DisconnectRequest" content = "empty" model = "closed">
     <attribute type = "communicationReference" required = "yes"/>
 </ElementType>

 <ElementType name = "DisconnectResponse" content = "empty" model = "closed">
     <attribute type = "communicationReference" required = "yes"/>
 </ElementType>

<ElementType name="Abort" content="empty" model="closed">
   <attribute type="communicationReference" required="no"/>
</ElementType>

 <!--Modbus Exception response-->

 <ElementType name = "ModbusExceptionRsp" content = "empty" model = "closed">
     <attribute type = "communicationReference" required = "yes"/>
     <!--Applied Modbus service, which failed-->
     <attribute type = "modbusService" required = "yes"/>
     <!-- Modbus Exception code-->
     <attribute type = "modbusExceptionCode" required = "yes"/>
 </ElementType>


 <!--Definition of Modbus services-->


 <!-- 0x01 Read Coils -->

 <ElementType name = "ReadCoilsReq" content = "empty" model = "closed">
     <attribute type = "communicationReference" required = "yes"/>
     <!-- Starting address: 0x0000 to 0xFFFF (2 bytes) -->
     <attribute type = "startAddress" required = "yes"/>
     <!--Quantity of coils: 0x0001 to 0x07D0 (2 bytes) -->
     <attribute type = "quantity" required = "yes"/>
 </ElementType>
```

```
    <ElementType name = "ReadCoilsRsp" content = "empty" model = "closed">
        <attribute type = "communicationReference" required = "yes"/>
        <!-- Coil status: ASCII string with each coil state coded in one character ("1"==TRUE; "0"==FALSE) -->
        <attribute type = "multipleCoilValues" required = "yes"/>
    </ElementType>


    <!-- 0x02 Read Discrete Inputs -->

    <ElementType name = "ReadDiscreteInputsReq" content = "empty" model = "closed">
        <attribute type = "communicationReference" required = "yes"/>
        <!-- Starting address: 0x0000 to 0xFFFF (2 bytes) -->
        <attribute type = "startAddress" required = "yes"/>
        <!-- Quantity of Inputs:  0x0001 to 0x07D0 (2 bytes)  -->
        <attribute type = "quantity" required = "yes"/>
    </ElementType>

    <ElementType name = "ReadDiscreteInputsRsp" content = "empty" model = "closed">
        <attribute type = "communicationReference" required = "yes"/>
        <!-- Input status: ASCII string with each discrete input state coded in one character ("1"==TRUE;
"0"==FALSE) -->
        <attribute type = "discreteInputsStatus" required = "yes"/>
    </ElementType>

    <!-- 0x03 Read Holding Registers -->

    <ElementType name = "ReadHoldingRegistersReq" content = "empty" model = "closed">
        <attribute type = "communicationReference" required = "yes"/>
        <!-- Starting address:  0x0000 to 0xFFFF (2 bytes) -->
        <attribute type = "startAddress" required = "yes"/>
        <!-- Quantity of registers: 0x0001 to 0x007D (2 bytes) -->
        <attribute type = "quantity" required = "yes"/>
    </ElementType>

    <ElementType name = "ReadHoldingRegistersRsp" content = "empty" model = "closed">
        <attribute type = "communicationReference" required = "yes"/>
        <!-- Register value: read holding register values -->
        <attribute type = "registerValues" required = "yes"/>
    </ElementType>


    <!-- 0x04 Read Input Registers -->

    <ElementType name = "ReadInputRegistersReq" content = "empty" model = "closed">
        <attribute type = "communicationReference" required = "yes"/>
        <!-- Starting address: 0x0000 to 0xFFFF (2 bytes)-->
        <attribute type = "startAddress" required = "yes"/>
        <!-- Quantity of Input registers: 0x0001 to 0x007D (2 bytes) -->
        <attribute type = "quantity" required = "yes"/>
    </ElementType>

    <ElementType name = "ReadInputRegistersRsp" content = "empty" model = "closed">
        <attribute type = "communicationReference" required = "yes"/>
        <!-- Input registers: read input register values -->
        <attribute type = "registerValues" required = "yes"/>
    </ElementType>


    <!-- 0x05 Write Single Coil -->

    <ElementType name = "WriteSingleCoilReq" content = "empty" model = "closed">
        <attribute type = "communicationReference" required = "yes"/>
        <!-- Output address: 0x0000 to 0xFFFF (2 bytes) -->
        <attribute type = "outputAddress" required = "yes"/>
        <!-- Output value: boolean (1="true", 0=="false") -->
        <attribute type = "singleCoilValue" required = "yes"/>
    </ElementType>

    <ElementType name = "WriteSingleCoilRsp" content = "empty" model = "closed">
```

```
    <attribute type = "communicationReference" required = "yes"/>
</ElementType>


<!-- 0x06 Write Single Register -->

<ElementType name = "WriteSingleRegisterReq" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- Register address: 0x0000 to 0xFFFF (2 bytes)  -->
    <attribute type = "outputAddress" required = "yes"/>
    <!-- Register value: 0x0000 to 0xFFFF (2 bytes) -->
    <attribute type = "singleRegister" required = "yes"/>
</ElementType>

<ElementType name = "WriteSingleRegisterRsp" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
</ElementType>


<!-- 0x07 Read Exception Status (serial line only!!) -->

<ElementType name = "ReadExceptionStatusReq" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
</ElementType>

<ElementType name = "ReadExceptionStatusRsp" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- Output data: 0x00 to 0xFF (1 byte) -->
    <attribute type = "exceptionStatus" required = "yes"/>
</ElementType>


<!-- 0x08 Diagnostics (serial line only!!) -->

<ElementType name = "DiagnosticsReq" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- Sub-function -->
    <attribute type = "diagnosticsSubFct" required = "yes"/>
    <!-- Data: Nx2 bytes -->
    <attribute type = "diagnosticsData" required = "yes"/>
</ElementType>

<ElementType name = "DiagnosticsRsp" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- Sub-function -->
    <attribute type = "diagnosticsSubFct" required = "yes"/>
    <!-- Data: Nx2 bytes -->
    <attribute type = "diagnosticsData" required = "no"/>
</ElementType>


<!-- 0x0B Get Comm Event Counter (serial line only!!) -->

<ElementType name = "GetCommEventCounterReq" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
</ElementType>

<ElementType name = "GetCommEventCounterRsp" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- Status:  0x0000 to 0xFFFF (2 bytes) -->
    <attribute type = "commStatus" required = "yes"/>
    <!-- Event count: 0x0000 to 0xFFFF (2 bytes) -->
    <attribute type = "eventCount" required = "yes"/>
</ElementType>


<!-- 0x0C Get Comm Event Log (serial line only!!) -->

<ElementType name = "GetCommEventLogReq" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
```

```
</ElementType>

<ElementType name = "GetCommEventLogRsp" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- Status: 0x0000 to 0xFFFF (2 bytes) -->
    <attribute type = "commStatus" required = "yes"/>
    <!-- Event count: 0x0000 to 0xFFFF (2 bytes) -->
    <attribute type = "eventCount" required = "yes"/>
    <!-- Message count: 0x0000 to 0xFFFF (2 bytes) -->
    <attribute type = "messageCount" required = "yes"/>
    <!-- Events: (N-6) x 1 byte (2 bytes) -->
    <attribute type = "events" required = "no"/>
</ElementType>


