

Health informatics — Interoperability of patient connected medical devices

ICS 35.240.80

National foreword

This Draft for Development is the English language version of ENV 13735:2001.

This publication is not to be regarded as a British Standard.

It is being issued in the Draft for Development series of publications and is of a provisional nature due to the limited duration of the European Prestandard. It should be applied on this provisional basis, so that information and experience of its practical application may be obtained.

Comments arising from the use if this Draft for Development are requested so that UK experience can be reported to the European organization responsible for its conversion into a European Standard. A review of this publication will be initiated 2 years after its publication by the European organization so that a decision can be taken on its status at the end of its three-year life. The commencement of the review period will be notified by an announcement in *Update Standards*.

According to the replies received by the end of the review period, the responsible BSI Committee will decide whether to support the conversion into a European Standard, to extend the life of the prestandard or to withdraw it. Comments should be sent in writing to the Secretary of BSI Technical Committee IST/35, Health Informatics, at 389 Chiswick High Road, London W4 4AL, giving the document reference and clause number and proposing, where possible, an appropriate revision of the text.

A list of organizations represented on this committee can be obtained on request to its secretary.

Cross-references

The British Standards which implement international or European publications referred to in this document may be found in the BSI Standards Catalogue under the section entitled “International Standards Correspondence Index”, or by using the “Find” facility of the BSI Standards Electronic Catalogue.

This Draft for Development, having been prepared under the direction of the DISC board, was published under the authority of the Standards Policy and Strategy Committee on 19 October 2001

Summary of pages

This document comprises a front cover, an inside front cover, the ENV title page, pages 2 to 175 and a back cover.

The BSI copyright date displayed in this document indicates when the document was last issued.

Amendments issued since publication

Amd. No.	Date	Comments

© BSI 19 October 2001

ISBN 0 580 38594 9

ICS 35.240.80

English version

Health informatics - Interoperability of patient connected medical devices

This European Prestandard (ENV) was approved by CEN on 7 January 2000 as a prospective standard for provisional application.

The period of validity of this ENV is limited initially to three years. After two years the members of CEN will be requested to submit their comments, particularly on the question whether the ENV can be converted into a European Standard.

CEN members are required to announce the existence of this ENV in the same way as for an EN and to make the ENV available promptly at national level in an appropriate form. It is permissible to keep conflicting national standards in force (in parallel to the ENV) until the final decision about the possible conversion of the ENV into an EN is reached.

CEN members are the national standards bodies of Austria, Belgium, Czech Republic, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland and United Kingdom.



EUROPEAN COMMITTEE FOR STANDARDIZATION
COMITÉ EUROPÉEN DE NORMALISATION
EUROPÄISCHES KOMITEE FÜR NORMUNG

Central Secretariat: rue de Stassart, 36 B-1050 Brussels

Contents

Foreword..... 4

Introduction 5

1 **Scope**..... 6

2 **Normative reference(s)** 6

3 **Term(s) and definition(s)** 6

4 **Symbols (and abbreviated terms)** 11

5 **Requirements** 13

6 **Models** 14

6.1 **Domain Information Model for the Communication Subject** 14

6.2 **Object Definitions**..... 16

6.3 **Dynamic Model** 24

7 **Messages** 30

7.1 **General**..... 30

7.2 **ACSE Protocol**..... 31

7.3 **Session Layer Protocol** 31

7.4 **Presentation Layer Protocol** 31

7.5 **ROSE Protocol**..... 32

7.6 **CMDISE Protocol (CMDIP)**..... 32

8 **Encoding rules** 33

8.1 **Overview** 33

8.2 **Medical Data Encoding Rules (MDER)** 34

8.3 **Abstract Syntax** 34

8.4 **Transfer Syntax** 34

9 **Application Profiles**..... 35

9.1 **General** 35

9.2 **Baseline Application Profile**..... 36

9.3 **Polling Mode Application Profile** 53

9.4 **Conformance** 61

9.5	Implementation Conformance Statements	61
Annex A (informative)	Example Optional Packages	67
Annex B (normative)	Nomenclature Extensions	70
Annex C (informative)	ACSE User Info Field	78
Annex D (normative)	IEEE 1073 Specific MIB Element Object Definitions	80
Annex E (informative)	Requirements and Scenarios	84
Annex F (Informative)	Mapping between Scenarios, Device Types and Application Profiles ...	101
Annex G (normative)	State Transition Tables	171

Foreword

This European Prestandard has been prepared by Technical Committee CEN/TC 251 "Health informatics", the secretariat of which is held by SIS.

According to the CEN/CENELEC Internal Regulations, the national standards organizations of the following countries are bound to announce this European Prestandard: Austria, Belgium, Czech Republic, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland and the United Kingdom.

