INTERNATIONAL STANDARD

ISO 20242-4

First edition
2011-12-15

Industrial automation systems and integration — Service interface for testing applications —

Part 4: Device capability profile template

Systèmes d'automatisation industrielle et intégration — Interface de service pour contrôler les applications —

Partie 4: Modèle de profil de capacité de dispositif

Reference number
ISO 20242-4:2011(E)

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 20242-4 was prepared by Technical Committee ISO/TC 184, *Automation systems and integration*, Subcommittee SC 5, *Interoperability, integration, and architectures for enterprise systems and automation applications*.

ISO 20242 consists of the following parts, under the general title *Industrial automation systems and integration — Service interface for testing applications*:

— *Part 1: Overview*

— *Part 2: Resource management service interface*

— *Part 3: Virtual device service interface*

— *Part 4: Device capability profile template*

The following parts are under preparation:

— *Part 5: Application program service interface*

— *Part 6: Conformance test methods, criteria and reports*

Introduction

The motivation for ISO 20242 stems from the desire of international automotive industries and their suppliers to facilitate the integration of automation and measurement devices, and other peripheral components for this purpose, into computer-based applications. ISO 20242 defines rules for the construction of device drivers and their behaviour in the context of an automation and/or measurement application.

The main goal of ISO 20242 is to provide users with:

— independence from the computer operating system;

— independence from the device connection technology (device interface/network);

— independence from device suppliers;

— the ability to ensure compatibility between device drivers and connected devices, and their behaviour in the context of a given computer platform;

— independence from the technological device development in the future.

ISO 20242 does not necessitate the development of new device families or the use of special interface technologies (networks). It encapsulates a device and its communication interface to make it compatible with other devices of that kind for a given application.

Industrial automation systems and integration — Service interface for testing applications —

Part 4: Device capability profile template

1 Scope

This part of ISO 20242 defines the formatting, syntax and semantic rules for describing

— device and coordinator capabilities with XML schema, and

— the configuration of devices with XML.

NOTE This part of ISO 20242 does not address the coordinator configuration, which will be addressed elsewhere in the ISO 20242 series.

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.

ISO 15745-1, *Industrial automation systems and integration — Open systems application integration framework — Part 1: Generic reference description*

ISO 20242-1, *Industrial automation systems and integration — Service interface for testing applications — Part 1: Overview*

ISO 20242-3, *Industrial automation systems and integration — Service interface for testing applications — Part 3: Virtual device service interface*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 20242-1, ISO 20242-3 and the following apply.

3.1
communication object
existing object which may be accessed with a communication function to read or write a value

[ISO 20242-1:2005, definition 2.3]

3.2
coordinator
program with a specified interface to handle the access of an application program to one or more device drivers and to manage real-time application aspects, synchronization and events

[ISO 20242-1:2005, definition 2.4]

3.3
coordinator capability description
text file containing information about the capabilities of coordinators in a defined format (i.e. structure, syntax)

3.4
device capability description
text file containing information about the capabilities of virtual devices in a defined format (i.e. structure, syntax)

[ISO 20242-1:2005, definition 2.5]

3.5
device driver
software module providing an ISO 20242-specified interface with service functions to call a platform adapter to access physical devices

[ISO 20242-2:2010, definition 3.1]

3.6
function object
instance describing one capability of a virtual device

[ISO 20242-3:2011, definition 3.4]

3.7
operation
instance describing one complete procedure

[ISO 20242-3:2011, definition 3.5]

3.8
parameterization instance description
information about the configuration of a coordinator and of virtual devices

3.9
virtual device
representation of one or more physical devices and/or stand-alone software modules that provide an unambiguous view of the resources of a communication interface

[ISO 20242-3:2011, definition 3.7]

4 Abbreviated terms

CCD	Coordinator Capability Description
DCD	Device Capability Description
DCPT	Device Capability Profile Template
PID	Parameterization Instance Description

VD	Virtual Device
VDSI	Virtual Device Service Interface
XML	eXtensible Markup Language

5 Device capability profile framework

5.1 General

For the design of device capability profile templates (DCPTs), device capability descriptions (DCDs) and coordinator capability descriptions (CCDs), a device capability profile framework as shown in Figure 1 shall be used. The generic DCPT shall use the generic information exchange profile template of ISO 15745-1 as a skeleton and shall be constructed in XML schema. The technology-specific DCPT describes technology-specific capabilities and shall extend the generic DCPT in XML schema. A DCD describes capabilities of virtual devices in a device driver and shall extend a technology-specific DCPT in XML schema. A CCD describes capabilities of a coordinator and shall extend a technology-specific DCPT in XML schema. A CCD imports DCDs for device drivers and describes capabilities of a system. A parameterization instance description (PID) describes configurations of a coordinator and device drivers. A PID shall be constructed by instantiations of one CCD and one to many DCDs. The PID is a realization of the information exchange profile in ISO 15745-1 and can be used together with other profiles of ISO 15745.

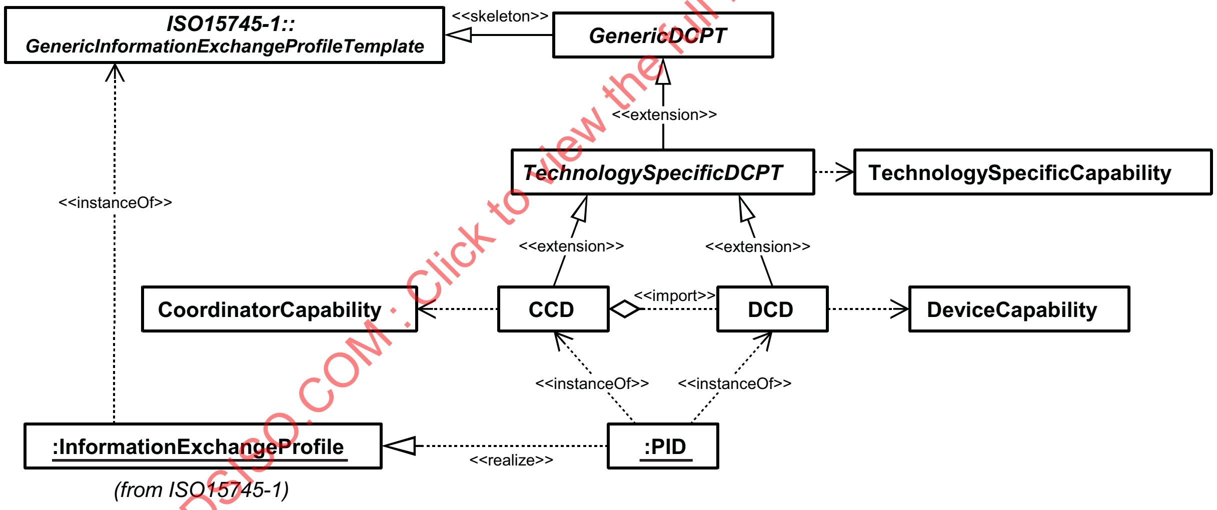


Figure 1 — Class diagram of device capability profile framework

Figure 2 shows the class diagram of CCD and DCD in the framework. The generic DCPT specifies the aggregate of the generic CCD and generic DCD. The generic CCD describes generic capabilities of a coordinator. The generic DCD describes generic capabilities of virtual devices. The technology-specific CCD in a technology-specific DCPT shall inherit the generic CCD and describes technology-specific capabilities of a coordinator. The technology-specific DCD shall inherit the generic DCD and describes technology-specific capabilities of virtual devices.

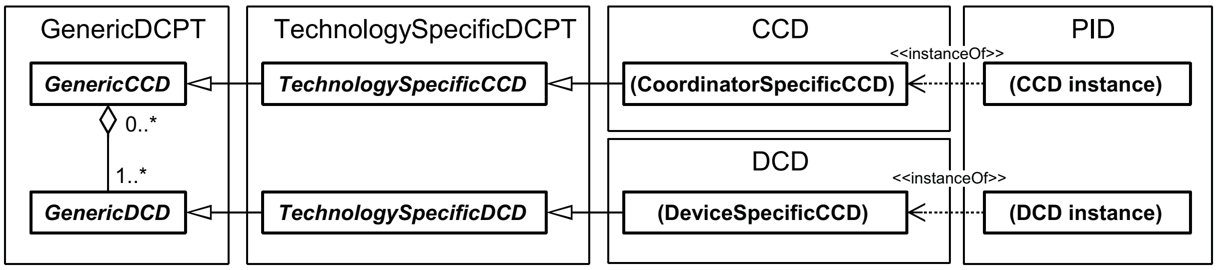


Figure 2 — Class diagram of CCD and DCD

The coordinator-specific CCD shall inherit the technology-specific CCD and describes coordinator-specific capabilities. The name of a coordinator-specific CCD can be described by the coordinator-specific name. The device-specific DCD shall inherit the technology-specific DCD and shows device-specific capabilities. The name of a device-specific DCD can be described by the device-driver-specific name. Between the generic CCD and the generic DCD, there shall be an aggregation relation as shown in Figure 2. A generic CCD may contain one to many generic DCDs. Different generic CCDs may use the same generic DCD. Generic DCDs may exist without a generic CCD. The coordinator-specific CCD and the device-specific DCD have the aggregation relation from inheritance. A CCD can import the DCDs of device drivers. The CCD instance in a PID shall be an XML instance of the coordinator-specific CCD XML schema. A DCD instance in a PID shall be an XML instance of a device-specific DCD XML schema. The XML tag name of a CCD instance shall be the name of the coordinator-specific CCD. The XML tag name of a DCD instance shall be the name of the device-specific DCD.

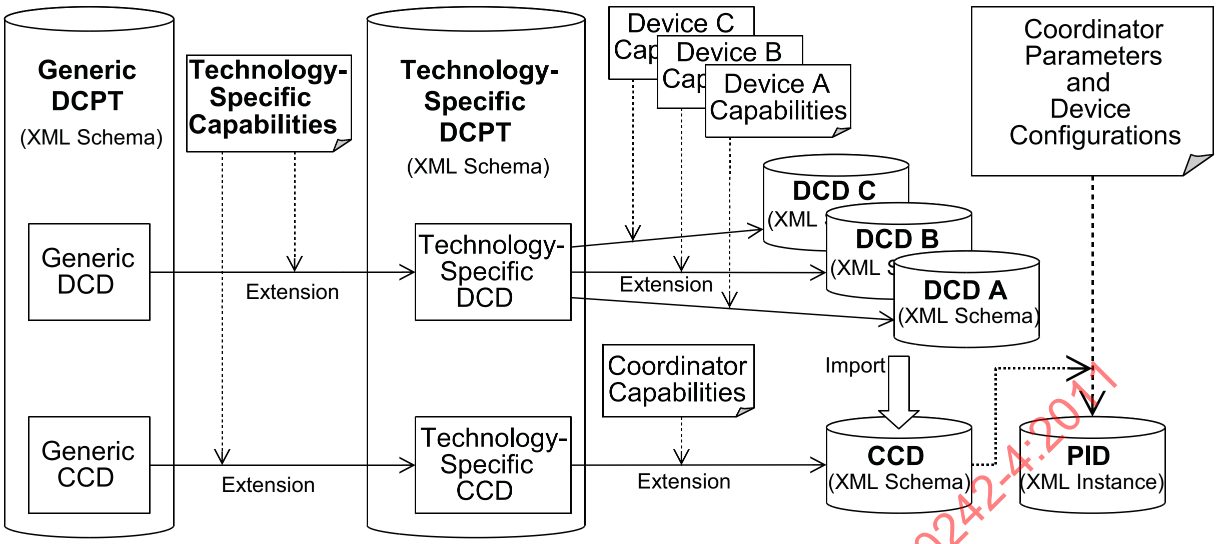
5.2 Creation procedure of DCD, CCD and PID

5.2.1 General

Figure 3 shows the creation procedure of DCD and CCD. Clause 6 defines the generic DCPT in XML schema. The technology of each service interface defines technology-specific capabilities as a technology-specific DCPT.

NOTE Technology-specific DCPTs for ASAM GDI, MICX and ORiN are described in Annexes A, B and C.

The vendor of a device, equipment or software module shall extend the technology-specific DCD of the technology-specific DCPT with the capabilities of the device or equipment. The resulting DCD XML schema shall be provided together with a device driver. The vendor of a coordinator shall extend the technology-specific CCD of the technology-specific DCPT with the capabilities of the coordinator. The resulting CCD XML schema shall be provided with the coordinator. A configuration tool can import the CCD and the required DCDs to create the PID XML instance with the configuration data for the required application. The PID shall contain all required instances with names and values. The coordinator can read the PID XML instance and perform a coordinator setup and the configuration of devices, equipment or software modules. The configuration shall be performed via the virtual device service interface (VDSI) of the device drivers as defined in ISO 20242-3.



NOTE In this figure, a cylinder shape shows the XML file and a paper shape shows the capability or configuration information. A solid arrow shows the extension of the XML schema file. A broken arrow shows the injection of capability or configuration information. A small dotted arrow shows the application of the XML schema.

Figure 3 — Creation procedure of CCD and DCD

5.2.2 Device capability description (DCD)

The DCD shall be an XML schema and shall contain the following:

— identification information of the device driver;

— a description of device capabilities of virtual devices supported by the device driver.

5.2.3 Coordinator capability description (CCD)

The CCD shall be an XML schema and shall contain the following:

— identification information of the coordinator software;

— a description of coordinator capabilities;

— qualification of the supported application program service interface.

5.2.4 Parameterization instance description (PID)

The PID shall be an XML instance and shall contain the following:

— identification information of the PID as information exchange profile of ISO 15745;

— a description of parameterization instances with application-defined names;

— configuration data for the device drivers.

6 Generic device capability profile template

6.1 General

The generic DCPT defines the common structure of DCPT and is not dependent on technology of a service interface. A technology-specific DCPT extends the generic DCPT.

6.2 Generic DCPT model

The generic DCPT extends the information of the VDSI model in ISO 20242-3. Figure 4 shows the class structure of the generic DCPT. The information exchange profile is the root class and contains the ISO 15745 header and the ISO 15745 body. The ISO 15745 header describes the identification information of a profile. The ISO 15745 body contains one or more generic CCDs. The information exchange profile, the ISO 15745 header and the ISO 15745 body are defined in ISO 15745-1.

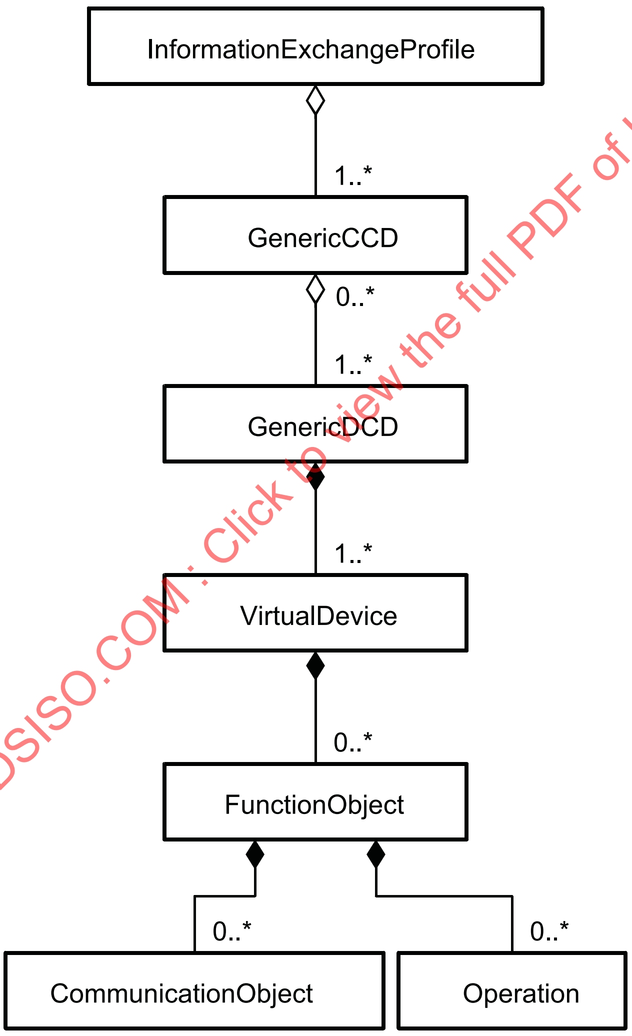


Figure 4 — Class diagram of generic DCPT model

The generic CCD is an abstract class that describes generic capabilities of a coordinator. The number of generic CCDs is equal to the number of coordinators.

A generic CCD contains generic DCDs. The generic DCD is an abstract class that describes generic capabilities of device drivers. The number of generic DCDs is equal to the number of device drivers.

A generic DCD contains virtual devices. The virtual device is an abstract class that describes generic capabilities of virtual devices.

A virtual device contains function objects. The function object is an abstract class that describes generic capabilities of functions of virtual devices.

A function object contains communication objects and operations.

The communication object is an abstract class that describes generic capabilities of communication objects defined in ISO 20242-3.

The operation is an abstract class that describes generic capabilities of operations defined in ISO 20242-3.