<!-- 0x0F Write Multiple Coils -->

<ElementType name = "WriteMultipleCoilsReq" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- Starting address: 0x0000 to 0xFFFF (2 bytes) -->
    <attribute type = "outputAddress" required = "yes"/>
    <!-- Outputs value: ASCII string with each coil state coded in one character ("1"==TRUE; "0"==FALSE) -->
    <attribute type = "multipleCoilValues" required = "yes"/>
</ElementType>

<ElementType name = "WriteMultipleCoilsRsp" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
</ElementType>


<!-- 0x10 Write Multiple Registers -->

<ElementType name = "WriteMultipleRegistersReq" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- Starting address: 0x0000 to 0xFFFF (2 bytes) -->
    <attribute type = "outputAddress" required = "yes"/>
    <!--Register value:  register values to write -->
    <attribute type = "registerValues" required = "yes"/>
</ElementType>

<ElementType name = "WriteMultipleRegistersRsp" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
</ElementType>


<!-- 0x11 Report Slave ID (serial line only!!) -->

<ElementType name = "ReportSlaveIDReq" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
</ElementType>

<ElementType name = "ReportSlaveIDRsp" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- data: This attribute contains the Slave ID, the Run Indicator Status and -->
  <!-- the additional device specific data -->
  <!-- in the same format and order as defined in the Modbus specification -->
    <attribute type = "data" required = "yes"/>
</ElementType>

<!-- 0x14/0x06 Read File Record -->

<!-- Definition of sub-request structure  -->

<ElementType name = "ReadFileSubRequest" content = "empty" model = "closed">
    <!-- Sub-request x. reference type: must be 0x06  (1 byte) -->
    <attribute type = "referenceType" default="06" required = "yes"/>
    <!-- Sub-request x. File Number: 0x0000 to 0xFFFF (2 bytes) -->
    <attribute type = "fileNumber" required = "yes"/>
    <!-- Sub-request x. Record Number: 0x0000 to 0x270F (2 bytes) -->
    <attribute type = "recordNumber" required = "yes"/>
```

```
    <!-- Sub-request x. Register Length: N= 2 bytes (number of registers = record length) -->
    <attribute type = "quantity" required = "yes"/>
</ElementType>

<!-- Definition of main request structure  -->

<ElementType name = "ReadFileRecordReq" content = "eltOnly"   model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- Sub-request: see ReadFileSubRequest -->
    <element type= "ReadFileSubRequest" minOccurs="1" maxOccurs="*" />
</ElementType>

<!-- Definition of sub-response structure  -->

<ElementType name = "ReadFileSubResponse" content = "empty" model = "closed">
    <!-- Sub-request x. Record Data: Nx2 bytes -->
    <attribute type = "recordData" required = "yes"/>
</ElementType>

<!-- Definition of main response structure  -->

<ElementType name = "ReadFileRecordRsp" content = "eltOnly" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- Sub-request: see ReadFileSubResponse -->
    <element type= "ReadFileSubResponse" minOccurs="1" maxOccurs="*" />
</ElementType>

<!-- 0x15/0x06 Write File Record -->

<!-- Definition of sub-request structure  -->

<ElementType name = "WriteFileSubRequest" content = "empty" model = "closed">
    <!-- Sub-request x. Reference Type: must be 0x06  (1 byte) -->
    <attribute type = "referenceType" default="06" required = "yes"/>
    <!-- Sub-request x. File Number: 0x0000 to 0xFFFF (2 bytes) -->
    <attribute type = "fileNumber" required = "yes"/>
    <!-- Sub-request x. Record Number: 0x0000 to 0x270F (2 bytes) -->
    <attribute type = "recordNumber" required = "yes"/>
    <!-- Sub-request x. Record Data: Nx2 bytes -->
    <attribute type = "recordData" required = "yes"/>
</ElementType>

<!-- Definition of main request structure  -->

<ElementType name = "WriteFileRecordReq" content = "eltOnly"   model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- Sub-request: see WriteFileSubRequest -->
    <element type= "WriteFileSubRequest" minOccurs="1" maxOccurs="*"/>
</ElementType>

<!-- Definition of main response structure  -->

<ElementType name = "WriteFileRecordRsp" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
</ElementType>

<!-- 0x16 Mask Write Register -->

<ElementType name = "MaskWriteRegisterReq" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- Reference address: 0x0000 to 0xFFFF (2 bytes)  -->
    <attribute type = "referenceAddress" required = "yes"/>
    <!-- AND_Mask: 0x0000 to 0xFFFF (2 bytes) -->
    <attribute type = "andMask" required = "yes"/>
    <!-- OR_Mask: 0x0000 to 0xFFFF (2 bytes) -->
    <attribute type = "orMask" required = "yes"/>
</ElementType>

<ElementType name = "MaskWriteRegisterRsp" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
```

```
</ElementType>

<!-- 0x17 Read/Write Registers -->

<ElementType name = "ReadWriteRegistersReq" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- Read starting address: 0x0000 to 0xFFFF (2 bytes) -->
    <attribute type = "readStartAddress" required = "yes"/>
    <!-- Quantity to read: 0x0001 to 0x007D (2 bytes) -->
    <attribute type = "readQuantity" required = "yes"/>
    <!-- Write starting address: 0x0000 to 0xFFFF (2 bytes) -->
    <attribute type = "writeStartAddress" required = "yes"/>
    <!-- Write register values: register values to write -->
    <attribute type = "writeRegisterValues" required = "yes"/>
</ElementType>

<ElementType name = "ReadWriteRegistersRsp" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- Read register values: read register values -->
    <attribute type = "readRegisterValues" required = "yes"/>
</ElementType>

<!-- 0x17 Read Fifo Queue -->

<ElementType name = "ReadFifoQueueReq" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- FIFO pointer address: 0x0000 to 0xFFFF (2 bytes) -->
    <attribute type = "fifoPointerAddress" required = "yes"/>
</ElementType>

<ElementType name = "ReadFifoQueueRsp" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- FIFO register values -->
    <attribute type = "fifoRegisterValues" required = "yes"/>
</ElementType>

<!-- 0x2B Encapsulated Interface Transport -->

<ElementType name = "EncapsulatedInterfaceTransportReq" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- MEI type: (1 byte) -->
    <attribute type = "meiType" required = "yes"/>
    <!-- MEI type specific data: n bytes -->
    <attribute type = "meiData" required = "yes"/>
</ElementType>

<ElementType name = "EncapsulatedInterfaceTransportRsp" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- MEI type: (1 byte) -->
    <attribute type = "meiType" required = "yes"/>
    <!-- MEI type specific data: n bytes -->
    <attribute type = "meiData" required = "yes"/>
</ElementType>

<!-- 0x2B/0x0E Read Device Identification -->

<!-- Definition of identification object structure -->

<ElementType name = "IdentificationObject" content = "empty" model = "closed">
    <!-- Object ID:  (1 byte) -->
    <attribute type = "objectId" required = "yes"/>
    <!-- Object Value -->
    <attribute type = "objectValue" required = "yes"/>
</ElementType>

<!-- Definition of main request structure  -->

<ElementType name = "ReadDeviceIdentificationReq" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- Read device ID code:  01 / 02/ 03 / 04 (1 byte) -->
```

```
    <attribute type = "readDeviceIdCode" required = "yes"/>
    <!-- Object ID: first object to read  (1 byte) -->
    <attribute type = "objectId" required = "yes"/>
</ElementType>