Introduction

Medical technology has developed in the context of rapid advances in the science of information technology and has benefited greatly by incorporating these advances into many medical devices. This has been done in a relatively unstructured manner with many devices being developed in isolation and in a way that precludes communication both between themselves and with hospital computer and data management systems. Those devices that can communicate generally use a proprietary communication protocol, which precludes connection to devices from different manufacturers.

The potential advantages of such communication are becoming more and more obvious. Consequently, there is a pressing need for technical standardisation and the development of services and protocols to enable such communication to exist in an easy and open way, with subsequent clinical, administrative and research benefits.

The first steps toward the development of such a standard were initiated by IEEE group p1073 in 1984 developing what has become know as the Medical Information Bus (MIB). The development of that set of standards has followed the ISO/OSI layers as demonstrated in the following figure:

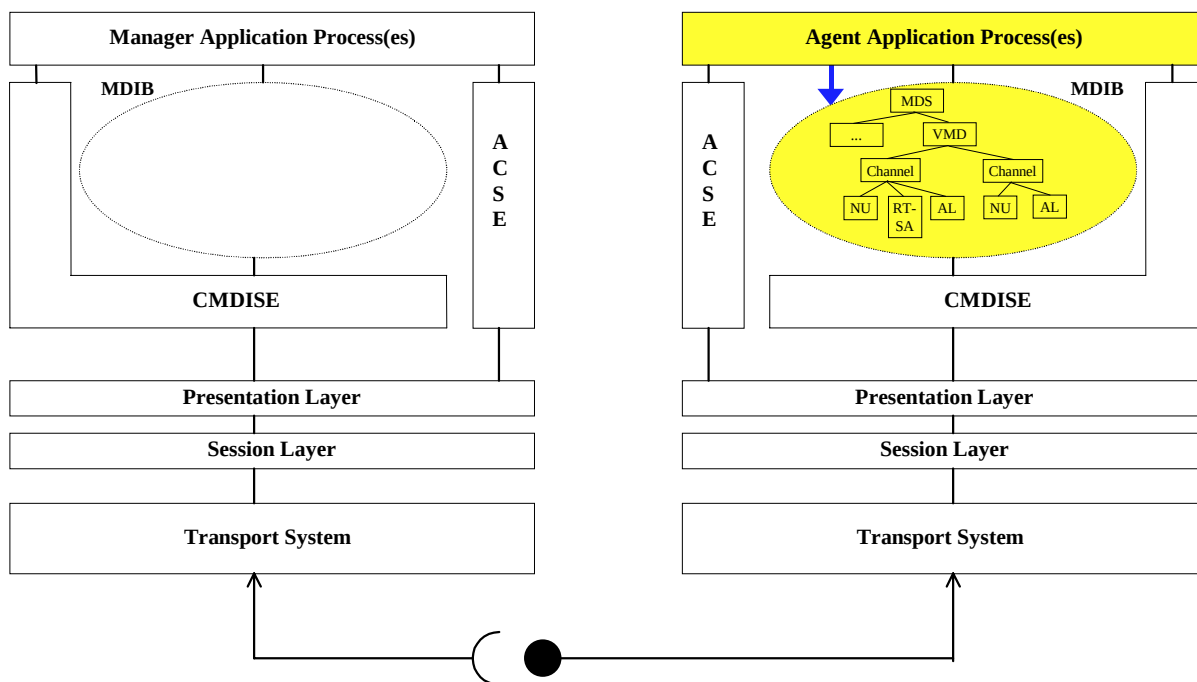


Figure 1 — . Example of agent manager communication system following ISO/OSI principals.

An essential element was the development of a standard for Vital Signs Information Representation, which covered the objects and attributes collectively known as the MDIB (area shaded in fig 1). This work has now been done (ENV 13734). The next stage, which is the focus of this project, is the development of necessary services and protocols, based on ISO/OSI standards, which enable true interoperability between medical devices. These cover the areas in the figure marked as ACSE, CMDISE, presentation layer and session layer. Interoperability of medical devices raises many technical, safety, and legal issues, which are worthy of consideration. The ongoing work of the IEEE standards group IEEE 1073 has been taken into consideration and reference is made particularly to their proposed standard 1073.2.0 Medical Device Application Profile (MDAP).

The primary user of this standard is a software engineer who is developing interface software for medical devices covered in the scope. This is one of a number of standards that the engineer should be familiar with, including the following:

ISO open systems interconnection (OSI) layered architecture.

IEEE 1073.1 MIB framework and overview.

CEN ENV 13734 "Vital Signs Information Representation"

1 Scope

This European standard applies to patient connected medical devices that can communicate patient related physiological data between devices and between a device and a computer system. Examples of applicable devices are given in annex E.

This standard specifies a communication controller model that should be used by applicable devices.