6.3 Generic DCPT XML schema

The generic DCPT XML schema in Figure 5 describes the information exchange profile template. The generic DCPT XML schema includes the generic CCD XML schema and refers to the generic CCD element. The DCPT header used in the annotation part of Figure 5 is defined in 7.2.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns="http://www.osi.ch/iso/ISO20242-4/GenericDCPT"
            targetNamespace="http://www.osi.ch/iso/ISO20242-4/GenericDCPT"
            elementFormDefault="qualified">

 <xsd:annotation>
  <xsd:appinfo source="DCPTHeader.xsd">
   <DCPTHeader>
    <DCPTIdentification>GenericDCPT</DCPTIdentification>
    <DCPTRevision>1.0</DCPTRevision>
    <DCPTName>Generic DCPT</DCPTName>
    <DCPTSource>GenericDCPT.xsd</DCPTSource>
    <DCPTDate>2011-07-01</DCPTDate>
   </DCPTHeader>
  </xsd:appinfo>
 </xsd:annotation>

 <!--   * Include GenericCCD *   -->
 <xsd:include schemaLocation="GenericCCD.xsd"/>

 <!--   * ISO15745 Profile Root *   -->
 <xsd:element name="ISO15745Profile">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element ref="ProfileHeader"/>
    <xsd:element ref="ProfileBody"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>

 <!-- * HEADER DATA TYPES *   -->
 <xsd:element name="ProfileHeader">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="ProfileIdentification" type="xsd:string"/>
    <xsd:element name="ProfileRevision"       type="xsd:string"/>
    <xsd:element name="ProfileName"           type="xsd:string"/>
    <xsd:element name="ProfileSource"         type="xsd:string"/>
    <xsd:element name="ProfileClassID"        type="ProfileClassID_DataType"
                                              fixed="InformationExchange"/>
    <xsd:element name="ProfileDate"           type="xsd:date"
                 minOccurs="0"/>
    <xsd:element name="AdditionalInformation" type="xsd:anyURI"
                 minOccurs="0"/>
    <xsd:element name="ISO15745Reference"     type="ISO15745Reference_DataType"/>
    <xsd:element name="IASInterfaceType"      type="IASInterface_DataType"
                                              fixed="CSI"
                 minOccurs="0"                maxOccurs="unbounded"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>

 <xsd:complexType name="ISO15745Reference_DataType">
  <xsd:sequence>
   <xsd:element name="ISO15745Part"           type="xsd:string"
                                              fixed="1"/>
   <xsd:element name="ISO15745Edition"        type="xsd:string"
                                              fixed="1"/>
   <xsd:element name="ProfileTechnology"      type="xsd:string"
                                              fixed="None"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:simpleType name="ProfileClassID_DataType">
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="AIP"/>
   <xsd:enumeration value="Process"/>
   <xsd:enumeration value="InformationExchange"/>
   <xsd:enumeration value="Resource"/>
   <xsd:enumeration value="Device"/>
   <xsd:enumeration value="CommunicationNetwork"/>
   <xsd:enumeration value="Equipment"/>
   <xsd:enumeration value="Human"/>
   <xsd:enumeration value="Material"/>
  </xsd:restriction>
 </xsd:simpleType>

 <xsd:simpleType name="IASInterface_DataType">
  <xsd:union>
   <xsd:simpleType>
    <xsd:restriction base="xsd:string">
     <xsd:enumeration value="CSI"/>
     <xsd:enumeration value="HCI"/>
     <xsd:enumeration value="ISI"/>
     <xsd:enumeration value="API"/>
     <xsd:enumeration value="CMI"/>
     <xsd:enumeration value="ESI"/>
     <xsd:enumeration value="FSI"/>
     <xsd:enumeration value="MTI"/>
     <xsd:enumeration value="SEI"/>
     <xsd:enumeration value="USI"/>
    </xsd:restriction>
   </xsd:simpleType>
   <xsd:simpleType>
    <xsd:restriction base="xsd:string">
     <xsd:length value="4"/>
    </xsd:restriction>
   </xsd:simpleType>
  </xsd:union>
 </xsd:simpleType>

 <!-- * BODY SECTION *  -->
 <xsd:element name="ProfileBody">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element ref="GenericCCD" maxOccurs="unbounded"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Figure 5 — Generic DCPT XML schema

The generic CCD XML schema in Figure 6 describes the generic CCD template. The generic CCD XML schema includes the generic DCD XML schema and refers to the generic DCD element. The DCPT header used in the annotation part of Figure 5 is defined in 7.2.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns="http://www.osi.ch/iso/ISO20242-4/GenericDCPT"
            targetNamespace="http://www.osi.ch/iso/ISO20242-4/GenericDCPT"
            elementFormDefault="qualified">

 <xsd:annotation>
  <xsd:appinfo source="DCPTHeader.xsd">
   <DCPTHeader>
    <DCPTIdentification>GenericCCD</DCPTIdentification>
    <DCPTRevision>1.0</DCPTRevision>
    <DCPTName>Generic CCD</DCPTName>
    <DCPTSource>GenericCCD.xsd</DCPTSource>
    <DCPTDate>2011-07-01</DCPTDate>
   </DCPTHeader>
  </xsd:appinfo>
 </xsd:annotation>

 <!--   * Include GenericDCD *   -->
 <xsd:include schemaLocation="GenericDCD.xsd"/>

 <!-- * Elements Declaration *  -->
 <xsd:element name="GenericCCD"   type="GenericCCDType" abstract="true"/>

 <xsd:complexType name="GenericCCDType" abstract="true">
  <xsd:sequence>
   <xsd:element ref="GenericDCD" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name"      type="xsd:string"/>
  <xsd:attribute name="category"  type="xsd:string"
                 use="required"   fixed="CCD"/>
 </xsd:complexType>
</xsd:schema>
```

Figure 6 — Generic CCD XML schema

The generic DCD XML schema in Figure 7 describes the generic DCD template. It includes templates of the virtual device, the function object, the communication object and the operation. The DCPT header used in the annotation part of Figure 5 is defined in 7.2.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns="http://www.osi.ch/iso/ISO20242-4/GenericDCPT"
            targetNamespace="http://www.osi.ch/iso/ISO20242-4/GenericDCPT"
            elementFormDefault="qualified">

 <xsd:annotation>
  <xsd:appinfo source="DCPTHeader.xsd">
   <DCPTHeader>
    <DCPTIdentification>GenericDCD</DCPTIdentification>
    <DCPTRevision>1.0</DCPTRevision>
    <DCPTName>Generic DCD</DCPTName>
    <DCPTSource>GenericDCD.xsd</DCPTSource>
    <DCPTDate>2009-03-16</DCPTDate>
   </DCPTHeader>
  </xsd:appinfo>
 </xsd:annotation>

 <!-- * Elements Declaration *  -->
 <xsd:element name="GenericDCD"
              type="GenericDCDType" abstract="true"/>
 <xsd:complexType name="GenericDCDType" abstract="true">
  <xsd:sequence>
   <xsd:element ref="VirtualDevice" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name"        type="xsd:string"/>
  <xsd:attribute name="category"    type="xsd:string"
                 use="required"     fixed="DCD"/>
 </xsd:complexType>

 <xsd:element name="VirtualDevice"
              type="VirtualDeviceType" abstract="true"/>

 <xsd:complexType name="VirtualDeviceType" abstract="true">
  <xsd:sequence>
   <xsd:element ref="FunctionObject" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name"        type="xsd:string"/>
  <xsd:attribute name="category"    type="xsd:string"
                 use="required"     fixed="MODULE"/>
 </xsd:complexType>

 <xsd:element name="FunctionObject"
              type="FunctionObjectType" abstract="true"/>

 <xsd:complexType name="FunctionObjectType" abstract="true">
  <xsd:sequence>
   <xsd:element ref="CommunicationObject" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="Operation"           minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name"        type="xsd:string"/>
  <xsd:attribute name="category"    type="xsd:string"
                 use="required"     fixed="INTERFACE"/>
 </xsd:complexType>

 <xsd:element name="CommunicationObject"
              type="CommunicationObjectType" abstract="true"/>

 <xsd:complexType name="CommunicationObjectType" abstract="true">
  <xsd:attribute  name="name" type="xsd:string"/>
 </xsd:complexType>

 <xsd:element name="Operation"
              type="OperationType" abstract="true"/>

 <xsd:complexType name="OperationType" abstract="true">
  <xsd:attribute name="name"        type="xsd:string"/>
  <xsd:attribute name="category"    type="xsd:string"
                 use="required"     fixed="OPERATION"/>
 </xsd:complexType>

</xsd:schema>
```

Figure 7 — Generic DCD XML schema

7 Common rules for DCPT

7.1 General

The common rules shall be used for the extension of DCPT. Names of XML elements (tags) may be defined corresponding to a coordinator and to devices. Any attributes and elements may be added for a special purpose that is not addressed by this part of ISO 20242.

7.2 DCPT header

The identification information of a DCD and CCD is described with xsd:appinfo element of xsd:annotation in XML Schema. The DCPT header contains the attributes specified in Table 1. The XML data types used in Table 1 are defined in REC-xmlschema-2-20041028[9].

Table 1 — DCPT header elements

Element	Description
DCPTIdentification	DCPT identification. XML data type : string EXAMPLE ABC-123-XX
DCPTRevision	Revision of the DCPT. XML data type : string EXAMPLE 2.34
DCPTName	Descriptive name of the DCPT. XML data type : string EXAMPLE DCD Thermometer
DCPTSource	Identification of the DCPT developer. XML data type : string EXAMPLE ASAM
DCPTClassID	Identification of the profile class. XML data type : string Valid profile classes are: GenericDCPT TechnlogySpecificDCPT CCD DCD EXAMPLE DCD
DCPTDate	The release date of this revision of the profile in CCYY-MM-DD format. XML data type : date EXAMPLE 2011-09-21
AdditionalInformation	Location of diagrams/additional information for the profile. This field is optional. XML data type : anyURI EXAMPLE http://www.asam.net
ISO20242Reference	Identifies the part of ISO 20242 (see ISO20242Part), together with its edition (see ISO20242Edition) and the technology (see Technology). Multiple references are allowed, e.g. for a device with more than one communication interface.
ISO20242Edition	Edition of the referenced part of ISO 20242. XML data type : positiveInteger EXAMPLE 1
Technology	Name of the referenced technology. XML data type : string If no ISO 20242 technology is applicable, then the value “None” shall be used. EXAMPLE None

The XML Schema is described in Figure 8.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

 <xsd:element name="DCPTHeader">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="DCPTIdentification" type="xsd:string"/>
    <xsd:element name="DCPTRevision" type="xsd:string"/>
    <xsd:element name="DCPTName" type="xsd:string"/>
    <xsd:element name="DCPTSource" type="xsd:string"/>
    <xsd:element name="DCPTClassID" type="DCPTClassID_DataType"/>
    <xsd:element name="DCPTDate" type="xsd:string"/>
    <xsd:element name="AdditionalInformation" type="xsd:anyURI" minOccurs="0"/>
    <xsd:element name="ISO20242Reference" type="ISO20242Reference_DataType"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>

 <xsd:simpleType name="DCPTClassID_DataType">
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="GenericDCPT"/>
   <xsd:enumeration value="TechnlogySpecificDCPT"/>
   <xsd:enumeration value="CCD"/>
   <xsd:enumeration value="DCD"/>
  </xsd:restriction>
 </xsd:simpleType>

 <xsd:complexType name="ISO20242Reference_DataType">
  <xsd:sequence>
   <xsd:element name="ISO20242Part" type="xsd:string" fixed="4"/>
   <xsd:element name="ISO20242Edition" type="xsd:string" fixed="1"/>
   <xsd:element name="Technology" type="xsd:string"/>
  </xsd:sequence>
 </xsd:complexType>
</xsd:schema>
```

Figure 8 — DCPT header XML schema

7.3 Extension of profile template

7.3.1 Creating DCD

A DCD contains device type specific elements which are derived from technology-specific elements and substitute the abstract elements of the generic DCPT.

NOTE This part of ISO 20242 does not require the import of the generic or technology-specific XML schemas into device-specific DCDs. However, independently created DCDs will need to contain the same information as DCDs with imported XML schemas of this part of ISO 20242.

The element names (tags) for the DCD may be specific to device types. The elements shall be supplemented by an attribute named "category" containing a keyword for the type of this element as defined in Table 2. The content of this attribute is a reference to the corresponding element of generic DCPT.

Table 2 — Keywords for type of element

Element of generic DCPT	Contents of XML attribute "category"
GenericCCD	CCD
GenericDCD	DCD
VirtualDevice	MODULE
FunctionObject	INTERFACE
Operation	OPERATION

Additional elements may get specific contents for attribute "category" in technology-specific annexes.

7.3.2 Assignment of device-specific elements

A device-specific DCD element contains one or more device-specific virtual device elements which shall not belong to any other device-specific DCD element.

A device-specific VD element is the substitute for the generic VD element and contains zero or more device-specific function object elements which shall not belong to any other device-specific VD element.

A device-specific function object element is the substitute for the generic function object element and contains zero or more device-specific communication object elements which shall not belong to any other device-specific function object element. Also, the device-specific function object element contains zero or more device-specific operation elements which shall not belong to any other device-specific function object element.

7.3.3 Instantiation order for VDSI

For the creation of virtual devices, function objects and communication objects via services VDSI_Initiate, VDSI_CreateFuncObject and VDSI_CreateCommObject of ISO 20242-3, an extra XML attribute, named "initOrder", shall supplement virtual device elements, function object elements and communication object elements. This XML attribute is of type unsignedInt as defined in XML schema and contains the order for creating the instance of the element via VDSI.

7.3.4 Parameterization of communication objects

If a value is assigned to a communication object, this value shall be written by the coordinator with the VDSI_Write service of VDSI. It is possible to define an extra order for writing values to communication objects. This is done using the XML attribute named "initOrder".

NOTE Annex A provides an example that shows how a communication object can be written multiple times.

7.4 Assignment of textual information

Any element of DCPT may have two kinds of textual information attached. A short message and a long message. These messages shall be defined in an extra XML instance and numbered as described in Clause 8. Message numbering is hierarchically organized in numbered text areas with numbered text entries. The text numbers are assigned to DCPT elements by optional XML attributes of type xsd:unsignedInt described in Table 3.

Table 3 — XML attributes for text assignment

XML attribute name	Description of content
areaMsg	number of area of short message
infMsg	number of short message
areaText	number of area of long message
infText	number of long message

7.5 Creating PID

7.5.1 XML instances of CCD classes

An XML instance of a coordinator-specific CCD class describes features of a coordinator. The CCD instance contains one or more XML instances of DCD classes.

7.5.2 XML instances of DCD classes

An XML instance of a device-specific DCD class describes a device driver that is an entity of a VDSI of ISO 20242-3. The DCD instance contains one or more XML instances of device-specific VD classes.

7.5.3 XML instances of VD classes

An XML instance of a device-specific VD class describes an instance of a VD created by service VDSI_Initiate of VDSI. There may be multiple instances of one device-specific VD class. The instance of a VD contains zero or more instances of device-specific function object classes.

7.5.4 XML instances of function object classes

An XML instance of a device-specific function object class describes an instance of a function object created by service VDSI_CreateFuncObject of VDSI. There may be multiple instances of one device-specific function object class. The instance of a function object contains zero or more instances of device-specific communication object classes and/or zero or more instances of device-specific operation classes.

7.5.5 XML instances of communication object classes

An XML instance of a device-specific communication object class describes an instance of a communication object created by service VDSI_CreateCommObject of VDSI. There shall be only one instance of one concrete communication object class.

7.5.6 XML instances of operation classes

An XML instance of a device-specific operation class describes an instance of an operation created implicitly with the instantiation of the associated function object class via VDSI. There shall be only one instance of one device-specific operation class.

8 Multilingual text elements

For textual information, this part of ISO 20242 defines three kinds of messages:

— short information messages;

— long information messages;

— error messages.

These messages shall be contained in an XML instance with a structure defined in Figure 9.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            elementFormDefault="qualified">

 <xsd:element name="DIT">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="Area" minOccurs="0" maxOccurs="unbounded">
     <xsd:complexType>
      <xsd:all>
       <xsd:element name="SubAreaText"    type="SubAreaType"/>
       <xsd:element name="SubAreaMessage" type="SubAreaType"/>
       <xsd:element name="SubAreaError"   type="SubAreaType"/>
      </xsd:all>
      <xsd:attribute name="number" type="xsd:unsignedInt" use="required"/>
      <xsd:attribute name="name"   type="xsd:Name"        use="optional"/>
     </xsd:complexType>
    </xsd:element>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>

 <xsd:complexType name="SubAreaType">
  <xsd:sequence>
   <xsd:element name="Entry"  type="EntryType"
                minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="EntryType">
  <xsd:sequence>
   <xsd:element name="Text"   type="LangSpecType"
                minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="number" type="xsd:unsignedInt" use="required"/>
  <xsd:attribute name="name"   type="xsd:Name"        use="optional"/>
 </xsd:complexType>

 <xsd:complexType name="LangSpecType">
  <xsd:simpleContent>
   <xsd:extension base="xsd:normalizedString">
    <xsd:attribute name="lang" type="xsd:language" use="required"/>
   </xsd:extension>
  </xsd:simpleContent>
 </xsd:complexType>
```

Figure 9 — Device information text XML schema

Long information messages are placed in XML element <SubAreaText>, short information messages are placed in element <SubAreaMessage> and error messages are placed in element <SubAreaError>. Language-specific messages are placed in XML elements <Text> with the XML attribute "lang" for identifying the language.

Annex A
(informative)

ASAM GDI device capability profile template

A.1 General

The ASAM GDI (Generic Device Interface) specifies a general device interface for testing applications. This annex defines the GDI-specific device capability profile template of ASAM GDI Version 4.4[3], and provides examples of DCD, CCD and PID.

A.2 GDI-specific profile model

A.2.1 General

The GDI-specific profile model contains all information that is necessary for the device capability description and all information that is necessary for parameterization. Figure A.1 shows the class diagram of GDI-specific device capability profile template.