<!-- Definition of main response structure  -->

<ElementType name = "ReadDeviceIdentificationRsp" content = "eltOnly" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- Read device ID code: 01 / 02/ 03 / 04 (1 byte) -->
    <attribute type = "readDeviceIdCode" required = "yes"/>
    <!-- Conformity level: (1 byte) -->
    <attribute type = "conformityLevel" required = "yes"/>
    <!-- More follows: boolean (1="true", 0=="false") -->
    <attribute type = "moreFollows" required = "yes"/>
    <!-- Next object ID:  (1 byte)-->
    <attribute type = "nextObjectId" required = "yes"/>
    <!-- Number of objects: number of identification objects returned in the response (1 byte)-->
    <attribute type = "numberOfObjects" required = "yes"/>
    <!-- Identification object: see IdentificationObject -->
    <element type= "IdentificationObject" minOccurs="1" maxOccurs="*"/>
</ElementType>

<!-- Private Modbus Request -->

<ElementType name = "PrivateModbusReq" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- Private Modbus request coded as hexadecimal byte-array -->
    <attribute type = "privateRequest" required = "yes"/>
</ElementType>

<ElementType name = "PrivateModbusRsp" content = "empty" model = "closed">
    <attribute type = "communicationReference" required = "yes"/>
    <!-- Private Modbus response coded as hexadecimal byte-array -->
    <attribute type = "privateResponse" required = "yes"/>
</ElementType>


<!-- Unconfirmed Private Modbus Request -->

<ElementType name = "UnconfirmedPrivateModbusReq" content = "empty" model = "closed">
   <attribute type = "communicationReference" required = "yes"/>
   <!-- Private Modbus request coded as hexadecimal byte-array -->
   <attribute type = "privateRequest" required = "yes"/>
</ElementType>

<ElementType name = "UnconfirmedPrivateModbusRsp" content = "empty" model = "closed">
   <attribute type = "communicationReference" required = "yes"/>
</ElementType>

<!-- Main FDT element -->

<ElementType name = "FDT" content = "eltOnly" order = "one" model = "closed">
    <attribute type = "schemaVersion"/>
    <attribute type = "fdt:nodeId"/>
    <group order = "one" maxOccurs="1" minOccurs="1">
        <element type = "ConnectRequest" />
        <element type = "ConnectResponse" />
        <element type = "DisconnectRequest" />
        <element type = "DisconnectResponse" />
     <element type = "Abort" />
        <element type = "ModbusExceptionRsp"/>
        <element type = "ReadCoilsReq"/>
        <element type = "ReadCoilsRsp"/>
        <element type = "ReadDiscreteInputsReq"/>
        <element type = "ReadDiscreteInputsRsp"/>
        <element type = "ReadHoldingRegistersReq"/>
        <element type = "ReadHoldingRegistersRsp"/>
        <element type = "ReadInputRegistersReq"/>
        <element type = "ReadInputRegistersRsp"/>
```

```
            <element type = "WriteSingleCoilReq"/>
            <element type = "WriteSingleCoilRsp"/>
            <element type = "WriteSingleRegisterReq"/>
            <element type = "WriteSingleRegisterRsp"/>
            <element type = "ReadExceptionStatusReq"/>
            <element type = "ReadExceptionStatusRsp"/>
            <element type = "DiagnosticsReq"/>
            <element type = "DiagnosticsRsp"/>
            <element type = "GetCommEventCounterReq"/>
            <element type = "GetCommEventCounterRsp"/>
            <element type = "GetCommEventLogReq"/>
            <element type = "GetCommEventLogRsp"/>
            <element type = "WriteMultipleCoilsReq"/>
            <element type = "WriteMultipleCoilsRsp"/>
            <element type = "WriteMultipleRegistersReq"/>
            <element type = "WriteMultipleRegistersRsp"/>
            <element type = "ReportSlaveIDReq"/>
            <element type = "ReportSlaveIDRsp"/>
            <element type = "ReadFileRecordReq"/>
            <element type = "ReadFileRecordRsp"/>
            <element type = "WriteFileRecordReq"/>
            <element type = "WriteFileRecordRsp"/>
            <element type = "MaskWriteRegisterReq"/>
            <element type = "MaskWriteRegisterRsp"/>
            <element type = "ReadWriteRegistersReq"/>
            <element type = "ReadWriteRegistersRsp"/>
            <element type = "ReadFifoQueueReq"/>
            <element type = "ReadFifoQueueRsp"/>
            <element type = "EncapsulatedInterfaceTransportReq"/>
            <element type = "EncapsulatedInterfaceTransportRsp"/>
            <element type = "ReadDeviceIdentificationReq"/>
            <element type = "ReadDeviceIdentificationRsp"/>
            <element type = "PrivateModbusReq"/>
            <element type = "PrivateModbusRsp"/>
            <element type = "UnconfirmedPrivateModbusReq"/>
            <element type = "UnconfirmedPrivateModbusRsp"/>
            <element type = "fdt:CommunicationError"/>
        </group>
    </ElementType>

</Schema>
```

10 Channel parameter data types - FDTModbusChannelParameterSchema

The data types specified in this clause are used with the following methods:

- IFdtChannel::GetChannelParameters()
- IFdtChannel::SetChannelParameters()

```
<?xml version = "1.0" encoding = "UTF-8"?>

<Schema name = "FDTModbusChannelParameterSchema"
     xmlns = "urn:schemas-microsoft-com:xml-data"
     xmlns:dt = "urn:schemas-microsoft-com:datatypes"
     xmlns:fdt = "x-schema:FDTDataTypesSchema.xml"
     xmlns:appId = "x-schema:FDTApplicationIdSchema.xml">

    <!--FDT channel parameter schema V1.0 for Modbus protocol -->

    <!--Definition of Attributes-->

    <AttributeType name = "schemaVersion" dt:type = "number" default = "1.0"/>
    <AttributeType name = "address" dt:type = "ui2"/>
    <AttributeType name = "modbusDataTypes"  dt:type = "enumeration" dt:values = "coil discreteInput
holdingRegister inputRegister"/>
    <AttributeType name = "quantity" dt:type = "ui2"/>
```

```
    <AttributeType name = "protectedByChannelAssignment" dt:type = "boolean"/>
    <AttributeType name = "frameApplicationTag" dt:type = "string"/>
    <AttributeType name = "scaleValue" dt:type = "string"/>
   <AttributeType name="statusChannel" dt:type="boolean"/>
   <AttributeType name="gatewayBusCategory" dt:type="uuid"/>

    <!--All elements of the enumeration represent data types specified in IEC 61131-3 -->
    <AttributeType name = "iecDataType"  dt:type = "enumeration" dt:values = "BOOL SINT INT DINT LINT USINT
UINT UDINT ULINT REAL LREAL TIME DATE TimeOfDay DateAndTime STRING BYTE WORD DWORD LWORD
WSTRING"/>

    <!--Definition of Elements-->

    <!--Definition of Modbus access data element-->

    <ElementType name = "ModbusAccessData" content = "empty" model = "closed">
        <attribute type = "fdt:nodeId"/>
        <!-- Starting address of the discrete input/coil/register -->
        <attribute type = "address" required = "yes"/>
        <!-- Type of data to be accessed -->
        <attribute type = "modbusDataTypes" required = "yes"/>
        <!-- Number of discrete inputs/coils/registers to be accessed -->
        <attribute type = "quantity" required = "yes"/>
        <!-- Access right for reading the Modbus data -->
        <attribute type="fdt:readAccess" required="no"/>
        <!-- Access right forwriting the Modbus data -->
        <attribute type="fdt:writeAccess" required="no"/>
    </ElementType>

    <!--Definition of UnitScaling element-->
    <ElementType name = "UnitScaling" content = "empty" model = "closed">
        <attribute type = "scaleValue" required = "yes"/>
    </ElementType>