This standard states applicable references or specifies protocols including the ACSE, ROSE, CMDISE, presentation and session layer protocols that shall be used by applicable devices.

This standard defines application profiles that shall be used by applicable devices, and that shall be quoted by manufacturers when claiming conformance to this standard. Appropriate conformance statement templates are provided.

2 Normative reference(s)

The following normative documents contain provisions that, through reference in this text, constitute provisions of this CEN Standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this CEN Standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For an undated reference, the latest edition of the normative document referred to will apply.

IEEE 1073.2.0: Standard for Medical Device Communications – Medical Device Application Profile

IEEE 1073.3.1: 1996 Standard for Medical Device Communications - Transport Profile - Connection Mode

IEEE 1073.4.1: 1996 Standard for Medical Device Communications - Physical Layer Interface - Cable-connected

ENV 13734: 1999 Vital Signs Information Representation (VITAL)

ISO 7498-1: 1994 Information Technology - Open Systems Interconnection - Basic Reference Model

ISO 8649: Information technology - Open Systems Interconnection - Service definition for the Association Control Service Element

ISO 8824: 1990 Information technology - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1).

ISO 8825 Information technology - Open Systems Interconnection - Specification of encoding rules for Abstract Syntax Notation One (ASN.1).

ISO 9596 Information technology - Open Systems Interconnection – Common Management Information Protocol Specification

3 Term(s) and definition(s)

3.1

Agent:

A Medical Device which provides data in a manager/agent communicating system.

3.2

Alarm:

A Signal which indicates abnormal events occurring to the patient or the device system.

3.3

Alert:

Synonym for the combination of patient-related physiological alarms, technical alarms and equipment user advisory signals.

3.4

Alert Monitor:

Object representing the output of a device or system alarm processor and as such the overall device or system alarm condition.

3.5

Alert Status:

Object representing the output of an alarm process that considers all alarm conditions in a scope that spans one or more objects.

3.6

Archival:

Relating to the storage of data over a prolonged period.

3.7

Association Control Service Element

Method used to establish logical connections between medical device systems.

3.8

Channel:

Umbrella object in the object model that groups together physiological measurement data and data derived from these data.

3.9

Class:

Description of one or more objects with a uniform set of attributes and services including a description of how to create new objects in the class.

3.10

Common Medical Device Information Service Element

With associated protocol provides services that allow access to the object instances in the Medical Data Information Base.

3.11

Communication Controller:

Part of a Medical Device System responsible for communications.

3.12

Communication Party:

Actor of the problem domain participating in the communication therein.

3.13

Communication Role:

Role of a party in a communication situation defining the party's behaviour in the communication. Associated with a communication role is a set of services which the party provides to other parties.

3.14

Data Format:

Arrangement of data in a file or stream.

3.15

Data Logger:

A medical device which is functioning in its capacity as a data storage and archival system.

3.16

Data Agent:

As a medical device a patient data acquisition system which provides the acquired data for other devices.

3.17

Data Structure:

Manner in which application entities construct the data set information resulting from the use of an information object.

3.18

Dictionary:

Description of the contents of the Medical Data Information Base containing vital signs information, device information, demographics, and other elements of the MDIB.

3.19

Discrete Parameter:

Vital signs measurement that can be expressed as a single numeric or textual value.

3.20

Domain Information Model:

The model describing common concepts and relationships for a problem domain.

3.21

Event:

A noteworthy occurrence that has a location in time and space.

3.22

Event Report:

Service (provided by the CMDISE) to report data relating to a managed object instance.

3.23

Framework:

A structure of processes and specifications designed to support the accomplishment of a specific task.

3.24

Graphic Parameter:

Vital signs measurement that requires a number of regularly sampled data points in order to be expressed properly.

3.25

Host system:

Term used as an abstraction of a medical system to which measurement devices are attached.

3.26

Information Objects:

Provide an abstract data model applicable to the communication of vital signs information and related patient data. The attributes of an information object definition describe its properties. Each information object definition does not represent a specific instance of real-world data, but rather a class of data which share the same properties.

3.27

Instance:

For example: object instance, application instance, information service element instance, VMD instance, class instance, operating instance

3.28

Intensive Care Unit:

Unit within a hospital in which critically ill patients are managed using multiple modes of monitoring and vital system supports.

3.29

Interchange Format:

The representation of the data elements and the structure of the message containing those data elements while in transfer between systems. The interchange format consists of a data set of construction elements and a syntax. The representation is technology specific.

3.30

Interoperability:

Idealised scheme whereby medical devices of differing types, models or manufacturers, are capable of working with each other, whether connected to each other directly or through a communication system. The communication system tracks location, connection, disconnection, replacement and re-siting of all devices.