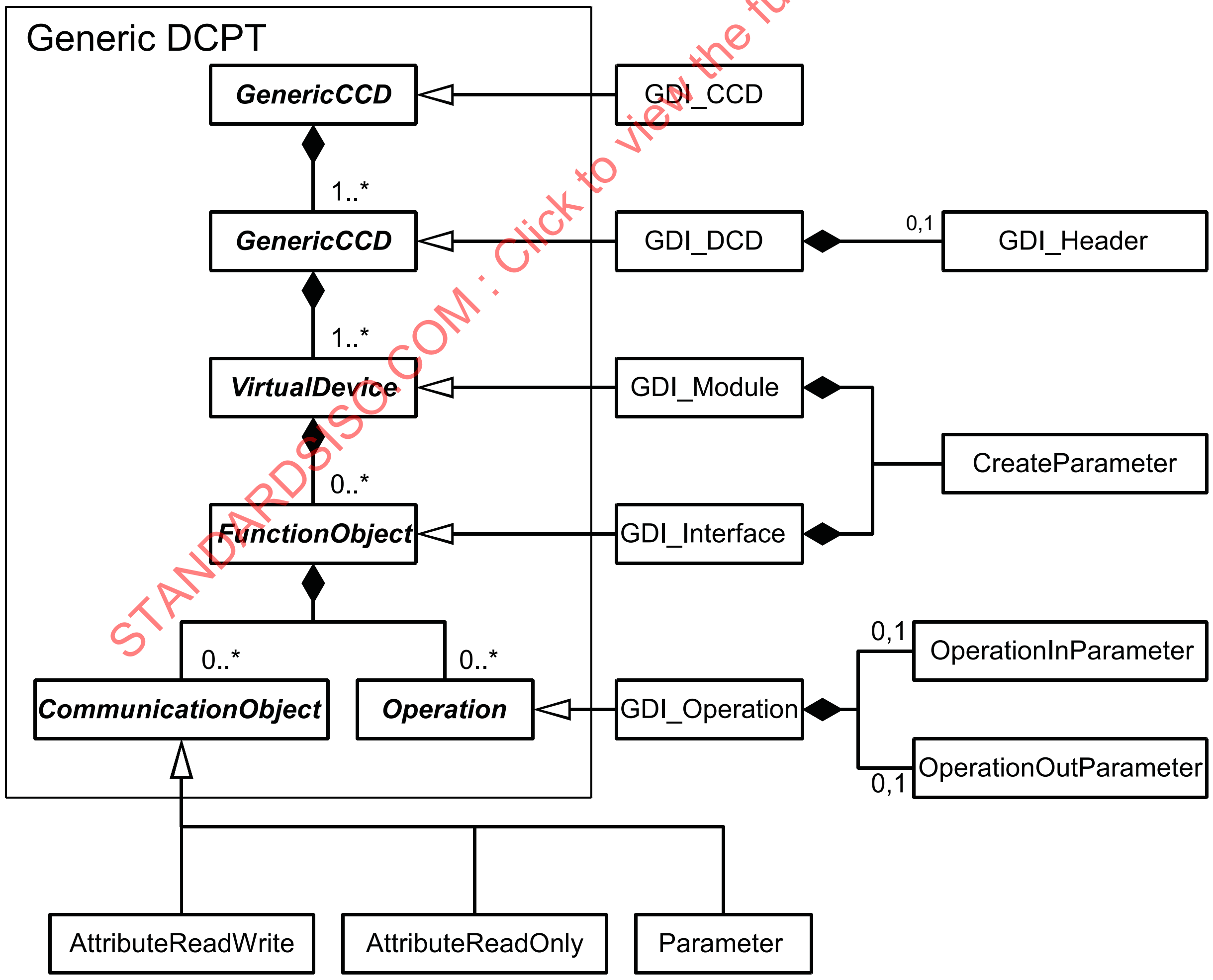


Figure A.1 — Class diagram of GDI-specific DCPT model

A.2.2 GDI_CCD

The GDI_CCD class shows the capabilities of the GDI-specific coordinator. It inherits the GenericCCD class and is an abstract class. A coordinator-specific CCD class inherits it and defines capabilities of the specific coordinator.

A.2.3 GDI_DCD

The GDI_DCD class shows the capabilities of the GDI-specific device driver. It inherits the GenericDCD class and is an abstract class. A device-specific GDI_DCD class inherits it and defines capabilities of the specific device driver.

A.2.4 GDI_Header

The GDI_Header class contains additional information, which is used for the instantiation of the device driver. The elements of GDI_Header are described in Table A.1. A definition of GDI_Header is also given in GDIcommon.xsd in A.6.2.

Table A.1 — Elements of GDI_Header

Element of GDI_Header		Element type	Description
DCD_Version		xsd:unsignedInt	A number for the DCD version
DeviceVersion		xsd:unsignedInt	A number for the device version
DriverName		xsd:string	The name of the driver
DriverVersion		xsd:unsignedInt	A number for the driver version
Factory		xsd:string	The name of the manufacturer
DIT		xsd:string	The name of the XML text file
GDI_Version	Major	xsd:unsignedByte	Major version number
	Minor	xsd:unsignedByte	Minor version number
	Revision	xsd:unsignedByte	Revision number

A.2.5 GDI_Module

The GDI_Module class shows capabilities of the GDI-specific virtual device. It inherits the VirtualDevice class and is an abstract class. A device-specific GDI_Module class inherits it and defines the capabilities of the specific virtual device. A GDI_Module class may have a CreateParameter and is identified by a number contained in an additional XML attribute named "moduleId" of type "xsd:unsignedShort".

A.2.6 GDI_Interface

The GDI_Interface class shows capabilities of function objects of the GDI-specific virtual device. It inherits the FunctionObject class and is an abstract class. A device-specific GDI_Interface class inherits it and defines the capabilities of the device-specific function object. A GDI_Interface class may have a CreateParameter and is identified by a number contained in an additional XML attribute named "funcId" of type "xsd:unsignedShort".

A.2.7 CreateParameter

The CreateParameter class describes a create parameter of a function object or a virtual device. It is an abstract class. Each device-specific CreateParameter class inherits it and defines the data type of the create parameter. VDSI services VDSI_Initiate and VDSI_CreateFuncObject use create parameters.

A.2.8 GDI_Operation

The GDI_Operation class describes an operation of the GDI-specific virtual device. It inherits the Operation class and is an abstract class. Each device-specific GDI_Operation class inherits it and defines the capabilities of the operation. Each GDI_Operation has one operation input parameter and one operation output parameter that may occur zero or one time in the XML instance. The assignment of a value to the operation input parameter in the XML instance (PID) marks that the operation shall be executed for configuration. A GDI_Operation is identified by a number contained in an additional XML attribute named "operationIdId" of type "xsd:unsignedShort".

A.2.9 OperationInParameter

The OperationInParameter marks the execution of an operation. It is an abstract class. Each device-specific OperationInParameter class inherits this class and defines the data type of the input parameter.

A.2.10 OperationOutParameter

The OperationOutParameter describes the return value of an operation. It is an abstract class and contains the value of the output parameter. Each device-specific OperationOutParameter class inherits this class and defines the data type of the value.

A.2.11 AttributeReadOnly

The AttributeReadOnly class describes a runtime value of the function object that can only be read. It inherits the CommunicationObject class and is an abstract class. Each device-specific AttributeReadOnly class inherits this class and defines the capabilities of each runtime value and the data type of the value.

A.2.12 AttributeReadWrite

The AttributeReadWrite class describes a runtime value of the function object that can be read and written. It inherits the CommunicationObject class, and is an abstract class. Each device-specific AttributeReadWrite class inherits this class and defines the capabilities of each runtime value and its data type.

A.2.13 Parameter

The Parameter class shows the capabilities of a parameter of the function object. A parameter can be read and written. The Parameter class inherits the CommunicationObject class and is an abstract class. Each device-specific Parameter class inherits this class and defines the capabilities of each parameter and its data type.

A.2.14 Identification of GDI communication objects

The instances of classes AttributeReadOnly, AttributeReadWrite and Parameter are identified by a number, which is the counting of these instances inside a FunctionObject instance, starting at 1.

A.3 Elements for typed Data and Configuration Scenarios

A.3.1 General

In 7.3.3, the XML attribute initOrder was introduced for describing sequences of device configuration, which concerns the creation of virtual devices, function objects and communication objects. There is another configuration scenario defined for the data for communication objects, which may get different values at different times. For this purpose, the class OrderedValue is introduced. OrderedValue is not necessary, if no data write scenario is defined, but may be used also for this case by using the same value in initOrder of CommunicationObject and OrderedValue.

Another element Value is defined to facilitate the inheritation of data types without influencing the inheritation of structural classes. With this method, the definition of data types for communication objects and create parameters is decoupled from the definition of the communication object itself. Value is not necessary, if data types are defined implicitly with the communication object.

A.3.2 OrderedValue

The OrderedValue class describes a value that can be set multiple at different times. Setting a value means to write the value by using services VDSI_Write or VDSI_Execute. This is applicable for objects of classes AttributeReadWrite, Parameter and OperationInParameter. The ordered value is an abstract class. It contains an initialising order in its XML attribute initOrder that describes when the value shall be set, and the value itself. Each device-specific OrderedValue class inherits the OrderedValue class and defines the data type of the value (see Figure A.2).

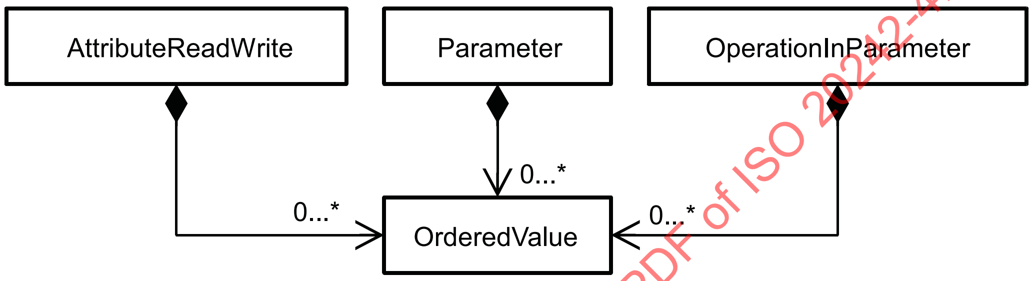


Figure A.2 — Element OrderedValue for configuration scenarios

A.3.3 Value

The Value class describes an element of type xsd:anyType, which may be restricted to any other type and therefore it is a placeholder for any element type defined in the device capability description elsewhere. The use of this extra element facilitates a decoupled inheritation for the definition of data types (see Figure A.3).

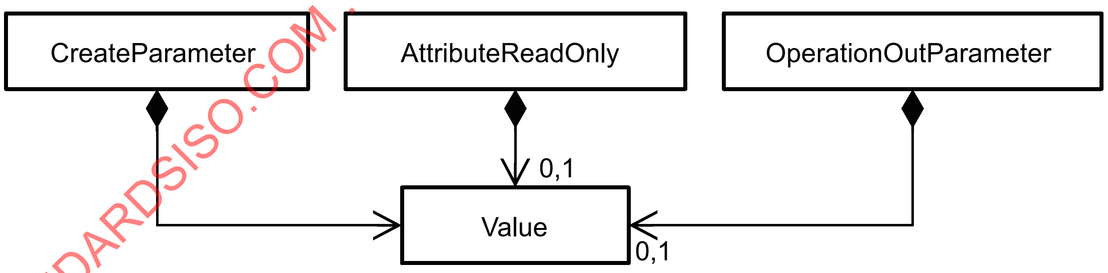


Figure A.3 — Element Value for decoupled type definitions

Value contains the data, which is exchanged with service primitive parameter Create Parameter, when service VDSI_Initiate is used for instantiation of a virtual device or when service VDSI_CreateFuncObject is used for instantiation of a function object.

For configuration, no data exchange happens with communication object AttributeReadOnly and with OperationOutParameter of GDI_Operation. But Value may be used to present the data type of the objects.

Value is also used together with OrderedValue for classes AttributeReadWrite, Parameter and OperationInParameter, if a configuration scenario with different values at different times is needed. If no configuration scenario is needed, Value may also be used instead of OrderedValue with AttributeReadWrite, Parameter and OperationInParameter (see Figure A.4).

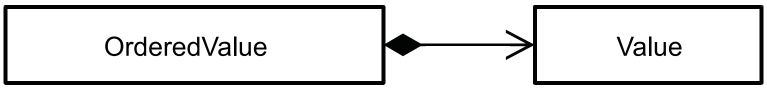


Figure A.4 — Elements Value and OrderedValue for configuration scenarios

A.4 Additional type identification to generic DCPT

Tables A.2 to A.4 contain technology-specific type identifications for create parameter, communication objects and operation parameters.

Table A.2 — Type identification for communication objects

Element of technology-specific DCPT	Contents of XML attribute "category"	Contents of XML attribute "readonly"[a]
AttributeReadWrite	ATTRIBUTE	false
AttributeReadOnly	ATTRIBUTE	true
Parameter	PARAMETER	(false)

[a] The extra XML attribute "readonly" of type xsd:boolean is mandatory for AttributeReadWrite and AttributeReadOnly. If it is also used with Parameter, it shall have the content "false".

Table A.3 — Type identification for operation parameters

Element of technology-specific DCPT	Contents of XML attribute "category"
OperationInParameter	IN
OperationOutParameter	OUT

Table A.4 — Type identification for create parameters

Element of technology-specific DCPT	Contents of XML attribute "category"
CreateParameter	CREATEPARAMETER

A.5 Additional XML attributes to generic DCPT

The communication objects AttributeReadOnly and AttributeReadWrite may be used for unsolicited data transmission with services VDSI_InfReport and VDSI_Accept. A request for using these services at runtime is done at configuration time by setting the XML attributes infReport and accept of type xsd:boolean to the value true (see Table A.5).

Table A.5 — XML attributes for unsolicited data transfer

XML attribute name	Used with object	Description of content
infReport	AttributeReadWrite, AttributeReadOnly	Request for using service VDSI_InfReport
accept	AttributeReadWrite	Request for using service VDSI_Accept

A.6 GDI-specific device capability profile template

A.6.1 General

GDIcommon.xsd is an XML schema that contains base classes. These base classes can be used to describe an XML schema that contains all information for the device capability description and all information that is necessary for parameterization. The defined types Parameter and AttributeReadWrite may be extended with elements OrderedValue and Value. In this case Value may have a data type, which may be defined in an extra XML schema. Also, the defined types CreateParameter, AttributeReadOnly and OperationOutParameter may be extended by Value elements of external defined data types.

NOTE If elements OrderedValue and Value are not used and the communication objects contain values of simple or complex data type, there might be another construction of the XML schema, based on the extension of value data types. In this case, GDIcommon.xsd can be used as a sample for the structure of a resulting XML instance and the necessary XML attributes.

A.6.2 XML schema : GDIcommon.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             elementFormDefault="qualified" attributeFormDefault="unqualified">

  <!-- Header information is contained in each schema file -->

  <xsd:annotation>
   <xsd:appinfo source="http://www.asam.net/ISO20242-4/DCPTHeader.xsd">
    <DCPTHeader>
      <DCPTIdentification>GDI_Common</DCPTIdentification>
      <DCPTRevision>1.0</DCPTRevision>
      <DCPTName>GDIcommon</DCPTName>
      <DCPTSource>GDIcommon.xsd</DCPTSource>
      <DCPTClassID>TechnlogySpecificDCPT</DCPTClassID>
      <DCPTDate>2009-03-16</DCPTDate>
      <ISO20242Reference>
        <ISO20242Edition>1</ISO20242Edition>
        <Technology>ASAM-GDI</Technology>
      </ISO20242Reference>
    </DCPTHeader>
   </xsd:appinfo>
  </xsd:annotation>

  <!-- Valid content for XML-Attribut category (kind of DCD element) -->

 <xsd:simpleType name="Category">
   <xsd:restriction base="xsd:string">
    <xsd:enumeration value="MODULE"/>
    <xsd:enumeration value="INTERFACE"/>
    <xsd:enumeration value="CREATEPARAMETER"/>
    <xsd:enumeration value="PARAMETER"/>
    <xsd:enumeration value="ATTRIBUTE"/>
    <xsd:enumeration value="OPERATION"/>
    <xsd:enumeration value="IN"/>
    <xsd:enumeration value="OUT"/>
    <xsd:enumeration value="DCD"/>
    <xsd:enumeration value="CCD"/>
   </xsd:restriction>
  </xsd:simpleType>

  <!-- Group of attributes for assigning multilingual text to elements -->

  <xsd:attributeGroup name="TextAttributes">
   <xsd:attribute name="areaMsg"    type="xsd:unsignedShort"  use="optional"/>
   <xsd:attribute name="infMsg"     type="xsd:unsignedShort"  use="optional"/>
   <xsd:attribute name="areaText"   type="xsd:unsignedShort"  use="optional"/>
   <xsd:attribute name="infText"    type="xsd:unsignedShort"  use="optional"/>
  </xsd:attributeGroup>

  <!-- Basic type definition for most elements used in the DCD -->

  <xsd:complexType name="TNamedDCDElement" abstract="true">
   <xsd:attribute name="name"      type="xsd:string"      use="optional"/>
   <xsd:attribute name="initOrder" type="xsd:unsignedInt" use="optional"/>
   <xsd:attributeGroup ref="TextAttributes"/>
  </xsd:complexType>

  <xsd:complexType name="TOrderedValue" abstract="true">
   <xsd:sequence>
    <xsd:element name="Value" type="xsd:anyType"/>
   </xsd:sequence>
   <xsd:attribute name="initOrder" type="xsd:unsignedInt" use="required"/>
  </xsd:complexType>


  <!-- Virtual Devices have additional XML-attributes "moduleId"
       and category="MODULE". -->

  <xsd:complexType name="GDI_Module" abstract="true">
   <xsd:complexContent>
    <xsd:extension base="TNamedDCDElement">
     <xsd:attribute name="moduleId"  type="xsd:unsignedShort"
                    use="required"/>
     <xsd:attribute name="category"  type="Category"
                    use="required"   fixed="MODULE"/>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>