    <ElementType name = "FDTChannel" content = "eltOnly" order = "seq" model = "closed">

        <attribute type = "fdt:nodeId"/>
        <!-- Unique identifier for a device, module or channel -->
        <attribute type = "fdt:tag" required = "yes"/>
        <!-- Unique identifier for an element within the device namespace -->
        <attribute type = "fdt:id" required = "yes"/>
        <!-- Human readable description within the context of a element -->
        <attribute type = "fdt:descriptor"/>
        <!-- TRUE if the channels is set to read only by the Frame Application -->
        <attribute type = "protectedByChannelAssignment" required = "yes"/>
        <!-- Standard FDT data type -->
        <attribute type = "fdt:dataType" required = "yes"/>
        <!-- Standard IEC 61131-3 data types -->
        <attribute type = "iecDataType" required = "no"/>
        <!-- Specifies a signal as input or output -->
        <attribute type = "fdt:signalType" required = "yes"/>
        <!-- Frame Application specific tag used for identification and navigation. -->
        <!-- The DTM should display this tag at channel specific user interfaces -->
        <attribute type = "frameApplicationTag"/>
        <!-- The appearance and the functionality of a DTM user interface is controlled by the  -->
        <!-- entry of the element applicationId, functionId, and operationPhase -->
        <attribute type = "appId:applicationId"/>
        <!-- TODO SemanticIDs need to be defined  -->
        <element type = "fdt:SemanticInformation" minOccurs = "0" maxOccurs = "*"/>
        <!-- Collection of EnumerationEntry  -->
        <element type = "fdt:BitEnumeratorEntries" minOccurs = "0" maxOccurs = "1"/>
        <!-- Enumeration element  -->
        <element type = "fdt:EnumeratorEntries" minOccurs = "0" maxOccurs = "1"/>
        <!-- Current unit and the collection of possible units of a process variable  -->
        <element type = "fdt:Unit" minOccurs = "0" maxOccurs = "1"/>
        <!-- scale value for the grapical representation of the process value -->
        <element type = "UnitScaling" minOccurs = "0" maxOccurs = "1"/>
        <!-- Address information needed to directly access the process data in the target device via Modbus  -->
        <element type = "ModbusAccessData" minOccurs = "0" maxOccurs = "1"/>
        <!-- Collection of alarms specified in FDT  -->
```

```
        <element type="fdt:Alarms" minOccurs="0" maxOccurs="1"/>
        <!-- Collection of ranges specified in FDT, which describe the valid range of a process value  -->
        <element type="fdt:Ranges" minOccurs="0" maxOccurs="1"/>
        <!-- Deadband is the amount of value changes that triggers for example new trend values.-->
        <element type = "fdt:Deadband" minOccurs = "0" maxOccurs = "1"/>
        <!-- Describes a substitute value which is used in combination of the behavior of disturbed channel values --
>
        <element type="fdt:SubstituteValue" minOccurs="0" maxOccurs="1"/>
        <!-- should be used if the data type is structured -->
        <element type = "fdt:StructuredElements" minOccurs = "0" maxOccurs = "1"/>
    </ElementType>

    <ElementType name = "FDTChannelType" content = "eltOnly" order = "seq" model = "closed">
        <attribute type = "fdt:nodeId"/>
        <element type = "fdt:VersionInformation"/>
      <!-- Unique identifier for a supported bus type like Profibus or HART according to the FDT specific CATID -->
      <attribute type="gatewayBusCategory" required="no"/>
      <!-- TRUE if the channel is for status information only -->
      <attribute type="statusChannel" required="no"/>
    </ElementType>

    <ElementType name = "FDT" content = "eltOnly" order = "seq" model = "closed">
        <attribute type = "schemaVersion"/>
        <attribute type = "fdt:nodeId"/>
        <element type = "FDTChannelType" minOccurs="1" maxOccurs="1"/>
        <element type = "FDTChannel" minOccurs="1" maxOccurs="1"/>
    </ElementType>
</Schema>
```

11 Device identification

11.1 Device type identification data types - FDTModbusIdentSchema

```
<?xml version = "1.0" encoding = "UTF-8"?>

<Schema name = "FDTModbusIdentSchema"
     xmlns = "urn:schemas-microsoft-com:xml-data"
     xmlns:dt = "urn:schemas-microsoft-com:datatypes">

    <!-- FDT ModbusIdent schema V1.0 for Modbus protocol -->
    <!-- The  FDT ModbusIdent schema contains the definition of the attributes for the optional Modbus
identification objects (Regular Category)  -->
    <!-- The Modbus identification objects can be read with the standard Modbus service ReadDeviceIdentification -
->

    <!--Definition of general Fdt attributes-->

    <AttributeType name = "schemaVersion" dt:type = "number" default = "1.0"/>
    <AttributeType name = "protocolName" dt:type = "string" default = "protocol_Modbus"/>
    <AttributeType name = "idDTMSupportLevel" dt:type = "enumeration" dt:values = "genericSupport
profileSupport blockspecificProfileSupport specificSupport identSupport"/>

    <!-- The Modbus identification objects are divided into two categories: -->

    <!-- - Basic category: -->
    <!--   Mandatory Modbus identification objects to required (mandatory) FDT identification elements -->

    <AttributeType name = "vendorName" dt:type = "string"/>
    <AttributeType name = "productCode" dt:type = "string"/>
    <AttributeType name = "majorMinorRevision" dt:type = "string"/>

    <!-- - Regular category: -->
    <!--   Optional Modbus identification objects to the optional FDT identification element IdValue-->
    <!--   Each Modbus identification object of the regular category is described with the following attributes -->

    <!-- name: Name of the Modbus identification object -->
    <AttributeType name = "name" dt:type = "enumeration" dt:values = "ProductName ModelName VendorURL
UserApplicationName"/>
```

```
    <!-- protocolSpecificName:Protocol specific name of the Modbus identification object (equals name) -->
    <AttributeType name = "protocolSpecificName" dt:type = "enumeration" dt:values = "ProductName ModelName
VendorURL UserApplicationName"/>
    <!-- value: Content of the Modbus identification object -->
    <AttributeType name = "value" dt:type = "string"/>


    <!-- Regular Expression attributes-->
    <AttributeType name = "match" dt:type = "string"/>
    <AttributeType name = "nomatch" dt:type = "string"/>


    <!-- Regular Expression element-->
    <ElementType name = "RegExpr" content = "empty" model = "closed">
        <attribute type = "match"/>
        <attribute type = "nomatch"/>
    </ElementType>
</Schema>
```

11.2 Topology scan data types - DTMModbusDeviceSchema

Used at IDtmEvents::OnScanResponse()

```
<?xml version = "1.0" encoding = "UTF-8"?>

<Schema name="DTMModbusDeviceSchema"
    xmlns="urn:schemas-microsoft-com:xml-data"
    xmlns:fdt="x-schema:FDTDataTypesSchema.xml"
    xmlns:dt="urn:schemas-microsoft-com:datatypes"
    xmlns:mb = "x-schema:FDTModbusAddressSchema.xml">

    <!--FDT V1.2 device schema for Modbus protocol V1.0 -->
    <!-- The  FDT DTMModbusDevice schema defines the mapping of the Modbus identification objects -->
    <!-- to standard FDT identification elements for FDT V1.2 -->
    <!-- The Modbus identification objects can be read with the standard Modbus service ReadDeviceIdentification -
->

    <!--Definition of general Fdt attributes-->
    <AttributeType name = "schemaVersion" dt:type = "number" default = "1.0"/>

    <!-- Definition of general FDT elements which contain the information about the applied protocol -->
    <!-- and the version of the protocol -->
    <AttributeType name = "protocolName" dt:type = "string" default = "protocol_Modbus"/>

    <!-- The mapping of Modbus identification objects to FDT identification elements is divided into two categories:
-->

    <!-- - Basic category: -->
    <!--   Mapping of mandatory Modbus identification objects to required (mandatory) FDT identification elements -
->

    <!--Modbus identification object VendorName -->
    <AttributeType name = "vendorName" dt:type = "string"/>
    <!--Modbus identification object ProductCode  -->
    <AttributeType name = "productCode" dt:type = "string"/>
    <!--Modbus identification object MajorMinorRevision  -->
    <AttributeType name = "majorMinorRevision" dt:type = "string"/>

    <!-- - Regular category: -->
    <!--   Mapping of optional Modbus identification objects to the optional FDT identification element IdValue-->
    <AttributeType name = "name" dt:type = "enumeration" dt:values = "ProductName ModelName VendorURL
UserApplicationName"/>
    <AttributeType name = "protocolSpecificName" dt:type = "enumeration" dt:values = "ProductName ModelName
VendorURL UserApplicationName"/>
    <AttributeType name = "value" dt:type = "string"/>

    <!-- Basic category: -->
```

```
    <!-- Mapping of Modbus identification elements required by the FDT DTMDeviceTypeIdent schema-->
    <!-- These elements represent all mandatory objects for Modbus device identification (basic category) -->

    <!--Modbus identification object VendorName is mapped to FDT element IdManufacturer -->
    <ElementType name = "IdManufacturer" content = "empty"  model = "closed">
        <attribute type = "vendorName" required="yes"/>
    </ElementType>

    <!--Modbus identification object ProductCode is mapped to FDT element IdTypeID -->
    <ElementType name = "IdTypeID" content = "empty"  model = "closed">
        <attribute type = "productCode" required="yes"/>
    </ElementType>

    <!--Modbus identification object MajorMinorRevision is mapped to FDT element IdSoftwareRevision -->
    <ElementType name = "IdSoftwareRevision" content = "empty"  model = "closed">
        <attribute type = "majorMinorRevision" required="yes"/>
    </ElementType>



    <!--Definition Modbus identification element not mandatory by the FDT DTMDeviceTypeIdent schema-->
    <!--This element represent all  objects for Modbus device identification (regular category) -->


    <ElementType name="IdValue" content="empty" model="closed">
      <attribute type="name" required="yes"/>
        <attribute type="value" required="yes"/>
        <attribute type="protocolSpecificName" required="yes"/>
    </ElementType>

    <ElementType name="IdValues" content="eltOnly" model="closed">
        <element type="IdValue" minOccurs="0" maxOccurs="*"/>
    </ElementType>

    <ElementType name="ModbusDevice" content="eltOnly" model="closed">
        <attribute type="fdt:nodeId" required="no"/>
        <attribute type="schemaVersion" required="no"/>
        <attribute type="protocolName" required="yes"/>
        <!-- Address information for Modbus TCP and Modbus Serial Line Devices  -->
        <group order = "one" maxOccurs="1" minOccurs="1">
            <element type = "mb:ModbusSerial"/>
            <element type = "mb:ModbusTCP"/>
        </group>
        <element type = "IdManufacturer" minOccurs="1" maxOccurs="1"/>
        <element type = "IdTypeID" minOccurs="1" maxOccurs="1"/>
        <element type = "IdSoftwareRevision" minOccurs="1" maxOccurs="1"/>
        <element type = "IdValues" minOccurs = "0" maxOccurs = "1"/>
    </ElementType>

</Schema>
```

11.3 Scan identification data types - FDTModbusScanIdentSchema

This schema defines the XML document provided by a scan response of a Modbus network.

```
<?xml version = "1.0" encoding = "UTF-8"?>

<Schema name="DTMModbusDeviceSchema"
    xmlns="urn:schemas-microsoft-com:xml-data"
    xmlns:fdt="x-schema:FDTDataTypesSchema.xml"
    xmlns:dt="urn:schemas-microsoft-com:datatypes"
    xmlns:mb = "x-schema:FDTModbusAddressSchema.xml">

    <!--FDT V1.2 device schema for Modbus protocol V1.0 -->
    <!-- The  FDT DTMModbusDevice schema defines the mapping of the Modbus identification objects -->
    <!-- to standard FDT identification elements for FDT V1.2 -->
```

```
    <!-- The Modbus identification objects can be read with the standard Modbus service ReadDeviceIdentification -
->

    <!--Definition of general Fdt attributes-->
    <AttributeType name = "schemaVersion" dt:type = "number" default = "1.0"/>

    <!-- Definition of general FDT elements which contain the information about the applied protocol -->
    <!-- and the version of the protocol -->
    <AttributeType name = "protocolName" dt:type = "string" default = "protocol_Modbus"/>

    <!-- The mapping of Modbus identification objects to FDT identification elements is divided into two categories:
-->

    <!-- - Basic category: -->
    <!--   Mapping of mandatory Modbus identification objects to required (mandatory) FDT identification elements -
->

    <!--Modbus identification object VendorName -->
    <AttributeType name = "vendorName" dt:type = "string"/>
    <!--Modbus identification object ProductCode  -->
    <AttributeType name = "productCode" dt:type = "string"/>
    <!--Modbus identification object MajorMinorRevision  -->
    <AttributeType name = "majorMinorRevision" dt:type = "string"/>

    <!-- - Regular category: -->
    <!--   Mapping of optional Modbus identification objects to the optional FDT identification element IdValue-->
    <AttributeType name = "name" dt:type = "enumeration" dt:values = "ProductName ModelName VendorURL
UserApplicationName"/>
    <AttributeType name = "protocolSpecificName" dt:type = "enumeration" dt:values = "ProductName ModelName
VendorURL UserApplicationName"/>
    <AttributeType name = "value" dt:type = "string"/>

    <!-- Basic category: -->
    <!-- Mapping of Modbus identification elements required by the FDT DTMDeviceTypeIdent schema-->
    <!-- These elements represent all mandatory objects for Modbus device identification (basic category) -->

    <!--Modbus identification object VendorName is mapped to FDT element IdManufacturer -->
    <ElementType name = "IdManufacturer" content = "empty"  model = "closed">
        <attribute type = "vendorName" required="yes"/>
    </ElementType>

    <!--Modbus identification object ProductCode is mapped to FDT element IdTypeID -->
    <ElementType name = "IdTypeID" content = "empty"  model = "closed">
        <attribute type = "productCode" required="yes"/>
    </ElementType>

    <!--Modbus identification object MajorMinorRevision is mapped to FDT element IdSoftwareRevision -->
    <ElementType name = "IdSoftwareRevision" content = "empty"  model = "closed">
        <attribute type = "majorMinorRevision" required="yes"/>
    </ElementType>



    <!--Definition Modbus identification element not mandatory by the FDT DTMDeviceTypeIdent schema-->
    <!--This element represent all  objects for Modbus device identification (regular category) -->


    <ElementType name="IdValue" content="empty" model="closed">
      <attribute type="name" required="yes"/>
        <attribute type="value" required="yes"/>
        <attribute type="protocolSpecificName" required="yes"/>
    </ElementType>

    <ElementType name="IdValues" content="eltOnly" model="closed">
        <element type="IdValue" minOccurs="0" maxOccurs="*"/>
    </ElementType>

    <ElementType name="ModbusDevice" content="eltOnly" model="closed">
        <attribute type="fdt:nodeId" required="no"/>
        <attribute type="schemaVersion" required="no"/>
```

```
        <attribute type="protocolName" required="yes"/>
        <!-- Address information for Modbus TCP and Modbus Serial Line Devices  -->
        <group order = "one" maxOccurs="1" minOccurs="1">
            <element type = "mb:ModbusSerial"/>
            <element type = "mb:ModbusTCP"/>
        </group>
        <element type = "IdManufacturer" minOccurs="1" maxOccurs="1"/>
        <element type = "IdTypeID" minOccurs="1" maxOccurs="1"/>
        <element type = "IdSoftwareRevision" minOccurs="1" maxOccurs="1"/>
        <element type = "IdValues" minOccurs = "0" maxOccurs = "1"/>
    </ElementType>

</Schema>
```

11.4 Device type identification data types - FDTModbusDeviceTypeIdentSchema

This subclause defines data types that are used to protocol specific information for device types.