  <!-- Function Objects have additional XML-attributes "funcId" and
       category="INTERFACE". -->

  <xsd:complexType name="GDI_Interface" abstract="true">
   <xsd:complexContent>
    <xsd:extension base="TNamedDCDElement">
     <xsd:attribute name="funcId"     type="xsd:unsignedShort"
                    use="required"/>
     <xsd:attribute name="category"   type="Category"
                    use="required"    fixed="INTERFACE"/>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>

  <!-- Create Parameter have an additional XML-attribute
       category="CREATEPARAMETER" -->

  <xsd:complexType name="CreateParameter" abstract="true">
    <xsd:attribute name="category"   type="Category"
                   use="required"    fixed="CREATEPARAMETER"/>
    <xsd:attributeGroup ref="TextAttributes"/>
  </xsd:complexType>


  <!-- Communication Objects which may be changed have readonly="false" -->

  <xsd:complexType name="CommunicationObjectReadWrite" abstract="true">
   <xsd:complexContent>
    <xsd:extension base="TNamedDCDElement">
    <xsd:attribute name="readonly"      type="xsd:boolean"
                                        fixed="false"/>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>
  <!-- Communication Objects which may not be changed have readonly="true" -->

  <xsd:complexType name="CommunicationObjectReadOnly" abstract="true">
   <xsd:complexContent>
    <xsd:extension base="TNamedDCDElement">
    <xsd:attribute name="readonly"      type="xsd:boolean"
                   use="required"       fixed="true"/>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>

  <!-- GDI-Parameter are Communication Objects that may be changed
       and have category="PARAMETER"  -->

  <xsd:complexType name="Parameter" abstract="true">
    <xsd:complexContent>
      <xsd:extension base="CommunicationObjectReadWrite">
        <xsd:attribute name="category" type="Category"
                       use="required"  fixed="PARAMETER"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- GDI-Attribute have to be separated in readonly and read/write
       Communication Objects  -->

  <xsd:complexType name="AttributeReadWrite" abstract="true">
    <xsd:complexContent>
      <xsd:extension base="CommunicationObjectReadWrite">
        <xsd:attribute name="category"  type="Category"
                       use="required"   fixed="ATTRIBUTE"/>
        <xsd:attribute name="infReport" type="xsd:boolean"/>
            <xsd:attribute name="accept"    type="xsd:boolean"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="AttributeReadOnly" abstract="true">
    <xsd:complexContent>
      <xsd:extension base="CommunicationObjectReadOnly">
        <xsd:attribute name="category"  type="Category"
                       use="required"   fixed="ATTRIBUTE"/>
        <xsd:attribute name="infReport" type="xsd:boolean"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>


  <!-- Operations -->

  <xsd:complexType name="GDI_Operation" abstract="true">
   <xsd:complexContent>
    <xsd:extension base="TNamedDCDElement">
     <xsd:attribute name="operationId"  type="xsd:unsignedShort"
                    use="required"/>
     <xsd:attribute name="category"     type="Category"
                    use="required"      fixed="OPERATION"/>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="OperationOutParameter" abstract="true">
    <xsd:attribute name="category" type="Category"
                   use="required"  fixed="OUT"/>
    <xsd:attributeGroup ref="TextAttributes"/>
  </xsd:complexType>
```

```
  <xsd:complexType name="OperationInParameter" abstract="true">
    <xsd:attribute name="category" type="Category"
                   use="required"  fixed="IN"/>
    <xsd:attributeGroup ref="TextAttributes"/>
  </xsd:complexType>

  <!-- Device Driver (when instantiated) -->

  <xsd:complexType name="TVersion">
    <xsd:sequence>
      <xsd:element name="Major"    type="xsd:unsignedByte"/>
      <xsd:element name="Minor"    type="xsd:unsignedByte"/>
      <xsd:element name="Revision" type="xsd:unsignedByte"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="THeader">
    <xsd:sequence>
      <xsd:element name="DCD_Version"    type="xsd:unsignedInt"/>
      <xsd:element name="DeviceVersion"  type="xsd:unsignedInt"/>
      <xsd:element name="DriverName"     type="xsd:string"/>
      <xsd:element name="DriverVersion"  type="xsd:unsignedInt"/>
      <xsd:element name="Factory"        type="xsd:string"/>
      <xsd:element name="DIT"            type="xsd:string"/>
      <xsd:element name="GDI_Version"    type="TVersion"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="GDI_DCD" abstract="true">
    <xsd:sequence>
      <xsd:element name="GDI_Header" type="THeader" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="category" type="Category"
                   use="required"  fixed="DCD"/>
    <xsd:attributeGroup ref="TextAttributes"/>
       <xsd:attribute name="dllPath"       type="xsd:string" use="required"/>
       <xsd:attribute name="driverVersion" type="xsd:int"    use="required"/>
  </xsd:complexType>

 </xsd:schema>
```

A.7 Examples of DCD

A.7.1 General

DCDa1.xsd and DCDa2.xsd are XML schema examples of DCD which extend the XML schema in GDICommon.xsd in accordance with the device capabilities of the device drivers. The names of the XML elements may be defined in accordance with the claims of the user. The structural information of an XML element is defined by the content of XML attribute "category".

A.7.2 XML schema : DCDa1.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xsd:annotation>
   <xsd:appinfo source="http://www.asam.net/ISO20242-4/DCPTHeader.xsd">
   <DCPTHeader>
    <DCPTIdentification>DCD1</DCPTIdentification>
    <DCPTRevision>1.0</DCPTRevision>
    <DCPTName>DCD1</DCPTName>
    <DCPTSource>DCD1.xsd</DCPTSource>
    <DCPTClassID>DCD</DCPTClassID>
    <DCPTDate>2009-03-16</DCPTDate>
    <ISO20242Reference>
     <ISO20242Edition>1</ISO20242Edition>
     <Technology>ASAM-GDI</Technology>
    </ISO20242Reference>
   </DCPTHeader>
   </xsd:appinfo>
  </xsd:annotation>

  <xsd:include schemaLocation="GDICommon.xsd"/>

  <!-- Main type is GDI_DCD with contained Virtual Device Types.
       There may be many different Virtual Device Types contained. -->

  <xsd:complexType name="Driver01">
   <xsd:complexContent>
    <xsd:extension base="GDI_DCD">
     <xsd:sequence>
      <xsd:element name="myDevice01" type="Device01"
                   minOccurs="0"     maxOccurs="unbounded"/>
     </xsd:sequence>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>

  <!-- Virtual Device Types have fixed numbers for identification in moduleID.
       This is a restriction of type GDI_Module. If you want to avoid
       restriction, move attribute moduleId from type GDI_Module with a fixed
       number to type Device01 below, which then is an extension of GDI_Module
       and type Device01_ will be obsolete. -->

  <xsd:complexType name="Device01_">
   <xsd:complexContent>
    <xsd:restriction base="GDI_Module">
     <xsd:attribute name="moduleId"   type="xsd:unsignedShort"
                    use="required"    fixed="1000"/>
    </xsd:restriction>
   </xsd:complexContent>
  </xsd:complexType>

  <!-- Virtual Device Types typically contain a Create Paramter of a specific
       type and several Function Objects of different types.
       This exemplary Function Object type describes an input channel and it is
       possible to use up to 16 channels with each Virtual Device instance. -->

 <xsd:complexType name="Device01">
   <xsd:complexContent>
    <xsd:extension base="Device01_">
     <xsd:sequence>
      <xsd:element name="NumOfChannel"  type="CRParamDevice"/>
      <xsd:element name="fnADInput"     type="IFADInput"
                   minOccurs="0"        maxOccurs="16"/>
     </xsd:sequence>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>

  <!-- A simple Create Parameter for the Virtual Device of type Device01-->

  <xsd:complexType name="CRParamDevice">
   <xsd:complexContent>
    <xsd:extension base="CreateParameter">
     <xsd:sequence>
      <xsd:element name="Value" type="xsd:unsignedShort" default="5"/>
     </xsd:sequence>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>
  <!-- Function Object Types have fixed numbers for identification in funcID.
       This is a restriction of type GDI_Interface. If you want to avoid
       restriction, move attribute funcId from type GDI_Interface with a fixed
       number to type IFADInput below, which then is an extension of
       GDI_Interface and type IFADInput_ will be obsolete. -->

  <xsd:complexType name="IFADInput_">
   <xsd:complexContent>
    <xsd:restriction base="GDI_Interface">
     <xsd:attribute name="funcId"    type="xsd:unsignedShort"
                    use="required"   fixed="1077"/>
    </xsd:restriction>
   </xsd:complexContent>
  </xsd:complexType>


  <!-- A simple Function Object type for the Virtual Device of type Device01.
       Function Object types typically contain a Create Parameter and several
       Communication Objects of kind GDI-Attribute or GDI-Parameter.
       An example with additional GDI-Operation is given in DCD2.xsd -->

  <xsd:complexType name="IFADInput">
   <xsd:complexContent>
    <xsd:extension base="IFADInput_">
     <xsd:sequence>
      <xsd:element name="Interrupt" type="CRParamIFADInput"/>
      <xsd:element name="Polarity"  type="PolarityType" minOccurs="0"/>
      <xsd:element name="Channel"   type="ChannelType"  minOccurs="0"/>
      <xsd:element name="ADValue"   type="ADValueType"  minOccurs="0"/>
     </xsd:sequence>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>


  <!-- A simple Create Parameter for the Function Object of type IFADInput -->

 <xsd:complexType name="CRParamIFADInput">
   <xsd:complexContent>
    <xsd:extension base="CreateParameter">
     <xsd:sequence>
      <xsd:element name="Value" type="xsd:unsignedShort" default="5"/>
     </xsd:sequence>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>

  <!-- A simple readonly GDI-Attribute for the Function Object
       of type IFADInput -->

  <xsd:complexType name="ADValueType">
   <xsd:complexContent>
    <xsd:extension base="AttributeReadOnly">
     <xsd:sequence>
      <xsd:element name="OrderedValue" type="OrderedValueAD" minOccurs="0"/>
     </xsd:sequence>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>

  <!-- A simple Value-Type for GDI-Attribute of type ADValueType (above) -->

 <xsd:complexType name="OrderedValueAD">
   <xsd:complexContent>
    <xsd:restriction base="TOrderedValue">
     <xsd:sequence>
      <xsd:element name="Value" type="xsd:unsignedShort" default="0"/>
     </xsd:sequence>
    </xsd:restriction>
   </xsd:complexContent>
  </xsd:complexType>

  <!-- A simple read/write GDI-Attribute for the Function Object
       of type IFADInput -->

  <xsd:complexType name="PolarityType">
   <xsd:complexContent>
    <xsd:extension base="AttributeReadWrite">
     <xsd:sequence>
      <xsd:element name="OrderedValue" type="OrderedValuePolarity"
                   minOccurs="0"       maxOccurs="unbounded"/>
     </xsd:sequence>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>

  <!-- A simple Value-Type for GDI-Attribute of type PolarityType (above) -->

  <xsd:complexType name="OrderedValuePolarity">
   <xsd:complexContent>
    <xsd:restriction base="TOrderedValue">
     <xsd:sequence>
      <xsd:element name="Value" type="TePolarity" default="0"/>
     </xsd:sequence>
    </xsd:restriction>
   </xsd:complexContent>
  </xsd:complexType>

 <xsd:simpleType name="TePolarity">
   <xsd:restriction base="xsd:short">
    <xsd:enumeration value="0"/>
    <xsd:enumeration value="1"/>
   </xsd:restriction>
  </xsd:simpleType>


  <!-- A simple GDI-Parameter for the Function Object of type IFADInput -->

  <xsd:complexType name="ChannelType">
   <xsd:complexContent>
    <xsd:extension base="Parameter">
     <xsd:sequence>
      <xsd:element name="OrderedValue" type="OrderedValueChannel"
                   minOccurs="0"       maxOccurs="unbounded"/>
     </xsd:sequence>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>

  <!-- A simple Value-Type for GDI-Parameter of type ChannelType (above) -->

  <xsd:complexType name="OrderedValueChannel">
   <xsd:complexContent>
    <xsd:restriction base="TOrderedValue">
     <xsd:sequence>
      <xsd:element name="Value" type="xsd:unsignedShort" default="0"/>
     </xsd:sequence>
    </xsd:restriction>
   </xsd:complexContent>
  </xsd:complexType>

 </xsd:schema>
```

A.7.3 XML schema : DCDa2.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xsd:annotation>
     <xsd:appinfo source="http://www.asam.net/ISO20242-4/DCPTHeader.xsd">
     <DCPTHeader>
       <DCPTIdentification>DCD2</DCPTIdentification>
       <DCPTRevision>1.0</DCPTRevision>
       <DCPTName>DCD2</DCPTName>
       <DCPTSource>DCD2.xsd</DCPTSource>
       <DCPTClassID>DCD</DCPTClassID>
       <DCPTDate>2007-03-16</DCPTDate>
       <ISO20242Reference>
         <ISO20242Edition>1</ISO20242Edition>
         <Technology>ASAM-GDI</Technology>
       </ISO20242Reference>
     </DCPTHeader>
     </xsd:appinfo>
   </xsd:annotation>

   <xsd:include schemaLocation="GDICommon.xsd"/>

  <!-- Main type is GDI_DCD with contained Virtual Device Types.
       There may be many different Virtual Device Types contained. -->

  <xsd:complexType name="Driver02">
     <xsd:complexContent>
       <xsd:extension base="GDI_DCD">
         <xsd:sequence>
           <xsd:element name="myDevice02" type="Device02"
                        minOccurs="0"     maxOccurs="unbounded"/>
         </xsd:sequence>
       </xsd:extension>
     </xsd:complexContent>
   </xsd:complexType>

 <!-- Virtual Device Types have fixed numbers for identification in moduleID.
       This is a restriction of type VDInstance. If you want to avoid
       restriction, move attribute moduleId from type GDI_Module with a fixed
       number to type Device02 below, which then is an extension of GDI_Module
       and type Device02_ will be obsolete. -->

  <xsd:complexType name="Device02_">
     <xsd:complexContent>
      <xsd:restriction base="GDI_Module">
        <xsd:attribute name="moduleId"   type="xsd:unsignedShort"
                       use="required"    fixed="1002"/>
      </xsd:restriction>
     </xsd:complexContent>
   </xsd:complexType>

  <!-- This exemplary Virtual Device Type only contains one Function Object
       type without restricting the number of instances. -->

  <xsd:complexType name="Device02">
     <xsd:complexContent>
       <xsd:extension base="Device02_">
         <xsd:sequence>
           <xsd:element name="myFunction02" type="MyFunction02Type"
                        minOccurs="0"       maxOccurs="unbounded"/>
         </xsd:sequence>
       </xsd:extension>
     </xsd:complexContent>
   </xsd:complexType>
```

```
<!-- Function Object Types have fixed numbers for identification in funcID.
     This is a restriction of type GDI_Interface. If you want to avoid
     restriction, move attribute funcId from type GDI_Interface with a fixed
     number to type MyFunction02Type below, which then is an extension of
     GDI_Interface and type MyFunction02Type_ will be obsolete. -->

 <xsd:complexType name="MyFunction02Type_">
   <xsd:complexContent>
     <xsd:restriction base="GDI_Interface">
       <xsd:attribute name="funcId"   type="xsd:unsignedShort"
                      use="required"  fixed="1008"/>
     </xsd:restriction>
   </xsd:complexContent>
 </xsd:complexType>

<!-- This Function Object type example contains only the Create Parameter
     and one optional Operation. -->

 <xsd:complexType name="MyFunction02Type">
   <xsd:complexContent>
     <xsd:extension base="MyFunction02Type_">
       <xsd:sequence>
         <xsd:element name="myCRPar02"      type="cpType02"/>
         <xsd:element name="myOperation02"  type="myOperation02Type"
                      minOccurs="0"/>
       </xsd:sequence>
     </xsd:extension>
   </xsd:complexContent>
 </xsd:complexType>

<!-- The Create Parameter of this Function Object type is a simple structure
     for a typical poor serial interface with restricted speed and fixed
     bits per transmitted character. -->

<xsd:complexType name="cpType02">
   <xsd:complexContent>
     <xsd:extension base="CreateParameter">
       <xsd:sequence>
         <xsd:element name="Value" type="serialCom"/>
       </xsd:sequence>
     </xsd:extension>
   </xsd:complexContent>
 </xsd:complexType>

<xsd:complexType name="serialCom">
   <xsd:sequence>
     <xsd:element name="speed" type="speedType"/>
     <xsd:element name="length" type="xsd:integer" fixed="8"/>
   </xsd:sequence>
 </xsd:complexType>

 <xsd:simpleType name="speedType">
   <xsd:restriction base="xsd:int">
     <xsd:enumeration value="2400"/>
     <xsd:enumeration value="4800"/>
     <xsd:enumeration value="9600"/>
   </xsd:restriction>
 </xsd:simpleType>

<!-- This Operation has optional input and output values. -->

<xsd:complexType name="myOperation02Type_">
   <xsd:complexContent>
     <xsd:restriction base="GDI_Operation">
       <xsd:attribute name="operationId"  type="xsd:unsignedShort"
                      use="required"      fixed="1009"/>
     </xsd:restriction>
   </xsd:complexContent>
   </xsd:complexType>

  <xsd:complexType name="myOperation02Type">
     <xsd:complexContent>
       <xsd:extension base="myOperation02Type_">
         <xsd:sequence>
           <xsd:element name="OutValue" type="op02outType"  minOccurs="0"/>
           <xsd:element name="InValue"  type="op02inType"   minOccurs="0"/>
         </xsd:sequence>
       </xsd:extension>
     </xsd:complexContent>
   </xsd:complexType>