```
<?xml version = "1.0" encoding = "UTF-8"?>

<Schema name = "FDTModbusDeviceTypeIdentSchema"
     xmlns = "urn:schemas-microsoft-com:xml-data"
     xmlns:dt = "urn:schemas-microsoft-com:datatypes"
     xmlns:modbusident = "x-schema:FDTModbusIdentSchema.xml"
     xmlns:fdt = "x-schema:FDTDataTypesSchema.xml">

    <!-- FDT DeviceTypeIdent schema V1.0 for Modbus protocol -->
    <!-- The  FDT DeviceTypeIdent schema defines the mapping of the Modbus identification objects -->
    <!-- to standard FDT identification elements -->
    <!-- The Modbus identification objects can be read with the standard Modbus service ReadDeviceIdentification -
->

    <!-- The mapping of Modbus identification objects to FDT identification elements is divided into two categories:
-->

    <!-- - Basic category: -->
    <!--   Mapping of mandatory Modbus identification objects to required (mandatory) FDT identification elements -
->

    <!-- - Regular category: -->
    <!--   Mapping of optional Modbus identification objects to the optional FDT identification element IdValue-->


    <!-- Definition of general FDT attributes-->
    <AttributeType name = "schemaVersion" dt:type = "number" default = "1.0"/>

    <!-- Definition of general FDT elements which contain the information about the applied protocol -->
    <!-- and the version of the protocol -->

    <ElementType name = "IdBusProtocol" content = "empty"  model = "closed">
        <attribute type = "modbusident:protocolName" required="yes"/>
    </ElementType>

    <!-- Basic category: -->
    <!-- Mapping of Modbus identification elements required by the FDT DTMDeviceTypeIdent schema-->
    <!-- These elements represent all mandatory objects for Modbus device identification (basic category) -->

    <!--Modbus identification object VendorName is mapped to FDT element IdManufacturer -->
    <ElementType name = "IdManufacturer" content = "eltOnly"  model = "closed">
        <attribute type = "modbusident:vendorName" required="yes"/>
        <element type = "modbusident:RegExpr" minOccurs = "0" maxOccurs = "*"/>
    </ElementType>

    <!--Modbus identification object ProductCode is mapped to FDT element IdTypeID -->
    <ElementType name = "IdTypeID" content = "eltOnly"  model = "closed">
        <attribute type = "modbusident:productCode" required="yes"/>
        <element type = "modbusident:RegExpr" minOccurs = "0" maxOccurs = "*"/>
    </ElementType>
```

```
    <!--Modbus identification object MajorMinorRevision is mapped to FDT element IdSoftwareRevision -->
    <ElementType name = "IdSoftwareRevision" content = "eltOnly"  model = "closed">
        <attribute type = "modbusident:majorMinorRevision" required="yes"/>
        <element type = "modbusident:RegExpr" minOccurs = "0" maxOccurs = "*"/>
    </ElementType>


    <!-- Regular category: -->
    <!-- Mapping of Modbus identification elements not required by the FDT DTMDeviceTypeIdent schema-->
    <!-- The information provided by each optional Modbus identification object (regular category) is mapped  -->
    <!-- to the standard FDT identification element IdValues, which may occur several times -->

    <!-- The following optional Modbus identification objects are supported: -->

    <!-- VendorUrl                Attribute name/Protocol specific name:"vendorUrl"                -->
    <!-- ProductName              Attribute name/Protocol specific name:"productName"              -->
    <!-- ModelName                Attribute name/Protocol specific name:"modelName"                -->
    <!-- UserApplicationName      Attribute name/Protocol specific name:"userApplicationName"    -->

    <ElementType name="IdValue" content="eltOnly" model="closed">
        <attribute type="modbusident:name" required="yes"/>
        <attribute type="modbusident:value" required="no"/>
        <attribute type="modbusident:protocolSpecificName" required="yes"/>
        <element type="modbusident:RegExpr" minOccurs="0" maxOccurs="*"/>
    </ElementType>

    <ElementType name="IdValues" content="eltOnly" model="closed">
        <element type="IdValue" minOccurs="0" maxOccurs="*"/>
    </ElementType>

    <!--FDT DeviceIdentification element -->

    <ElementType name = "DeviceIdentification" content = "eltOnly" model = "closed">
        <attribute type = "modbusident:idDTMSupportLevel" required="yes"/>
        <element type = "IdBusProtocol" minOccurs="1" maxOccurs="1"/>
        <element type = "IdManufacturer" minOccurs="1" maxOccurs="1"/>
        <element type = "IdTypeID" minOccurs="1" maxOccurs="1"/>
        <element type = "IdSoftwareRevision" minOccurs="1" maxOccurs="1"/>
        <element type = "IdValues" minOccurs = "0" maxOccurs = "1"/>
    </ElementType>

    <ElementType name = "DeviceIdentifications" content = "eltOnly" model = "closed">
        <element type = "DeviceIdentification" minOccurs = "1" maxOccurs = "*"/>
    </ElementType>

    <ElementType name = "FDT" content = "eltOnly" model = "closed">
        <element type = "DeviceIdentifications"/>
    </ElementType>

</Schema>
```

11.5 XSLT Transformation

```
<?xml version="1.0" encoding="UTF-8"?>
<!--FDT XSL transformation for Modbus protocol V1.0-->
<!--This XSL transformation defines the transformation of an XML document with Modbus specific parameters to a
protocol independent -->
<!--XML document as it is required in FDT for the device identification and the scan -->

<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0" xmlns:str="http://xsltsl.org/string"
xmlns:ident="x-schema:DTMIdentSchema.xml" xmlns:fdt="x-schema:FDTDataTypesSchema.xml"
xmlns:scanident="x-schema:DTMScanIdentSchema.xml" xmlns:devident="x-
schema:DTMDeviceTypeIdentSchema.xml" xmlns:modbusdevice="x-
schema:FDTModbusDeviceTypeIdentSchema.xml" xmlns:modbusident="x-schema:FDTModbusIdentSchema.xml"
xmlns:modbusscan="x-schema:FDTModbusScanIdentSchema.xml" xmlns:mb="x-
schema:FDTModbusAddressSchema.xml" exclude-result-prefixes="str modbusdevice modbusident modbusscan
mb">
    <!-- Definition of transformation output format -->
```

```
    <xsl:output method="xml" indent="no" encoding="UTF-8"/>
    <xsl:variable name="FileVersion" select="1.0"/>
    <!-- select root element FDT -->
    <xsl:template match="/">