  <!-- The input parameters for this operation may be a sequence of ordered
       values. If accordingly instantiated, from this follow multiple requests
       (and confirmations) of service VDSI_Execute (ISO 20242.3) -->

   <xsd:complexType name="op02inType">
     <xsd:complexContent>
       <xsd:extension base="OperationInParameter">
         <xsd:sequence>
           <xsd:element name="Input"    type="orderedvalueop02in"
                                        maxOccurs="unbounded"/>
         </xsd:sequence>
       </xsd:extension>
     </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="orderedvalueop02in">
     <xsd:complexContent>
       <xsd:restriction base="TOrderedValue">
         <xsd:sequence>
           <xsd:element name="Value" type="xsd:double"/>
         </xsd:sequence>
       </xsd:restriction>
     </xsd:complexContent>
   </xsd:complexType>

  <!-- The output parameters for this operation is a special type "void",
       which defines, that the accordingly XML element has to be empty.
       Implementations of service VDSI_Execute (e.g. Annex A of ISO 20242.3)
       will define this case also (e.g. NULL pointer). It simply means,
       that there will be no output value for this operation -->

   <xsd:complexType name="op02outType">
     <xsd:complexContent>
       <xsd:extension base="OperationOutParameter">
         <xsd:sequence>
           <xsd:element name="Value" type="void"/>
         </xsd:sequence>
       </xsd:extension>
     </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="void"/>

 </xsd:schema>
```

A.8 Example of CCD

A.8.1 General

CCDa.xsd is an XML schema example of CCD that describes an XML schema for the parameterization of two device drivers.

A.8.2 XML schema : CCDa.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             elementFormDefault="qualified" attributeFormDefault="unqualified">

   <xsd:annotation>
     <xsd:appinfo source="http://www.asam.net/ISO20242-4/DCPTHeader.xsd">
     <DCPTHeader>
     <DCPTIdentification>CCD1</DCPTIdentification>
     <DCPTRevision>1.0</DCPTRevision>
     <DCPTName>CCDa</DCPTName>
     <DCPTSource>CCDa.xsd</DCPTSource>
     <DCPTClassID>CCD</DCPTClassID>
     <DCPTDate>2009-03-16</DCPTDate>
     <ISO20242Reference>
       <ISO20242Edition>1</ISO20242Edition>
       <Technology>ASAM-GDI</Technology>
     </ISO20242Reference>
     </DCPTHeader>
     </xsd:appinfo>
   </xsd:annotation>

   <!-- A coodinator capability description contains the capability
        descriptions of all devices, which will be usable -->
   <xsd:include schemaLocation="DCDa1.xsd"/>
   <xsd:include schemaLocation="DCDa2.xsd"/>

   <!--  * ISO15745 Profile Root *  -->
   <xsd:element name="ISO15745Profile">
     <xsd:complexType>
       <xsd:sequence>
         <xsd:element ref="ProfileHeader"/>
         <xsd:element ref="ProfileBody"/>
       </xsd:sequence>
     </xsd:complexType>
   </xsd:element>

     <!-- * HEADER DATA TYPES * -->
   <xsd:element name="ProfileHeader">
     <xsd:complexType>
       <xsd:sequence>
         <xsd:element name="ProfileIdentification"  type="xsd:string"/>
         <xsd:element name="ProfileRevision"        type="xsd:string"/>
         <xsd:element name="ProfileName"            type="xsd:string"/>
         <xsd:element name="ProfileSource"          type="xsd:string"/>
         <xsd:element name="ProfileClassID"         type="ProfileClassID_DataType"
                                                    fixed="InformationExchange"/>
         <xsd:element name="ProfileDate"            type="xsd:date"
                      minOccurs="0"/>
         <xsd:element name="AdditionalInformation"  type="xsd:anyURI"
                      minOccurs="0"/>
         <xsd:element name="ISO15745Reference"      type="ISO15745Reference_DataType"/>
         <xsd:element name="IASInterfaceType"       type="IASInterface_DataType"
                                                    fixed="CSI"
                      minOccurs="0"                 maxOccurs="unbounded"/>
       </xsd:sequence>
     </xsd:complexType>
   </xsd:element>

   <xsd:complexType name="ISO15745Reference_DataType">
     <xsd:sequence>
       <xsd:element name="ISO15745Part"       type="xsd:string" fixed="1"/>
       <xsd:element name="ISO15745Edition"    type="xsd:string" fixed="1"/>
       <xsd:element name="ProfileTechnology"  type="xsd:string" fixed="None"/>
     </xsd:sequence>
   </xsd:complexType>
```

```
  <xsd:simpleType name="ProfileClassID_DataType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="AIP"/>
      <xsd:enumeration value="Process"/>
      <xsd:enumeration value="InformationExchange"/>
      <xsd:enumeration value="Resource"/>
      <xsd:enumeration value="Device"/>
      <xsd:enumeration value="CommunicationNetwork"/>
      <xsd:enumeration value="Equipment"/>
      <xsd:enumeration value="Human"/>
      <xsd:enumeration value="Material"/>
    </xsd:restriction>
  </xsd:simpleType>

   <xsd:simpleType name="IASInterface_DataType">
    <xsd:union>
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="CSI"/>
          <xsd:enumeration value="HCI"/>
          <xsd:enumeration value="ISI"/>
          <xsd:enumeration value="API"/>
          <xsd:enumeration value="CMI"/>
          <xsd:enumeration value="ESI"/>
          <xsd:enumeration value="FSI"/>
          <xsd:enumeration value="MTI"/>
          <xsd:enumeration value="SEI"/>
          <xsd:enumeration value="USI"/>
        </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:length value="4"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:union>
  </xsd:simpleType>

   <!-- * BODY SECTION *  -->
  <xsd:element name="ProfileBody">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="CCD" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="DCD1"     type="Driver01"
                           minOccurs="0"   maxOccurs="unbounded"/>
              <xsd:element name="DCD2"     type="Driver02"
                           minOccurs="0"   maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="category" type="Category"
                           use="required"  fixed="CCD"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

A.9 Example of PID

A.9.1 General

SamplePIDa.xml is an XML data example of PID that uses CCDa.xsd as XML schema and describes parameter instances of the coordinator and the device drivers.

A.9.2 XML : SamplePIDa.xml

```
<?xml version="1.0" encoding="UTF-8"?>
 <ISO15745Profile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:noNamespaceSchemaLocation="CCDa.xsd">
   <ProfileHeader>
     <ProfileIdentification>ExampleOfPID</ProfileIdentification>
     <ProfileRevision>1.0</ProfileRevision>
     <ProfileName>Example of PID</ProfileName>
     <ProfileSource>SamplePIDa.xml</ProfileSource>
     <ProfileClassID>InformationExchange</ProfileClassID>
     <ProfileDate>2009-03-16</ProfileDate>
     <AdditionalInformation>http://www.asam.net/ISO20242-4/XPID</AdditionalInformation>
     <ISO15745Reference>
       <ISO15745Part>1</ISO15745Part>
       <ISO15745Edition>1</ISO15745Edition>
       <ProfileTechnology>None</ProfileTechnology>
     </ISO15745Reference>
     <IASInterfaceType>CSI</IASInterfaceType>
   </ProfileHeader>

   <ProfileBody>
    <CCD category="CCD">

     <DCD1 driverVersion="1" dllPath="ndAD.dll" category="DCD">

       <!-- initialize device 1-->
       <myDevice01 initOrder="1" moduleId="1000" category="MODULE">
         <NumOfChannel category="CREATEPARAMETER">
           <Value>255</Value>
         </NumOfChannel>

         <!-- initialize fnADInput -->
         <fnADInput funcId="1077" initOrder="2" category="INTERFACE">
           <Interrupt category="CREATEPARAMETER">
             <Value>5</Value>
           </Interrupt>

           <!-- initialize parameter Channel -->
           <Channel initOrder="3" category="PARAMETER" readonly="false">
           <!-- write channel several times -->
           <OrderedValue initOrder="4">
             <Value>0</Value>
           </OrderedValue>
           <OrderedValue initOrder="14">
             <Value>1</Value>
           </OrderedValue>
           <OrderedValue initOrder="24">
             <Value>2</Value>
           </OrderedValue>
           </Channel>

           <!-- initialize the read-only attribute ADValue -->
           <ADValue initOrder="3" category="ATTRIBUTE"
                      readonly="true" infReport="true"/>
         </fnADInput>
       </myDevice01>
     </DCD1>
     <DCD2 driverVersion="1" dllPath="dcd2.dll" category="DCD">

        <!-- initialize device 02 -->
        <myDevice02 initOrder="1" moduleId="1002" category="MODULE">

         <!-- initialize myFunction02 -->
         <myFunction02 funcId="1008" initOrder="2" category="INTERFACE">
       <myCRPar02 category="CREATEPARAMETER">
         <Value>
           <speed>4800</speed>
           <length>8</length>
         </Value>
       </myCRPar02>

           <!-- initialize myOperation02 -->
       <myOperation02 initOrder="3" category="OPERATION" operationId="1009">

       <!-- execute the operation several times -->
       <InValue category="IN">
         <Input initOrder="13">
           <Value>7.0</Value>
         </Input>
         <Input initOrder="23">
           <Value>14.0</Value>
         </Input>
         <Input initOrder="33">
           <Value>24.0</Value>
         </Input>
       </InValue>
       </myOperation02>

     </myFunction02>
        </myDevice02>
     </DCD2>
     </CCD>
   </ProfileBody>
 </ISO15745Profile>
```

Annex B
(informative)

Device capability profile templates for manufacturing applications

B.1 General

The MICX (Manufacturing Information Collaboration with XML technology) specifies the information collaboration using XML for manufacturing applications. This annex defines the MICX-specific device capability profile template of the service interface as information collaboration using OASIS PPS[4],[5], and provides examples of DCD, CCD and PID.

B.2 MICX profile model

B.2.1 General

The MICX-specific profile model contains all information that is necessary for the device capability description and all information that is necessary for parameterization. Figure B.1 shows the class diagram of MICX-specific device capability profile template.

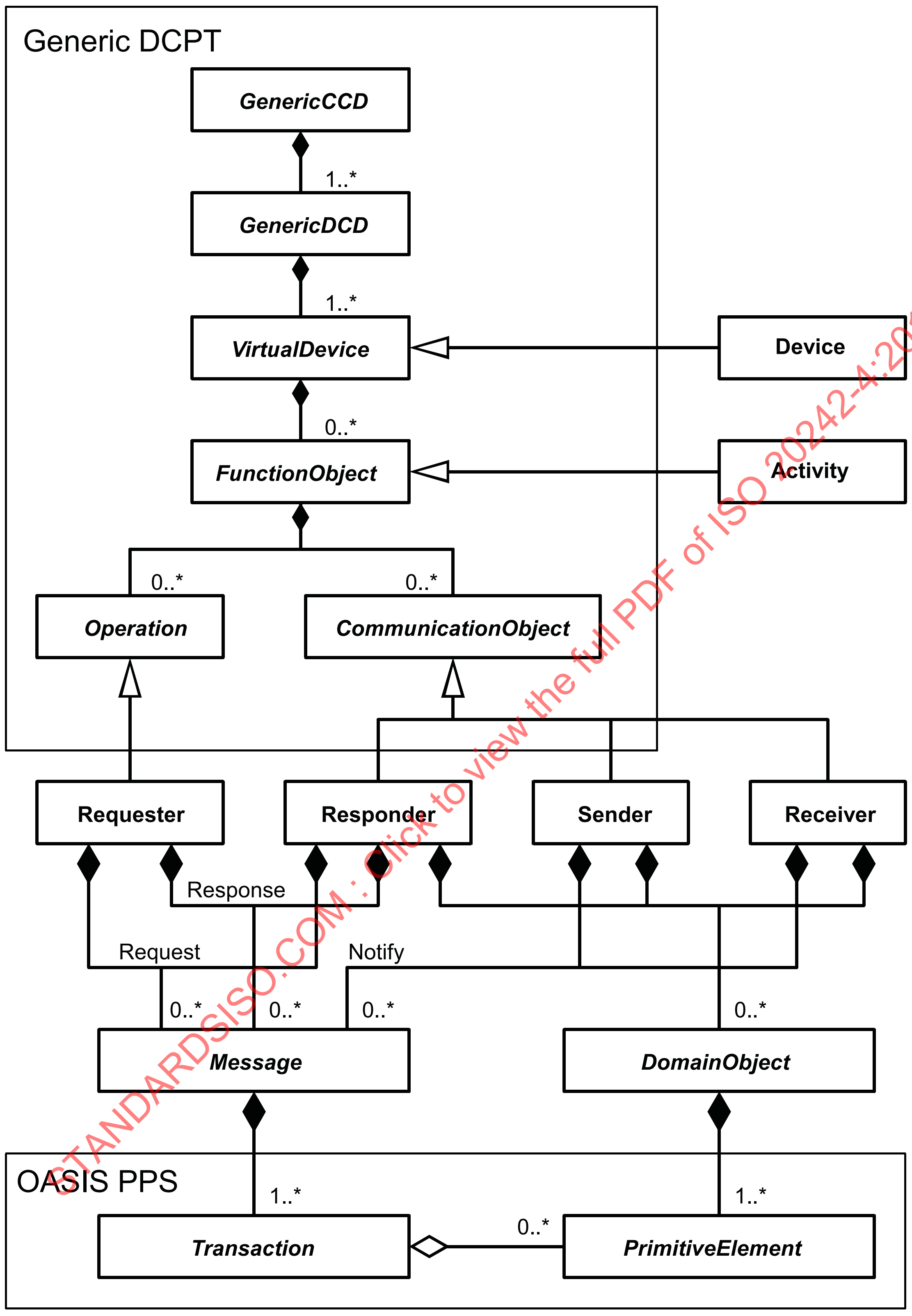


Figure B.1 — Class diagram of MICX-specific DCPT model

B.2.2 Device

The device class shows the capabilities of the MICX-specific virtual device. It inherits the virtual device class. A device-specific device class inherits it and defines the capabilities of the device-specific virtual device.

B.2.3 Activity

The activity class shows the capabilities of MICX-specific activity of the device. It inherits the function object class, and is an abstract class. A device-specific activity class inherits it and defines the capabilities of the device-specific activity.

B.2.4 Requester

The requester class shows the capabilities of the requester used by the MICX-specific activity. It inherits the operation class. It has zero or more message classes of the request message which can be sent and zero or more message classes of the response message which can be received. The requester is defined in OASIS PPS-2[5].

B.2.5 Responder

The responder class shows the capabilities of the responder used by the MICX-specific activity. It inherits the communication object class. It has zero or more message classes of the request message which can be received and zero or one message class of the response message which can be sent. It has zero or more domain object classes to access the data of the request and the response. The responder is defined in OASIS PPS-2[5].

B.2.6 Sender

The sender class shows the capabilities of the sender used by the MICX-specific activity. It inherits the communication object class. It has zero or more message classes of the notify message which can be sent. It has zero or more domain object classes to write the data of the notify message. The sender is defined in OASIS PPS-2[5].

B.2.7 Receiver

The receiver class shows the capabilities of the receiver used by the MICX-specific activity. It inherits the communication object class. It has zero or more message classes of the notify message which can be received. It has zero or more domain object classes to read the data of the notify message. The receiver is defined in OASIS PPS-2[5].

B.2.8 Message

The message class shows the capabilities of the message which the MICX-specific activity can send and receive. It has one or more transaction classes.

B.2.9 Domain object

The domain object class shows the capabilities of the domain object which the responder, the sender or the receiver has. It has one or more primitive element classes.

B.2.10 Transaction

The transaction class shows the capabilities of the transaction of the message which the MICX-specific activity can send and receive. It has the primitive element classes. Each specific transaction class inherits the abstract transaction class. The transaction is defined in OASIS PPS-2[5].

B.2.11 Primitive element

The primitive element class shows the capabilities of the OASIS PPS primitive element of the message which the MICX-specific activity can send and receive. There are nine types of primitive element in OASIS PPS; plan, order, party, task, operation, lot, resource, process and item. Primitive elements are defined in OASIS PPS-1[4].

B.3 Additional type identification to generic DCPT

Table B.1 contains technology-specific type identifications for communication objects.

Table B.1 — Type identification for communication objects

Element of technology-specific DCPT	Contents of XML attribute "category"[a]
Responder	RESPONDER
Sender	SENDER
Receiver	RECEIVER

[a] With MICX, the content of attribute "category" has no relation to the status of the virtual device. Any content will have a similar meaning as content ATTRIBUTE has with ASAM GDI as described in Annex A.

B.4 MICX-specific device capability profile template

B.4.1 General

MICXCommon.xsd is the XML schema of MICX-specific DCPT that extends XML schema of generic DCPT in 6.3 in accordance with the MICX-specific DCPT model of Figure B.1. It imports XML schemas in OASIS PPS Part 1 and Part 2.

B.4.2 XML schema : MICXCommon.xsd

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns="http://www.mstc.or.jp/micx/ISO20242-4/MICX"
            xmlns:dcpt="http://www.osi.ch/iso/ISO20242-4/GenericDCPT"
            xmlns:pps="http://docs.oasis-open.org/pps/ns/core-elements"
            xmlns:ppst="http://docs.oasis-open.org/pps/ns/transaction-messages"
            targetNamespace="http://www.mstc.or.jp/micx/ISO20242-4/MICX"
            elementFormDefault="qualified">

  <xsd:annotation>
    <xsd:appinfo source="http://www.osi.ch/iso/ISO20242-4/DCPTHeader.xsd">
      <DCPTHeader>
        <DCPTIdentification>MICXCommon</DCPTIdentification>
        <DCPTRevision>1.0</DCPTRevision>
        <DCPTName>MICX Specific DCPT</DCPTName>
        <DCPTSource>MICXCommon.xsd</DCPTSource>
        <DCPTClassID>InformationExchangeTemplate</DCPTClassID>
        <DCPTDate>2009-02-25</DCPTDate>
        <DCPTVender>FAOP</DCPTVender>
        <TechnologyReference>
          <Technology>FAOP-MICX</Technology>
          <Revision>1.0</Revision>
        </TechnologyReference>
      </DCPTHeader>
    </xsd:appinfo>
  </xsd:annotation>
```

```
<!--  * Import GenericDCPT,*  -->
<xsd:import namespace="http://www.osi.ch/iso/ISO20242-4/GenericDCPT"
            schemaLocation="GenericDCPT.xsd"/>

<!--  * Import PPS core elements,*  -->
<xsd:import namespace="http://docs.oasis-open.org/pps/ns/core-elements"
            schemaLocation="pps-core-elements-1.0.xsd"/>

<!--  * Import PPS transaction messages,*  -->
<xsd:import namespace="http://docs.oasis-open.org/pps/ns/transaction-messages"
            schemaLocation="pps-transaction-messages-1.0.xsd"/>

<!-- Valid content for XML-Attribut category (kind of DCD element) -->
<xsd:simpleType name="categoryType">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="CCD"/>
  <xsd:enumeration value="DCD"/>
  <xsd:enumeration value="MODULE"/>
  <xsd:enumeration value="INTERFACE"/>
  <xsd:enumeration value="OPERATION"/>
  <xsd:enumeration value="RESPONDER"/>
  <xsd:enumeration value="SENDER"/>
  <xsd:enumeration value="RECEIVER"/>
 </xsd:restriction>
</xsd:simpleType>

<!-- Valid message type (kind of Messge element) -->
<xsd:simpleType name="messageType">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="REQUEST"/>
  <xsd:enumeration value="RESPONSE"/>
  <xsd:enumeration value="NOTIFY"/>
 </xsd:restriction>
</xsd:simpleType>

<!-- * Elements Declaration *  -->
<xsd:element name="RequestMessage" type="MessageType" abstract="true"/>
<xsd:element name="ResponseMessage" type="MessageType" abstract="true"/>
<xsd:element name="NotifyMessage" type="MessageType" abstract="true"/>
<xsd:element name="DomainObject" type="DomainObjectType" abstract="true"/>
<xsd:element name="Transaction" type="ppst:TransactionType" abstract="true"/>
<xsd:element name="PrimitiveElement" type="pps:PrimitiveType" abstract="true"/>

<!-- * Type Definition *  -->
<xsd:complexType name="DCDType">
 <xsd:complexContent>
  <xsd:extension base="dcpt:GenericDCDType"/>
 </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="DeviceType">
 <xsd:complexContent>
  <xsd:extension base="dcpt:VirtualDeviceType"/>
 </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="ActivityType">
 <xsd:complexContent>
  <xsd:extension base="dcpt:FunctionObjectType"/>
 </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="RequesterType">
 <xsd:complexContent>
  <xsd:extension base="dcpt:OperationType">
   <xsd:sequence>
    <xsd:element ref="RequestMessage" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="ResponseMessage" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="ResponderType">
  <xsd:complexContent>
   <xsd:extension base="dcpt:CommunicationObjectType">
    <xsd:sequence>
     <xsd:element ref="RequestMessage" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="ResponseMessage" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="DomainObject" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="category" type="categoryType" use="required" fixed="RESPONDER"/>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="SenderType">
  <xsd:complexContent>
   <xsd:extension base="dcpt:CommunicationObjectType">
    <xsd:sequence>
     <xsd:element ref="NotifyMessage" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="DomainObject" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="category" type="categoryType" use="required" fixed="SENDER"/>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="ReceiverType">
  <xsd:complexContent>
   <xsd:extension base="dcpt:CommunicationObjectType">
    <xsd:sequence>
     <xsd:element ref="NotifyMessage" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="DomainObject" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="category" type="categoryType" use="required" fixed="RECEIVER"/>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="MessageType">
  <xsd:choice minOccurs="0">
   <xsd:element ref="Transaction" maxOccurs="unbounded"/>
  </xsd:choice>
  <xsd:attribute name="name" type="xsd:string"/>
  <xsd:attribute name="message" type="messageType" use="required"/>
 </xsd:complexType>

 <xsd:complexType name="DomainObjectType">
  <xsd:sequence minOccurs="0">
   <xsd:element ref="PrimitiveElement" maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType>
  <xsd:attribute name="name" type="xsd:string"/>
</xsd:schema>
```

B.5 Example of DCD schema

B.5.1 General

DCDb.xsd is an XML schema example of DCD that extends the XML schema of MICX-specific DCPT in Clause B.4 in accordance with the device capabilities of the device driver.

B.5.2 XML schema : DCDb.xsd

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns="http://www.mstc.or.jp/micx/ISO20242-4/b/Example"
            xmlns:micx="http://www.mstc.or.jp/micx/ISO20242-4/MICX"
            xmlns:pps="http://docs.oasis-open.org/pps/ns/core-elements"
            xmlns:ppst="http://docs.oasis-open.org/pps/ns/transaction-messages"
            targetNamespace="http://www.mstc.or.jp/micx/ISO20242-4/b/Example"
            elementFormDefault="qualified">

 <xsd:annotation>
  <xsd:appinfo source="http://www.osi.ch/iso/ISO20242-4/DCPTHeader.xsd">
    <DCPTHeader>
        <DCPTIdentification>DCD</DCPTIdentification>
        <DCPTRevision>1.0</DCPTRevision>
        <DCPTName>DCD Example</DCPTName>
        <DCPTSource>DCDb.xsd</DCPTSource>
        <DCPTClassID>InformationExchangeTemplate</DCPTClassID>
        <DCPTDate>2009-02-25</DCPTDate>
        <DCPTVender>FAOP</DCPTVender>
        <TechnologyReference>
          <Technology>FAOP-MICX</Technology>
          <Revision>1.0</Revision>
        </TechnologyReference>
      </DCPTHeader>
    </xsd:appinfo>
  </xsd:annotation>

 <!--  *  Import MICXCommon.xsd,*  -->
 <xsd:import namespace="http://www.mstc.or.jp/micx/ISO20242-4/MICX"
             schemaLocation="MICXCommon.xsd"/>

 <!--  * Import PPS core elements,*  -->
 <xsd:import namespace="http://docs.oasis-open.org/pps/ns/core-elements"
             schemaLocation="pps-core-elements-1.0.xsd"/>

 <!--  * Import PPS transaction messages,*  -->
 <xsd:import namespace="http://docs.oasis-open.org/pps/ns/transaction-messages"
             schemaLocation="pps-transaction-messages-1.0.xsd"/>

 <!-- * Elements Declaration * -->
 <xsd:element name="Device" type="DeviceType"/>
 <xsd:element name="ProductionControl" type="ProductionControlType"/>
 <xsd:element name="LoadRecipe" type="LoadRecipeType"/>
 <xsd:element name="ExecutionControl" type="ExecutionControlType"/>
 <xsd:element name="GetInformation" type="GetInformationType"/>
 <xsd:element name="EventReport" type="EventReportType"/>

 <!-- * DCD Type Definition * -->
 <xsd:complexType name="DCDType">
  <xsd:sequence>
   <xsd:element ref="Device" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string"/>
  <xsd:attribute name="category" type="micx:categoryType" use="required" fixed="DCD"/>
 </xsd:complexType>

 <!-- * Device Type Definition * -->
 <xsd:complexType name="DeviceType">
  <xsd:sequence>
   <xsd:element ref="ProductionControl" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string"/>
  <xsd:attribute name="category" type="micx:categoryType" use="required" fixed="MODULE"/>
 </xsd:complexType>
 <!-- * ProductionControl(Activity) Type Definition * -->
 <xsd:complexType name="ProductionControlType">
  <xsd:sequence>
   <xsd:element ref="LoadRecipe" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="ExecutionControl" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="GetInformation" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="EventReport" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string"/>
  <xsd:attribute name="category" type="micx:categoryType" use="required"
fixed="INTERFACE"/>
 </xsd:complexType>

 <!-- * LoadRecipe(Requester) Type Definition * -->
 <xsd:complexType name="LoadRecipeType">
  <xsd:sequence>
   <xsd:element ref="LoadRecipeRequest" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="LoadRecipeResponse" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="category" type="micx:categoryType" use="required"
fixed="OPERATION"/>
 </xsd:complexType>

 <!-- * ExecutionControl(Requester) Type Definition * -->
 <xsd:complexType name="ExecutionControlType">
  <xsd:sequence>
   <xsd:element ref="ExecutionRequest" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="ExecutionResponse" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string"/>

  <xsd:attribute name="category" type="micx:categoryType" use="required"
fixed="OPERATION"/>
 </xsd:complexType>

 <!-- * GetInformation(Requester) Type Definition * -->
 <xsd:complexType name="GetInformationType">
  <xsd:sequence>
   <xsd:element ref="GetInformationRequest" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="GetInformationResponse" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string"/>
  <xsd:attribute name="category" type="micx:categoryType"
               use="required" fixed="OPERATION"/>
 </xsd:complexType>

 <!-- * EventReportType(Receiver) Type Definition * -->
 <xsd:complexType name="EventReportType">
  <xsd:sequence>
   <xsd:element ref="NotifyEvent" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element ref="EventObject" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string"/>
  <xsd:attribute name="category" type="micx:categoryType"
               use="required" fixed="RECEIVER"/>
 </xsd:complexType>

 <!-- * EventObject Type Definition * -->
 <xsd:element name="EventObject">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="EquipmentObject" type="EquipmentType"
               minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="PersonnelObject" type="PersonnelType"
               minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="ExecutionObject" type="ExecutionType"
               minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>

 <!-- Definition Type of Equipment elements extended on primitive element -->
 <!-- Equipment element -->
 <xsd:element name="Equipment" type="EquipmentType" substitutionGroup="pps:Resource"/>
 <!-- Equipment Type -->
 <xsd:complexType name="EquipmentType">
  <xsd:complexContent>
   <xsd:restriction base="pps:PrimitiveType">
    <xsd:sequence>
     <xsd:element ref="pps:Capacity" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Spec" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Event" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Description" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Author" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Date" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:string" use="required"/>
    <xsd:attribute name="key" type="xsd:long"/>
    <xsd:attribute name="name" type="xsd:string" fixed="Equipment"/>
    <xsd:attribute name="type" type="xsd:string"/>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

 <!-- Definition Type of Personnel element extended on primitive element -->
 <!-- Personnel element -->
 <xsd:element name="Personnel" type="PersonnelType" substitutionGroup="pps:Resource"/>
 <!-- Personnel Type -->
 <xsd:complexType name="PersonnelType">
  <xsd:complexContent>
   <xsd:restriction base="pps:PrimitiveType">
    <xsd:sequence>
     <xsd:element ref="pps:Capacity" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Spec" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Event" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Description" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Author" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Date" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:string" use="required"/>
    <xsd:attribute name="key" type="xsd:long"/>
    <xsd:attribute name="name" type="xsd:string" fixed="Personnel"/>
    <xsd:attribute name="type" type="xsd:string"/>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

 <!-- Definition Type of Recipe element extended on primitive element -->
 <!-- Recipe element -->
 <xsd:element name="Recipe" type="RecipeType" substitutionGroup="pps:Process"/>
 <!-- customer Type -->
 <xsd:complexType name="RecipeType">
  <xsd:complexContent>
   <xsd:restriction base="pps:PrimitiveType">
    <xsd:sequence>
     <xsd:element ref="pps:Produce" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Consume" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Assign" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Spec" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Description" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Author" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Date" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:string" use="required"/>
    <xsd:attribute name="key" type="xsd:long"/>
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="type" type="xsd:string" fixed="Recipe"/>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

 <!-- Definition Type of Work element extended on primitive element -->
 <!-- Work element -->
 <xsd:element name="Work" type="WorkType" substitutionGroup="pps:Item"/>
 <!-- customer Type -->
 <xsd:complexType name="WorkType">
  <xsd:complexContent>
   <xsd:restriction base="pps:PrimitiveType">
    <xsd:sequence>
     <xsd:element ref="pps:Produce" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Consume" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Spec" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Description" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Author" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Date" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:string" use="required"/>
    <xsd:attribute name="key" type="xsd:long"/>
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="parent" type="xsd:string"/>
    <xsd:attribute name="type" type="xsd:string" fixed="Work"/>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

 <!-- Definition Type of Execution Order element extended on primitive element -->
 <!-- Execution element -->
 <xsd:element name="Execution" type="ExecutionType" substitutionGroup="pps:Operation"/>
 <!-- Execution Type -->
 <xsd:complexType name="ExecutionType">
  <xsd:complexContent>
   <xsd:restriction base="pps:PrimitiveType">
    <xsd:sequence>
     <xsd:element ref="pps:Produce" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Consume" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Assign" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Progress" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Spec" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Start" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:End" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Event" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Description" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Author" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="pps:Date" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:string" use="required"/>
    <xsd:attribute name="key" type="xsd:long"/>
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="type" type="xsd:string" fixed="Operation"/>
    <xsd:attribute name="status" type="xsd:string"/>
    <xsd:attribute name="process" type="xsd:string"/>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

 <!-- Definition Type of Accepted Work element extended on relational element -->
 <!—AcceptedWork element -->
 <xsd:element name="AcceptedWork" type="AcceptedWorkType"
              substitutionGroup="pps:Produce"/>
 <!—AcceptedWork Type -->
 <xsd:complexType name="AcceptedWorkType">
  <xsd:complexContent>
   <xsd:restriction base="pps:RelationalType">
```

```
    <xsd:sequence>
     <xsd:element ref="pps:Qty" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="type" type="xsd:string" fixed="Accepted"/>
    <xsd:attribute name="lot" type="xsd:string"/>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

 <!-- Definition Type of DefectiveWork element extended on relational element -->
 <!-- DefectiveWork element -->
 <xsd:element name="DefectiveWork" type="DefectiveWorkType"
              substitutionGroup="pps:Produce"/>
 <!—DefectiveWork Type -->
 <xsd:complexType name="DefectiveWorkType">
  <xsd:complexContent>
   <xsd:restriction base="pps:RelationalType">
    <xsd:sequence>
     <xsd:element ref="pps:Qty" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="type" type="xsd:string" fixed="Defective"/>
    <xsd:attribute name="lot" type="xsd:string"/>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

 <!—Definition Type of Assigned Equipment element extended on relational element -->
 <!—AssignedEquipment element -->
 <xsd:element name="AssignedEquipment" type="AssignedEquipmentType"
substitutionGroup="pps:Assign"/>
 <!—AssignedEquipment Type -->
 <xsd:complexType name="AssignedEquipmentType">
  <xsd:complexContent>
   <xsd:restriction base="pps:RelationalType">
    <xsd:sequence>
     <xsd:element ref="pps:Qty" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="type" type="xsd:string" fixed="Equipment"/>
    <xsd:attribute name="resource" type="xsd:string"/>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

 <!—Definition Type of OrderName element extended on specific element -->
 <!—OrderName element -->
 <xsd:element name="OrderName" type="OrderNameType" substitutionGroup="pps:Spec"/>
 <!—OrderName Type -->
 <xsd:complexType name="OrderNameType">
  <xsd:complexContent>
   <xsd:restriction base="pps:SpecificType">
    <xsd:sequence>
     <xsd:element ref="pps:Char" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="type" type="xsd:string" fixed="Order"/>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

 <!—Definition Type of Length element extended on specific element -->
 <!-- Length element -->
 <xsd:element name="Length" type="LengthType" substitutionGroup="pps:Spec"/>
 <xsd:element name="Qty" substitutionGroup="pps:Qty"/>
 <!-- Length Type -->
 <xsd:complexType name="LengthType">
  <xsd:complexContent>
   <xsd:restriction base="pps:SpecificType">
    <xsd:sequence>
     <xsd:element ref="pps:Qty" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="type" type="xsd:string" fixed="Length"/>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

 <!—Message -->
 <!—Definition Request Message of LoadRecipe -->
 <xsd:element name="LoadRecipeRequest">
  <xsd:complexType>
   <xsd:complexContent>
    <xsd:extension base="micx:MessageType">
     <xsd:sequence>
      <xsd:element name="RecipeRecord">
       <xsd:complexType>
        <xsd:complexContent>
         <xsd:restriction base="ppst:TransactionType">
          <xsd:sequence>
           <xsd:element ref="ppst:App" minOccurs="0"/>
           <xsd:element ref="ppst:Condition" minOccurs="0" maxOccurs="unbounded"/>
           <xsd:choice minOccurs="0">
            <xsd:element ref="pps:Item" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="pps:Process" minOccurs="0" maxOccurs="unbounded"/>
           </xsd:choice>
          </xsd:sequence>
          <xsd:attribute name="id" type="xsd:string" use="required"/>
          <xsd:attribute name="action" type="xsd:string" fixed="Add"/>
          <xsd:attribute name="transaction" type="xsd:string"/>
          <xsd:attribute name="profile" type="xsd:string"/>
          <xsd:attribute name="confirm" type="xsd:string"/>
          <xsd:attribute name="create" type="xsd:dateTime"/>
          <xsd:attribute name="sender" type="xsd:string"/>
          <xsd:attribute name="description" type="xsd:string"/>
         </xsd:restriction>
        </xsd:complexContent>
       </xsd:complexType>
      </xsd:element>
     </xsd:sequence>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>
 </xsd:element>