            <!-- check if the XML document which shall be transfomed contains the information -->
            <!-- of a device identification or of a bus scan -->
            <xsl:apply-templates select="//modbusscan:ScanIdentifications"/>
            <xsl:apply-templates select="//modbusdevice:DeviceIdentifications"/>

    </xsl:template>
    <!-- the XML document contains the information for a device identification -->
    <xsl:template match="modbusdevice:DeviceIdentifications">

        <xsl:element name="devident:FDT">
           <xsl:element name="devident:DeviceIdentifications">
            <xsl:apply-templates select="modbusdevice:DeviceIdentification"/>
           </xsl:element>
        </xsl:element>
    </xsl:template>

    <!-- the XML document contains the information of a bus scan -->
    <xsl:template match="modbusscan:ScanIdentifications">
        <xsl:element name="scanident:FDT">
           <xsl:element name="scanident:ScanIdentifications">
               <xsl:attribute name="fdt:busCategory"><xsl:value-of select="./@fdt:busCategory"/></xsl:attribute>
               <xsl:attribute name="resultState"><xsl:value-of select="./@resultState"/></xsl:attribute>
               <xsl:apply-templates select="modbusscan:ScanIdentification"/>
           </xsl:element>
        </xsl:element>
    </xsl:template>


    <!-- ****************************************  transformation of the bus scan information
***********************************************-->
    <xsl:template match="modbusscan:ScanIdentification">
        <xsl:element name="scanident:ScanIdentification">
            <xsl:apply-templates select="./fdt:CommunicationError"/>

           <!-- used bus protocol -->
        <xsl:element name="scanident:IdBusProtocol">
            <xsl:attribute name="ident:protocolSpecificName"><xsl:text>Modbus</xsl:text></xsl:attribute>
            <xsl:attribute name="ident:value"><xsl:value-of
select="modbusscan:IdBusProtocol/@modbusident:protocolName"/></xsl:attribute>
        </xsl:element>

        <!-- bus protocol version -->
           <xsl:element name="scanident:IdBusProtocolVersion">
            <xsl:attribute name="ident:protocolSpecificName"><xsl:text>not applicable</xsl:text></xsl:attribute>
        </xsl:element>

        <!-- Modbus specific addressing depending on the physical layer -->
        <!-- Check if ModbusSerial element exists -->
        <xsl:apply-templates select=".//mb:ModbusSerial"/>

           <!-- Check if ModbusTCP element exists -->
        <xsl:apply-templates select=".//mb:ModbusTCP"/>
        <!-- device specific information received with ReadDeviceIdentification service -->
        <!-- mandatory identification objects specified in the Modbus application protocol specification (basic
category) -->

           <!-- Vendor Name -->
           <xsl:element name="scanident:IdManufacturer">
               <xsl:attribute name="ident:protocolSpecificName"><xsl:text>Vendor Name</xsl:text></xsl:attribute>
               <xsl:attribute name="ident:value"><xsl:value-of
select="modbusscan:IdManufacturer/@modbusident:vendorName"/></xsl:attribute>
           </xsl:element>

           <!-- Product Code -->
        <xsl:element name="scanident:IdTypeID">
```

```
            <xsl:attribute name="ident:protocolSpecificName"><xsl:text>Product Code</xsl:text></xsl:attribute>
                <xsl:attribute name="ident:value"><xsl:value-of
select="modbusscan:IdTypeID/@modbusident:productCode"/></xsl:attribute>
        </xsl:element>

            <!-- MajorMinor Revision -->
            <xsl:element name="scanident:IdSoftwareRevision">
                <xsl:attribute
name="ident:protocolSpecificName"><xsl:text>MajorMinorRevision</xsl:text></xsl:attribute>
                <xsl:attribute name="ident:value"><xsl:value-of
select="modbusscan:IdSoftwareRevision/@modbusident:majorMinorRevision"/></xsl:attribute>
            </xsl:element>

        <!-- not applicable elements -->
            <xsl:element name="scanident:IdHardwareRevision">
                <xsl:attribute name="ident:protocolSpecificName"><xsl:text>not applicable</xsl:text></xsl:attribute>
        </xsl:element>

            <xsl:element name="scanident:IdDeviceTag">
                <xsl:attribute name="ident:protocolSpecificName"><xsl:text>not applicable</xsl:text></xsl:attribute>
            </xsl:element>

            <xsl:element name="scanident:IdSerialNumber">
                <xsl:attribute name="ident:protocolSpecificName"><xsl:text>not applicable</xsl:text></xsl:attribute>
            </xsl:element>

            <!-- optional identification objects specified in the Modbus application protocol specification (regular
category) -->
            <xsl:element name="scanident:IdValues">
                <xsl:apply-templates select="./modbusscan:IdValues"/>
            </xsl:element>
      </xsl:element>
    </xsl:template>

   <!-- template for fdt:CommunicationError element -->
    <xsl:template match="fdt:CommunicationError">
      <xsl:copy-of select="."/>
    </xsl:template>

    <!-- get address for Modbus Serial Line -->
    <xsl:template match="mb:ModbusSerial">
        <xsl:element name="scanident:IdAddress">
            <xsl:attribute name="ident:protocolSpecificName"><xsl:text>ModbusSerial</xsl:text></xsl:attribute>
            <xsl:attribute name="ident:value"><xsl:value-of select="@slaveAddress"/></xsl:attribute>
        </xsl:element>
    </xsl:template>

    <!-- get address for ModbusTCP -->
    <xsl:template match="mb:ModbusTCP">
        <xsl:element name="scanident:IdAddress">
            <xsl:attribute name="ident:protocolSpecificName"><xsl:text>ModbusTCP</xsl:text></xsl:attribute>
            <xsl:attribute name="ident:value"><xsl:value-of select="@tcpAddress"/></xsl:attribute>
        </xsl:element>
    </xsl:template>

    <!--check if optional identification objects are  provided (regular category) -->
    <xsl:template match="modbusscan:IdValues">
        <xsl:apply-templates select="./modbusscan:IdValue"/>
    </xsl:template>

    <!-- transformation of  optional identification objects (regular category) -->
    <xsl:template match="modbusscan:IdValue">
        <xsl:element name="scanident:IdValue">
            <xsl:call-template name="genScanName">
                <xsl:with-param name="name" select="@modbusident:name"/>
            </xsl:call-template>
            <xsl:call-template name="genScanMatch">
                <xsl:with-param name="value" select="@modbusident:value"/>
            </xsl:call-template>
            <xsl:call-template name="genScanProtocolSpecificName">
```

```
                <xsl:with-param name="protocolSpecificName" select="@modbusident:protocolSpecificName"/>
            </xsl:call-template>
        </xsl:element>
    </xsl:template>

    <xsl:template name="genScanName">
      <xsl:param name="name"/>
        <xsl:if test="$name">
            <xsl:attribute name="ident:name"><xsl:value-of select="$name"/></xsl:attribute>
        </xsl:if>
    </xsl:template>

    <xsl:template name="genScanMatch">
        <xsl:param name = "value"/>
        <xsl:if test = "$value">
            <xsl:attribute name = "ident:value">
                <xsl:value-of select = "$value"/>
            </xsl:attribute>
        </xsl:if>
    </xsl:template>

    <xsl:template name="genScanProtocolSpecificName">
        <xsl:param name="protocolSpecificName"/>
        <xsl:if test="$protocolSpecificName">
            <xsl:attribute name="ident:protocolSpecificName"><xsl:value-of
select="$protocolSpecificName"/></xsl:attribute>
        </xsl:if>
    </xsl:template>