 <!—Definition Response Message of LoadRecipe -->
 <xsd:element name="LoadRecipeResponse">
  <xsd:complexType>
   <xsd:complexContent>
    <xsd:extension base="micx:MessageType">
     <xsd:sequence>
      <xsd:element name="RecipeRecord">
       <xsd:complexType>
        <xsd:complexContent>
         <xsd:restriction base="ppst:TransactionType">
          <xsd:sequence>
           <xsd:element ref="ppst:Error" minOccurs="0" maxOccurs="unbounded"/>
           <xsd:element ref="ppst:App" minOccurs="0"/>
           <xsd:choice minOccurs="0">
            <xsd:element ref="pps:Item" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="pps:Process" minOccurs="0" maxOccurs="unbounded"/>
           </xsd:choice>
          </xsd:sequence>
          <xsd:attribute name="id" type="xsd:string" use="required"/>
          <xsd:attribute name="action" type="xsd:string" fixed="Confirm"/>
          <xsd:attribute name="transaction" type="xsd:string"/>
          <xsd:attribute name="profile" type="xsd:string"/>
          <xsd:attribute name="create" type="xsd:dateTime"/>
          <xsd:attribute name="sender" type="xsd:string"/>
          <xsd:attribute name="description" type="xsd:string"/>
         </xsd:restriction>
        </xsd:complexContent>
       </xsd:complexType>
      </xsd:element>
     </xsd:sequence>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>
 </xsd:element>

 <!—Definition Request Message of Execution -->
 <xsd:element name="ExecutionRequest">
  <xsd:complexType>
   <xsd:complexContent>
    <xsd:extension base="micx:MessageType">
     <xsd:sequence>
      <xsd:element name="ExecutionOrder">
       <xsd:complexType>
        <xsd:complexContent>
         <xsd:restriction base="ppst:TransactionType">
          <xsd:sequence>
           <xsd:element ref="ppst:App" minOccurs="0"/>
           <xsd:element ref="ppst:Condition" minOccurs="0" maxOccurs="unbounded"/>
           <xsd:choice minOccurs="0">
            <xsd:element ref="pps:Operation" maxOccurs="unbounded"/>
           </xsd:choice>
          </xsd:sequence>
          <xsd:attribute name="id" type="xsd:string" use="required"/>
          <xsd:attribute name="action" type="xsd:string" fixed="Add"/>
          <xsd:attribute name="transaction" type="xsd:string"/>
          <xsd:attribute name="profile" type="xsd:string"/>
          <xsd:attribute name="confirm" type="xsd:string"/>
          <xsd:attribute name="create" type="xsd:dateTime"/>
          <xsd:attribute name="sender" type="xsd:string"/>
          <xsd:attribute name="description" type="xsd:string"/>
         </xsd:restriction>
        </xsd:complexContent>
       </xsd:complexType>
      </xsd:element>
     </xsd:sequence>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>
 </xsd:element>

 <!—Definition Response Message of Execution -->
 <xsd:element name="ExecutionResponse">
  <xsd:complexType>
   <xsd:complexContent>
    <xsd:extension base="micx:MessageType">
     <xsd:sequence>
      <xsd:element name="ExecutionOrder">
       <xsd:complexType>
        <xsd:complexContent>
         <xsd:restriction base="ppst:TransactionType">
          <xsd:sequence>
           <xsd:element ref="ppst:Error" minOccurs="0" maxOccurs="unbounded"/>
           <xsd:element ref="ppst:App" minOccurs="0"/>
           <xsd:choice>
            <xsd:choice minOccurs="0">
             <xsd:element ref="pps:Operation" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:choice>
           </xsd:choice>
          </xsd:sequence>
          <xsd:attribute name="id" type="xsd:string" use="required"/>
          <xsd:attribute name="action" type="xsd:string" fixed="Confirm"/>
          <xsd:attribute name="transaction" type="xsd:string"/>
          <xsd:attribute name="profile" type="xsd:string"/>
          <xsd:attribute name="create" type="xsd:dateTime"/>
          <xsd:attribute name="sender" type="xsd:string"/>
          <xsd:attribute name="description" type="xsd:string"/>
         </xsd:restriction>
        </xsd:complexContent>
       </xsd:complexType>
      </xsd:element>
     </xsd:sequence>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>
 </xsd:element>

 <!—Definition Request Message of GetInformation -->
 <xsd:element name="GetInformationRequest">
  <xsd:complexType>
   <xsd:complexContent>
    <xsd:extension base="micx:MessageType">
     <xsd:choice>
      <xsd:element name="ExecutionOrder">
       <xsd:complexType>
        <xsd:complexContent>
         <xsd:restriction base="ppst:TransactionType">
          <xsd:sequence>
           <xsd:element ref="ppst:App" minOccurs="0"/>
           <xsd:element ref="ppst:Condition" minOccurs="0" maxOccurs="unbounded"/>
           <xsd:element ref="ppst:Selection" maxOccurs="unbounded"/>
           <xsd:element ref="ppst:Header" minOccurs="0"/>
           <xsd:choice minOccurs="0">
            <xsd:element ref="pps:Operation" minOccurs="0" maxOccurs="unbounded"/>
           </xsd:choice>
          </xsd:sequence>
          <xsd:attribute name="id" type="xsd:string" use="required"/>
          <xsd:attribute name="action" type="xsd:string" fixed="Get"/>
          <xsd:attribute name="transaction" type="xsd:string"/>
          <xsd:attribute name="profile" type="xsd:string"/>
          <xsd:attribute name="create" type="xsd:dateTime"/>
          <xsd:attribute name="sender" type="xsd:string"/>
          <xsd:attribute name="description" type="xsd:string"/>
         </xsd:restriction>
        </xsd:complexContent>
       </xsd:complexType>
      </xsd:element>
      <xsd:element name="ResourceRecord">
       <xsd:complexType>
        <xsd:complexContent>
         <xsd:restriction base="ppst:TransactionType">
          <xsd:sequence>
           <xsd:element ref="ppst:App" minOccurs="0"/>
           <xsd:element ref="ppst:Condition" minOccurs="0" maxOccurs="unbounded"/>
           <xsd:choice minOccurs="0">
            <xsd:element ref="pps:Resource" maxOccurs="unbounded"/>
           </xsd:choice>
          </xsd:sequence>
          <xsd:attribute name="id" type="xsd:string" use="required"/>
          <xsd:attribute name="action" type="xsd:string"/>
          <xsd:attribute name="transaction" type="xsd:string" fixed="Get"/>
          <xsd:attribute name="profile" type="xsd:string"/>
          <xsd:attribute name="confirm" type="xsd:string"/>
          <xsd:attribute name="create" type="xsd:dateTime"/>
          <xsd:attribute name="sender" type="xsd:string"/>
          <xsd:attribute name="description" type="xsd:string"/>
         </xsd:restriction>
        </xsd:complexContent>
       </xsd:complexType>
      </xsd:element>
      <xsd:element name="InventryRecord">
       <xsd:complexType>
```

```
        <xsd:complexContent>
         <xsd:restriction base="ppst:TransactionType">
          <xsd:sequence>
           <xsd:element ref="ppst:App" minOccurs="0"/>
           <xsd:element ref="ppst:Condition" minOccurs="0" maxOccurs="unbounded"/>
           <xsd:choice minOccurs="0">
            <xsd:element ref="pps:Item" maxOccurs="unbounded"/>
           </xsd:choice>
          </xsd:sequence>
          <xsd:attribute name="id" type="xsd:string" use="required"/>
          <xsd:attribute name="action" type="xsd:string"/>
          <xsd:attribute name="transaction" type="xsd:string" fixed="Get"/>
          <xsd:attribute name="profile" type="xsd:string"/>
          <xsd:attribute name="confirm" type="xsd:string"/>
          <xsd:attribute name="create" type="xsd:dateTime"/>
          <xsd:attribute name="sender" type="xsd:string"/>
          <xsd:attribute name="description" type="xsd:string"/>
         </xsd:restriction>
        </xsd:complexContent>
       </xsd:complexType>
      </xsd:element>
     </xsd:choice>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>
 </xsd:element>

 <!—Definition Response Message GetInformation -->
 <xsd:element name="GetInformationResponse">
  <xsd:complexType>
   <xsd:complexContent>
    <xsd:extension base="micx:MessageType">
     <xsd:choice>
      <xsd:element name="ExecutionOrder">
       <xsd:complexType>
        <xsd:complexContent>
         <xsd:restriction base="ppst:TransactionType">
          <xsd:sequence>
           <xsd:element ref="ppst:Error" minOccurs="0" maxOccurs="unbounded"/>
           <xsd:element ref="ppst:App" minOccurs="0"/>
           <xsd:element ref="ppst:Header" minOccurs="0"/>
           <xsd:choice minOccurs="0">
            <xsd:element ref="pps:Operation" minOccurs="0" maxOccurs="unbounded"/>
           </xsd:choice>
          </xsd:sequence>
          <xsd:attribute name="id" type="xsd:string" use="required"/>
          <xsd:attribute name="action" type="xsd:string" fixed="Show"/>
          <xsd:attribute name="transaction" type="xsd:string"/>
          <xsd:attribute name="profile" type="xsd:string"/>
          <xsd:attribute name="create" type="xsd:dateTime"/>
          <xsd:attribute name="sender" type="xsd:string"/>
          <xsd:attribute name="description" type="xsd:string"/>
         </xsd:restriction>
        </xsd:complexContent>
       </xsd:complexType>
      </xsd:element>
      <xsd:element name="ResourceRecord">
       <xsd:complexType>
        <xsd:complexContent>
         <xsd:restriction base="ppst:TransactionType">
          <xsd:sequence>
           <xsd:element ref="ppst:Error" minOccurs="0" maxOccurs="unbounded"/>
           <xsd:element ref="ppst:App" minOccurs="0"/>
           <xsd:element ref="ppst:Header" minOccurs="0"/>
           <xsd:choice minOccurs="0">
            <xsd:element ref="pps:Resource" maxOccurs="unbounded"/>
           </xsd:choice>
          </xsd:sequence>
          <xsd:attribute name="id" type="xsd:string" use="required"/>
          <xsd:attribute name="action" type="xsd:string"/>
          <xsd:attribute name="transaction" type="xsd:string" fixed="Show"/>
          <xsd:attribute name="profile" type="xsd:string"/>
          <xsd:attribute name="create" type="xsd:dateTime"/>
          <xsd:attribute name="sender" type="xsd:string"/>
          <xsd:attribute name="description" type="xsd:string"/>
         </xsd:restriction>
        </xsd:complexContent>
       </xsd:complexType>
      </xsd:element>
      <xsd:element name="InventryRecord">
       <xsd:complexType>
        <xsd:complexContent>
         <xsd:restriction base="ppst:TransactionType">
          <xsd:sequence>
           <xsd:element ref="ppst:Error" minOccurs="0" maxOccurs="unbounded"/>
           <xsd:element ref="ppst:App" minOccurs="0"/>
           <xsd:element ref="ppst:Header" minOccurs="0"/>
           <xsd:choice minOccurs="0">
            <xsd:element ref="pps:Item" maxOccurs="unbounded"/>
           </xsd:choice>
          </xsd:sequence>
          <xsd:attribute name="id" type="xsd:string" use="required"/>
          <xsd:attribute name="action" type="xsd:string"/>
          <xsd:attribute name="transaction" type="xsd:string" fixed="Show"/>
          <xsd:attribute name="profile" type="xsd:string"/>
          <xsd:attribute name="create" type="xsd:dateTime"/>
          <xsd:attribute name="sender" type="xsd:string"/>
          <xsd:attribute name="description" type="xsd:string"/>
         </xsd:restriction>
        </xsd:complexContent>
       </xsd:complexType>
      </xsd:element>
     </xsd:choice>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>
 </xsd:element>

 <!—Definition Notify Message of NotifyEvent -->
 <xsd:element name="NotifyEvent">
  <xsd:complexType>
   <xsd:complexContent>
    <xsd:extension base="micx:MessageType">
     <xsd:choice>
      <xsd:element name="ExecutionOrder">
       <xsd:complexType>
        <xsd:complexContent>
         <xsd:restriction base="ppst:TransactionType">
          <xsd:sequence>
           <xsd:element ref="ppst:App" minOccurs="0"/>
           <xsd:element ref="ppst:Header" minOccurs="0"/>
           <xsd:choice minOccurs="0">
            <xsd:element ref="pps:Operation" maxOccurs="unbounded"/>
           </xsd:choice>
          </xsd:sequence>
          <xsd:attribute name="id" type="xsd:string" use="required"/>
          <xsd:attribute name="action" type="xsd:string" fixed="Notify"/>
          <xsd:attribute name="transaction" type="xsd:string"/>
          <xsd:attribute name="profile" type="xsd:string"/>
          <xsd:attribute name="create" type="xsd:dateTime"/>
          <xsd:attribute name="sender" type="xsd:string"/>
          <xsd:attribute name="description" type="xsd:string"/>
         </xsd:restriction>
        </xsd:complexContent>
       </xsd:complexType>
      </xsd:element>
      <xsd:element name="ResourceRecord">
       <xsd:complexType>
        <xsd:complexContent>
         <xsd:restriction base="ppst:TransactionType">
          <xsd:sequence>
           <xsd:element ref="ppst:App" minOccurs="0"/>
           <xsd:element ref="ppst:Header" minOccurs="0"/>
           <xsd:choice minOccurs="0">
            <xsd:element ref="pps:Resource" maxOccurs="unbounded"/>
           </xsd:choice>
          </xsd:sequence>
          <xsd:attribute name="id" type="xsd:string" use="required"/>
          <xsd:attribute name="action" type="xsd:string"/>
          <xsd:attribute name="transaction" type="xsd:string" fixed="Notify"/>
          <xsd:attribute name="profile" type="xsd:string"/>
          <xsd:attribute name="create" type="xsd:dateTime"/>
          <xsd:attribute name="sender" type="xsd:string"/>
          <xsd:attribute name="description" type="xsd:string"/>
         </xsd:restriction>
        </xsd:complexContent>
       </xsd:complexType>
      </xsd:element>
     </xsd:choice>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

B.6 Example of CCD

B.6.1 General

CCDb.xsd is an XML schema example of CCD that extends the XML schema of MICX-specific DCPT in Clause B.4 in accordance with the coordinator capabilities of the coordinator that imports the DCDs of the device drivers.

B.6.2 XML schema : CCDb.xsd

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns="http://www.mstc.or.jp/micx/ISO20242-4/b/Example"
            xmlns:micx="http://www.mstc.or.jp/micx/ISO20242-4/MICX"
            targetNamespace="http://www.mstc.or.jp/micx/ISO20242-4/b/Example"
            elementFormDefault="qualified">

   <xsd:annotation>
      <xsd:appinfo source="http://www.osi.ch/iso/ISO20242-4/DCPTHeader.xsd">
        <DCPTHeader>
          <DCPTIdentification>CCD</DCPTIdentification>
          <DCPTRevision>1.0</DCPTRevision>
          <DCPTName>CCD Sample</DCPTName>
          <DCPTSource>CCDb.xsd</DCPTSource>
          <DCPTClassID>InformationExchangeTemplate</DCPTClassID>
          <DCPTDate>2009-02-25</DCPTDate>
          <DCPTVender>FAOP</DCPTVender>
          <TechnologyReference>
            <Technology>FAOP-MICX</Technology>
            <Revision>1.0</Revision>
          </TechnologyReference>
        </DCPTHeader>
      </xsd:appinfo>
    </xsd:annotation>
```

```
<!--  *  Import MICXCommon.xsd,*  -->
<xsd:import namespace="http://www.mstc.or.jp/micx/ISO20242-4/MICX"
          schemaLocation="MICXCommon.xsd"/>

<!-- A coodinator capability description contains the capability descriptions of
     all devices, which will be usable -->
<xsd:include schemaLocation="DCDb.xsd"/>

<!--  * ISO15745 Profile Root *  -->
<xsd:element name="ISO15745Profile">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="ProfileHeader"/>
      <xsd:element ref="ProfileBody"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

  <!-- * HEADER DATA TYPES *  -->
<xsd:element name="ProfileHeader">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="ProfileIdentification"  type="xsd:string"/>
      <xsd:element name="ProfileRevision"        type="xsd:string"/>
      <xsd:element name="ProfileName"            type="xsd:string"/>
      <xsd:element name="ProfileSource"          type="xsd:string"/>
      <xsd:element name="ProfileClassID"         type="ProfileClassID_DataType"
                                                 fixed="InformationExchange"/>
      <xsd:element name="ProfileDate"            type="xsd:date"
                   minOccurs="0"/>
      <xsd:element name="AdditionalInformation"  type="xsd:anyURI"
                   minOccurs="0"/>
      <xsd:element name="ISO15745Reference"      type="ISO15745Reference_DataType"/>
      <xsd:element name="IASInterfaceType"       type="IASInterface_DataType"
                                                 fixed="CSI"
                   minOccurs="0"                 maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:complexType name="ISO15745Reference_DataType">
  <xsd:sequence>
    <xsd:element name="ISO15745Part"       type="xsd:string" fixed="1"/>
    <xsd:element name="ISO15745Edition"    type="xsd:string" fixed="1"/>
    <xsd:element name="ProfileTechnology"  type="xsd:string" fixed="None"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="ProfileClassID_DataType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="AIP"/>
    <xsd:enumeration value="Process"/>
    <xsd:enumeration value="InformationExchange"/>
    <xsd:enumeration value="Resource"/>
    <xsd:enumeration value="Device"/>
    <xsd:enumeration value="CommunicationNetwork"/>
    <xsd:enumeration value="Equipment"/>
    <xsd:enumeration value="Human"/>
    <xsd:enumeration value="Material"/>
  </xsd:restriction>
</xsd:simpleType>

 <xsd:simpleType name="IASInterface_DataType">
  <xsd:union>
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="CSI"/>
        <xsd:enumeration value="HCI"/>
```

```
            <xsd:enumeration value="ISI"/>
            <xsd:enumeration value="API"/>
            <xsd:enumeration value="CMI"/>
            <xsd:enumeration value="ESI"/>
            <xsd:enumeration value="FSI"/>
            <xsd:enumeration value="MTI"/>
            <xsd:enumeration value="SEI"/>
            <xsd:enumeration value="USI"/>
          </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:length value="4"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:union>
    </xsd:simpleType>

    <!-- * BODY SECTION *  -->
    <xsd:element name="ProfileBody">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="CCD" maxOccurs="unbounded">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="DCD1" type="DCDType"
                             minOccurs="0" maxOccurs="unbounded"/>
              </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string"/>
              <xsd:attribute name="category" type="micx:categoryType"
                             use="required" fixed="CCD"/>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

B.7 Example of PID

B.7.1 General

SamplePIDb.xml is an XML data example of the PID which use CCDb.xsd and DCDb.xsd as XML schema. It describes parameter instances of the recipe loading to a NC lathe.

B.7.2 XML : SamplePIDb.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ISO15745Profile xmlns="http://www.mstc.or.jp/micx/ISO20242-4/b/Example"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.mstc.or.jp/micx/ISO20242-4/b/Example CCDb.xsd">

 <ProfileHeader>
    <ProfileIdentification>ExamplePIDb</ProfileIdentification>
    <ProfileRevision>1.0</ProfileRevision>
    <ProfileName>Example of PID for MICX</ProfileName>
    <ProfileSource>SamplePIDb.xml</ProfileSource>
    <ProfileClassID>InformationExchange</ProfileClassID>
    <ProfileDate>2009-02-25</ProfileDate>
    <AdditionalInformation>http://www.mstc.or.jp/micx/ISO20242-4/PID
    </AdditionalInformation>
    <ISO15745Reference>
       <ISO15745Part>1</ISO15745Part>
       <ISO15745Edition>1</ISO15745Edition>
       <ProfileTechnology>None</ProfileTechnology>
       </ISO15745Reference>
    <IASInterfaceType>CSI</IASInterfaceType>
 </ProfileHeader>

 <ProfileBody>
  <CCD name="ExamplePID" category="CCD">
   <DCD1 name="MICX" category="DCD">
    <Device name="NC_lathe" category="MODULE">
     <ProductionControl name="maching" category="INTERFACE">
      <LoadRecipe category="OPERATION">
       <LoadRecipeRequest message="REQUEST">
        <RecipeRecord id="001" action="Add" confirm="Always">
         <Recipe id="f001" name="machining">
          <AcceptedWork item="B"/>
          <AssignedEquipment resource="NC_C"/>
          <Length name="insideDiameter">
           <Qty value="15" unit="mm"/>
          </Length>
          <Length name="outsideDiameter">
           <Qty value="36" unit="mm"/>
          </Length>
          <Length name="thickness">
           <Qty value="2.6" unit="mm"/>
          </Length>
         </Recipe>
        </RecipeRecord>
       </LoadRecipeRequest>
       <LoadRecipeRequest message="REQUEST">
        <RecipeRecord id="002" action="Add" sender="MESX" confirm="Always">
         <Recipe id="f002" name="machining">
          <AcceptedWork item="B"/>
          <AssignedEquipment resource="NC_C"/>
          <Length name="insideDiameter">
           <Qty value="16" unit="mm"/>
          </Length>
          <Length name="outsideDiameter">
           <Qty value="40" unit="mm"/>
          </Length>
          <Length name="thickness">
           <Qty value="2.6" unit="mm"/>
          </Length>
         </Recipe>
        </RecipeRecord>
       </LoadRecipeRequest>
      </LoadRecipe>
     </ProductionControl>
    </Device>
   </DCD1>
  </CCD>
 </ProfileBody>
</ISO15745Profile>
```

Annex C
(informative)

Device capability profile templates for ORiN robot applications

C.1 General

The ORiN (Open Robot interface for the Network) specifies a general device interface for applications. This annex defines the ORiN-specific device capability profile template of ORiN Version 2.1[6], and provides examples of DCD, CCD and PID.

C.2 ORiN-specific profile model

C.2.1 General

The ORiN-specific profile model contains all information that is necessary for the device capability description and all information that is necessary for parameterization. Figure C.1 shows the class diagram of ORiN-specific device capability profile template.

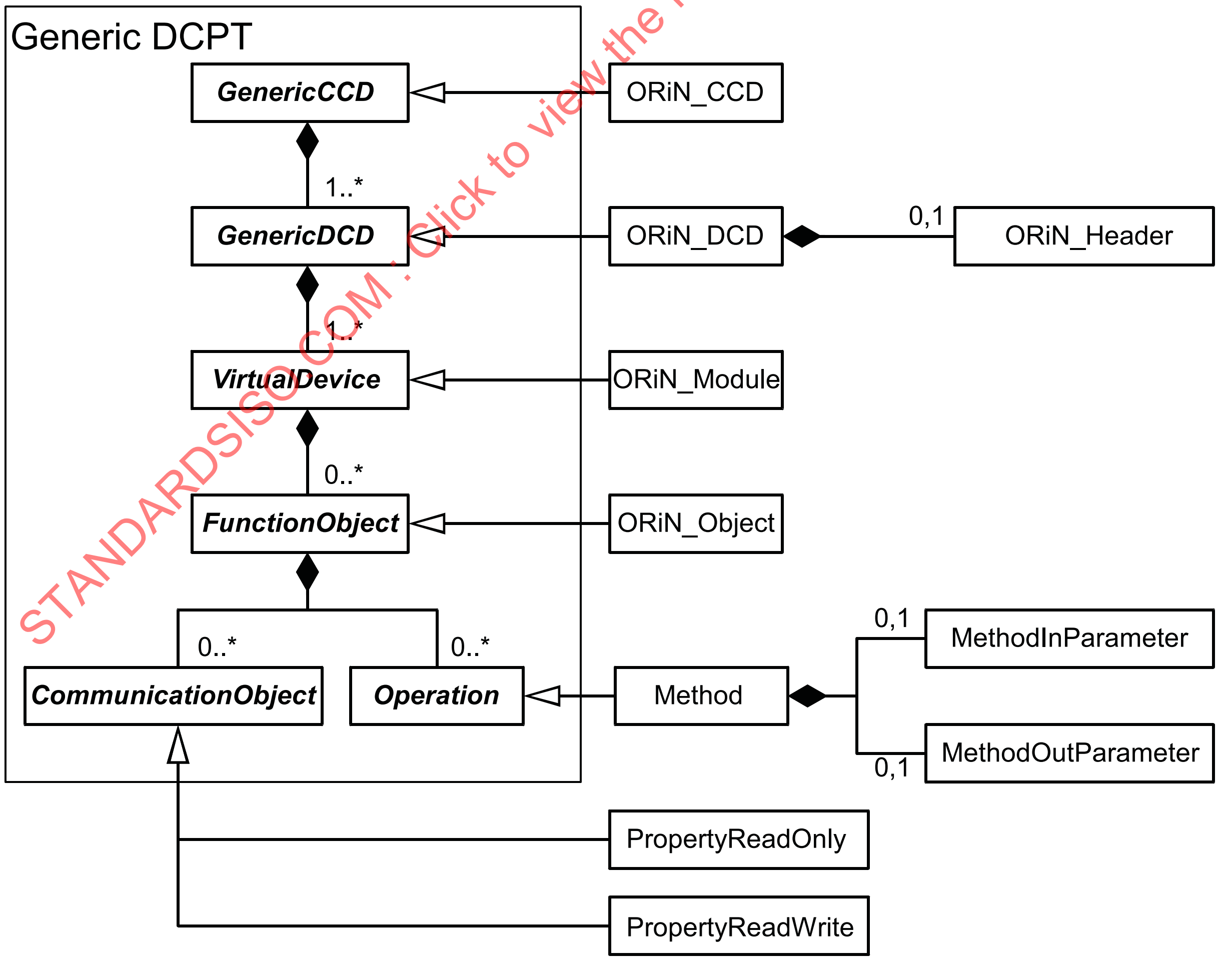


Figure C.1 — Class diagram of ORiN-specific DCPT model

C.2.2 ORiN_CCD

The ORiN_CCD class shows the capabilities of the ORiN-specific coordinator. It inherits the GenericCCD class and is an abstract class. A coordinator-specific ORiN_CCD class inherits it and defines capabilities of the specific coordinator.

C.2.3 ORiN_DCD

The ORiN_DCD class shows the capabilities of the ORiN-specific device driver. It inherits the GenericDCD class and is an abstract class. A device-specific ORiN_DCD class inherits it and defines capabilities of the specific device driver.

C.2.4 ORiN_Header

The ORiN_Header class contains additional information, which is used for the instantiation of the device driver. The elements of ORiN_Header are described in Table C.1. A definition of ORiN_Header is also given in ORiNcommon.xsd in A.6.2.

Table C.1 — Elements of ORiN_Header

Element of ORiN_Header		Element type	Description
DCD_Version		xsd:unsignedInt	A number for the DCD version
DeviceVersion		xsd:unsignedInt	A number for the device version
ProviderName		xsd:string	The name of ORiN provider
ProviderVersion		xsd:unsignedInt	A number for the ORiN provider version
Factory		xsd:string	The name of the manufacturer
DIT		xsd:string	The name of the XML text file
ORiN_Version	Major	xsd:unsignedByte	Major version number
	Minor	xsd:unsignedByte	Minor version number
	Revision	xsd:unsignedByte	Revision number

C.2.5 ORiN_Module

The ORiN_Module class shows capabilities of the ORiN-specific virtual device. It inherits the VirtualDevice class and is an abstract class. A device-specific ORiN_Module class inherits it and defines the capabilities of the specific virtual device. An ORiN_Module class may have a CreateParameter and is identified by a number contained in an additional XML attribute named "moduleId" of type "xsd:unsignedShort".

C.2.6 ORiN_Object

The ORiN_Object class shows capabilities of function objects of the ORiN-specific virtual device. It inherits the FunctionObject class and is an abstract class. A device-specific ORiN_Object class inherits it and defines the capabilities of the device-specific function object. An ORiN_Object class may have a CreateParameter and is identified by a number contained in an additional XML attribute named "funcId" of type "xsd:unsignedShort".

C.2.7 Method

The Method class describes an operation of the ORiN-specific virtual device. It inherits the Operation class and is an abstract class. Each device-specific Method class inherits it and defines the capabilities of the operation. Each Method has one operation input parameter and one operation output parameter that may occur zero or one time in the XML instance. The assignment of a value to the operation input parameter in the XML instance (PID) marks that the operation shall be executed for configuration. A Method is identified by a number contained in an additional XML attribute named "operationIdId" of type "xsd:unsignedShort".

C.2.8 MethodInParameter

The MethodInParameter marks the execution of an operation. It is an abstract class. Each device-specific MethodInParameter class inherits this class and defines the data type of the input parameter.

C.2.9 MethodOutParameter

The MethodOutParameter describes the return value of an operation. It is an abstract class and contains the value of the output parameter. Each device-specific MethodOutParameter class inherits this class and defines the data type of the value.

C.2.10 PropertyReadOnly

The PropertyReadOnly class describes a runtime value of the function object that can only be read. It inherits the CommunicationObject class and is an abstract class. Each device-specific PropertyReadOnly class inherits this class and defines the capabilities of each runtime value and the data type of the value.

C.2.11 PropertyReadWrite

The PropertyReadWrite class describes a runtime value of the function object that can be read and written. It inherits the CommunicationObject class, and is an abstract class. Each device-specific PropertyReadWrite class inherits this class and defines the capabilities of each runtime value and its data type.

C.2.12 Identification of ORiN communication objects

The instances of classes PropertyReadOnly and PropertyReadWrite are identified by a number, which is the counting of these instances inside a FunctionObject instance, starting at 1.

C.3 Elements for typed Data and Configuration Scenarios

C.3.1 General

An element Value is defined to facilitate the inheritation of data types without influencing the inheritation of structural classes. With this method, the definition of data types for communication objects and create parameters is decoupled from the definition of the communication object itself. Value is not necessary, if data types are defined implicitly with the communication object.

C.3.2 Value

The Value class describes an element of type xsd:anyType, which may be restricted to any other type and therefore it is a placeholder for any element type defined in the device capability description elsewhere. The use of this extra element facilitates a decoupled inheritation for the definition of data types (see Figure C.2).

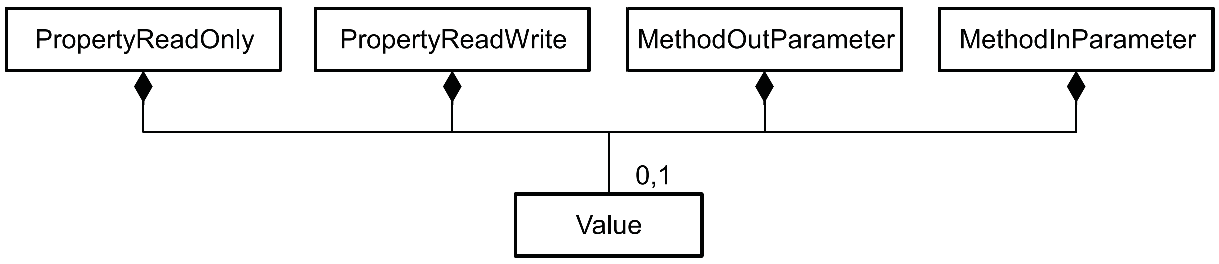


Figure C.2 — Element Value for decoupled type definitions

C.4 Additional type identification to generic DCPT

Tables C.2 to C.4 contain technology-specific type identifications for communication objects and operation parameters.

Table C.2 — Type identification for communication objects

Element of technology-specific DCPT	Contents of XML attribute "category"	Contents of XML attribute "readonly"[a]
AttributeReadWrite	ATTRIBUTE	false
AttributeReadOnly	ATTRIBUTE	true
Parameter	PARAMETER	(false)

[a] The extra XML attribute "readonly" of type xsd:boolean is mandatory for AttributeReadWrite and AttributeReadOnly.

Table C.3 — Type identification for operation parameters

Element of technology-specific DCPT	Contents of XML attribute "category"
OperationInParameter	IN
OperationOutParameter	OUT

Table C.4 — Type identification for create parameters

Element of technology-specific DCPT	Contents of XML attribute "category"
CreateParameter	CREATEPARAMETER

C.5 Additional XML attributes to generic DCPT

The communication objects AttributeReadOnly and AttributeReadWrite may be used for unsolicited data transmission with services VDSI_InfReport and VDSI_Accept. A request for using these services at runtime is done at configuration time by setting the XML attributes infReport and accept of type xsd:boolean to the value true (see Table C.5).

Table C.5 — XML attributes for unsolicited data transfer

XML attribute name	Used with object	Description of content
infReport	AttributeReadWrite, AttributeReadOnly	Request for using service VDSI_InfReport
accept	AttributeReadWrite	Request for using service VDSI_Accept

C.6 ORiN-specific device capability profile template

C.6.1 General

ORiNcommon.xsd is an XML schema that contains base classes. These base classes can be used to describe an XML schema that contains all information for the device capability description and all information that is necessary for parameterization. The defined types AttributeReadWrite may be extended with elements

Value. In this case Value may have a data type, which may be defined in an extra XML schema. Also, the defined types AttributeReadOnly and OperationOutParameter may be extended by Value elements of external defined data types.

NOTE If elements Value are not used and the communication objects contain values of simple or complex data type, there might be another construction of the XML schema, based on the extension of value data types. In this case, ORiNcommon.xsd can be used as a sample for the structure of a resulting XML instance and the necessary XML attributes.

C.6.2 XML schema : ORiNcommon.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    ;=========================================================================
    ; ORiNcommon.xsd                                       Edition: 27-Feb-2010
    ;=========================================================================
-->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            elementFormDefault="qualified"
            attributeFormDefault="unqualified">

  <!-- Header information is contained in each schema file -->
  <xsd:annotation>
    <xsd:appinfo source="http://www.ORiN.jp/ISO20242-4/DCPTHeader.xsd">
      <DCPTHeader>
        <DCPTIdentification>ORiN_Common</DCPTIdentification>
        <DCPTRevision>1.0</DCPTRevision>
        <DCPTName>ORiNcommon</DCPTName>
        <DCPTSource>ORiNcommon.xsd</DCPTSource>
        <DCPTClassID>TechnlogySpecificDCPT</DCPTClassID>
        <DCPTDate>2010-02-27</DCPTDate>
        <ISO20242Reference>
          <ISO20242Edition>1</ISO20242Edition>
          <Technology>ORiN</Technology>
        </ISO20242Reference>
      </DCPTHeader>
    </xsd:appinfo>
  </xsd:annotation>

  <!-- Valid content for XML-Attribut category (kind of DCD element) -->
  <xsd:simpleType name="Category">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="MODULE"/>
      <xsd:enumeration value="INTERFACE"/>
      <xsd:enumeration value="ATTRIBUTE"/>
      <xsd:enumeration value="OPERATION"/>
      <xsd:enumeration value="IN"/>
      <xsd:enumeration value="OUT"/>
      <xsd:enumeration value="DCD"/>
      <xsd:enumeration value="CCD"/>
    </xsd:restriction>
  </xsd:simpleType>

  <!-- Group of attributes for assigning multilingual text to elements -->
  <xsd:attributeGroup name="TextAttributes">
    <xsd:attribute name="areaMsg"    type="xsd:unsignedShort"
                   use="optional"/>
    <xsd:attribute name="infMsg"     type="xsd:unsignedShort"
                   use="optional"/>
    <xsd:attribute name="areaText"   type="xsd:unsignedShort"
                   use="optional"/>
    <xsd:attribute name="infText"    type="xsd:unsignedShort"
                   use="optional"/>
  </xsd:attributeGroup>
```