    <!-- ****************************************  transformation of the device identification information
************************************************-->
    <xsl:template match="modbusdevice:DeviceIdentification">
        <xsl:element name="devident:DeviceIdentification">
            <xsl:attribute name="ident:idDTMSupportLevel"><xsl:value-of
select="@modbusident:idDTMSupportLevel"/></xsl:attribute>

          <!-- used bus protocol -->
            <xsl:element name="devident:IdBusProtocol">
                <xsl:attribute name="ident:protocolSpecificName"><xsl:text>Modbus</xsl:text></xsl:attribute>
                <xsl:attribute name="ident:value"><xsl:value-of
select="modbusdevice:IdBusProtocol/@modbusident:protocolName"/></xsl:attribute>
            </xsl:element>

          <!-- bus protocol version -->
            <xsl:element name="devident:IdBusProtocolVersion">
            <xsl:attribute name="ident:protocolSpecificName"><xsl:text>not applicable</xsl:text></xsl:attribute>
            </xsl:element>

          <!-- device specific information received with ReadDeviceIdentification service -->
            <!-- mandatory identification objects specified in the Modbus application protocol specification (basic
category) -->

          <xsl:apply-templates select="modbusdevice:IdManufacturer"/>
          <xsl:apply-templates select="modbusdevice:IdTypeID"/>
          <xsl:apply-templates select="modbusdevice:IdSoftwareRevision"/>

          <!-- not applicable element -->
            <xsl:element name="devident:IdHardwareRevision">
                <xsl:attribute name="ident:protocolSpecificName"><xsl:text>not applicable</xsl:text></xsl:attribute>
            </xsl:element>

          <!-- optional identification objects specified in the Modbus application protocol specification (regular
category) -->
            <xsl:element name="devident:IdValues">
                <xsl:apply-templates select="./modbusdevice:IdValues"/>
            </xsl:element>
        </xsl:element>
    </xsl:template>
```

```
   <!-- Vendor Name -->
   <xsl:template match="modbusdevice:IdManufacturer">
      <xsl:element name="devident:IdManufacturer">
            <xsl:attribute name="ident:protocolSpecificName"><xsl:text>Vendor Name</xsl:text></xsl:attribute>
            <xsl:attribute name="ident:value"><xsl:value-of select="@modbusident:vendorName"/></xsl:attribute>
            <xsl:call-template name="genMatch">
               <xsl:with-param select="modbusdevice:IdManufacturer" name="value" />
            </xsl:call-template>
      </xsl:element>
   </xsl:template>

   <!-- Product Code -->
   <xsl:template match="modbusdevice:IdTypeID">
        <xsl:element name="devident:IdTypeID">
         <xsl:attribute name="ident:protocolSpecificName"><xsl:text>Product Code</xsl:text></xsl:attribute>
         <xsl:attribute name="ident:value"><xsl:value-of select="@modbusident:productCode"/></xsl:attribute>
         <xsl:call-template name="genMatch" >
            <xsl:with-param select="modbusdevice:IdTypeID" name="value" />
         </xsl:call-template>
            </xsl:element>
   </xsl:template>


   <!-- MajorMinor Revision -->
   <xsl:template match="modbusdevice:IdSoftwareRevision">
      <xsl:element name="devident:IdSoftwareRevision">
         <xsl:attribute name="ident:protocolSpecificName"><xsl:text>MajorMinorRevision</xsl:text></xsl:attribute>
         <xsl:attribute name="ident:value"><xsl:value-of
select="@modbusident:majorMinorRevision"/></xsl:attribute>
         <xsl:call-template name="genMatch">
            <xsl:with-param select="modbusdevice:IdSoftwareRevision" name="value" />
      </xsl:call-template>
      </xsl:element>
   </xsl:template>


    <!--check if optional identification objects are  provided (regular category) -->
    <xsl:template match="modbusdevice:IdValues">
        <xsl:apply-templates select="./modbusdevice:IdValue"/>
    </xsl:template>

    <!-- transformation of  optional identification objects (regular category) -->
    <xsl:template match="modbusdevice:IdValue">
        <xsl:element name="devident:IdValue">
            <xsl:call-template name="genName">
                <xsl:with-param name="name" select="@modbusident:name"/>
            </xsl:call-template>
            <xsl:if test="@modbusident:value">
                <xsl:attribute name="ident:value"><xsl:value-of select="@modbusident:value"/></xsl:attribute>
            </xsl:if>
            <xsl:call-template name="genProtocolSpecificName">
                <xsl:with-param name="protocolSpecificName" select="@modbusident:protocolSpecificName"/>
            </xsl:call-template>
            <xsl:call-template name="genMatch">
                <xsl:with-param select="modbusdevice:IdValue" name="value" />
            </xsl:call-template>
        </xsl:element>
    </xsl:template>

    <xsl:template name="genName">
        <xsl:param name="name"/>
        <xsl:if test="$name">
            <xsl:attribute name="ident:name"><xsl:value-of select="$name"/></xsl:attribute>
        </xsl:if>
    </xsl:template>

    <xsl:template name="genProtocolSpecificName">
        <xsl:param name="protocolSpecificName"/>
        <xsl:if test="$protocolSpecificName">
```

```
            <xsl:attribute name="ident:protocolSpecificName"><xsl:value-of
select="$protocolSpecificName"/></xsl:attribute>
        </xsl:if>
    </xsl:template>

    <xsl:template name="genMatch">
        <xsl:param name="value"/>
         <xsl:apply-templates select="modbusident:RegExpr"/>
    </xsl:template>

    <xsl:template match="modbusident:RegExpr">
        <!-- copy pattern info -->
        <xsl:call-template name="genPattern">
            <xsl:with-param name="match" select="@match"/>
            <xsl:with-param name="nomatch" select="@nomatch"/>
        </xsl:call-template>
    </xsl:template>

    <!-- generation of the pattern for the regular expressions -->
    <!-- These regular expressions can be used by a DTM in order to define a range of supported physical device
types  -->
    <xsl:template name="genPattern">
        <xsl:param name="match"/>
        <xsl:param name="nomatch"/>
        <xsl:choose>
            <xsl:when test="$match">
                <xsl:element name="ident:RegExpr">
                    <xsl:attribute name="match"><xsl:value-of select="$match"/></xsl:attribute>
                    <xsl:if test="$nomatch">
                        <xsl:attribute name="nomatch"><xsl:value-of select="$nomatch"/></xsl:attribute>
                    </xsl:if>
                </xsl:element>
            </xsl:when>
            <xsl:when test="$nomatch">
                <xsl:element name="ident:RegExpr">
                    <xsl:attribute name="nomatch"><xsl:value-of select="$nomatch"/></xsl:attribute>
                    <xsl:if test="$match">
                        <xsl:attribute name="match"><xsl:value-of select="$match"/></xsl:attribute>
                    </xsl:if>
                </xsl:element>
            </xsl:when>
        </xsl:choose>
    </xsl:template>
</xsl:transform>
```

Bibliography

[1] *FDT Interface Specification V1.2*, Order No. of FDT Joint Interest Group: 0001-0001-001

[2] *FDT Interface Specification V1.2.1*, Order No. of FDT Group: 0001-0001-002

[3] IEC 61158-5-15, *Industrial communication networks – Fieldbus specifications – Part 5-15: Application layer service definition – Type 15 elements*

_____

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

3, rue de Varembé
PO Box 131
CH-1211 Geneva 20
Switzerland

Tel: + 41 22 919 02 11
Fax: + 41 22 919 03 00
info@iec.ch
www.iec.ch