3.31

Latency:

In a communications scenario latency is the time delay between sending a signal from one device and receiving it by another device.

3.32

Lower Layers:

Layers 1 to 4 of the ISO/OSI reference model. These layers cover mechanical, electrical and general communication protocol specifications.

3.33

Manager:

Device which receives data in a manager/agent communicating system.

3.34

Manager Agent model:

Communication model whereby one device (agent) provides data and another device (manager) receives data.

3.35

Medical Data Information Base:

The concept of an object oriented database storing (at least) vital signs information.

3.36

Medical Device System:

Abstraction for system comprising one or more medical devices.

3.37

Medical Devices:

Devices, apparatus, or systems used to prevent, measure, diagnose, or treat diseases in humans, and which do not normally enter metabolic pathways. For the purposes of this standard the scope of medical devices is further limited to those patient connected medical devices which provide support for electronic communications.

3.38

Monitor:

Patient data acquisition device designed to provide warning of possible patient deterioration.

3.39

Object Attributes:

Data, which, together with methods, define an object.

3.40

Object Class:

A descriptor used in association with a group of objects with similar properties (attributes), common behaviour (operations), common relationships to other objects, and common semantics.

3.41

Object Diagram:

Diagram showing connections between objects in a system.

3.42

Object Oriented Analysis:

Method of analysis described by Coad and Yourdon but the term Object-Oriented is used widely to describe similar techniques.

3.43

Object:

A concept, an abstraction or a thing with crisp boundaries and a meaning for the problem at hand.

3.44

Open System:

Set of protocols allowing computers of different origins to be linked together.

3.45

Operation:

A function or transformation that may be applied to or by objects in a class. (Sometimes also called service.)

3.46

Problem Domain:

The field of health care under consideration in a modelling process.

3.47

Protocol:

A standard set of rules describing the transfer of data between devices. It specifies the format of the data, and the signals to start, control and end the transfer.

3.48

Scanner:

Observer and "summariser" of object attribute values.

3.49

Scenario

Formal description of a class of business activities including the semantics of business agreements, conventions, and information content.

3.50

Service:

A specific behaviour that a communication party in a specific role is responsible for exhibiting.

3.51

Syntax:

The syntax of an Interchange Format (IF) describes the rules for combining the construction elements of the Interchange Format.

3.52

System:

Demarcated part of the perceivable universe, existing in time and space, that may be regarded as a set of elements and relationships between these elements.

3.53

Timestamp:

An attribute or field in data, which denotes the time of data generation.

3.54

Top Object:

Ultimate base class for all other objects belonging to one model.

3.55

Upper Layers:

Layers 5 to 7 of the ISO/OSI reference model. These layers cover application, presentation, and session specifications and functionalities.

3.56

Virtual Medical Device:

An abstract representation of a medical related subsystem of a Medical Device System.

3.57

Virtual Medical Object

An abstract representation of an object in the medical subject of the Domain Information Model.

3.58

Vital Signs:

Clinical information relating to one or more patients. It can be measured by or derived from apparatus connected to the patient, or it is otherwise gathered from the patient.

3.59

Waveform:

Here, used as graphic data, waveform data, varying data. Vital signs data, which is usually presented to the clinician in a graphical form.

4 Symbols (and abbreviated terms)

ACSE	Association Control Service Element
APDU	Application Protocol Data Unit
ASN.1	Abstract Syntax Notation One
BCC	Bedside Communication Controller
BER	Basic Encoding Rules
CC	Communication Controller
CEN	European Committee for Standardisation
CMDIP	Common Medical Device Information Protocol
CMDISE	Common Medical Device Information Service Element
DCC	Device Communication Controller
DIM	Domain Information Model
DIS	Department Information System
FSM	Finite State Machine
ID	Identification
IEEE	Institute of Electrical and Electronic Engineers
IF	Interchange Format
ISO	International Standards Organisation

LAN	Local Area Network
MDAP	Medical Device Application Profile (IEEE 1073.2.0 standard)
MDIB	Medical Data Information Base
MDS	Medical Device System
MIB	Management Information Base
MTU	Maximum Transmit Unit
OID	Object Identifier
OSI	Open Systems Interconnection
PDU	Protocol Data Unit
ROSE	Remote Operation Service Element
SNMP	Simple Network Management Protocol
SNTP	Simple Network Time Protocol
UML	Unified Modelling Language
VITAL	Vital Signs Information Representation standard ENV 13734
VMD	Virtual Medical Device
VMO	Virtual Medical Object
VSIR	Alternative abbreviation for VITAL defined above

5 Requirements

A Baseline Application profile and a Polling mode profile are defined in clause 9. In addition a set of optional packages may be required in order to provide for additional functionality. Examples of the optional packages are outlined in informative annex A. Full definitions of all the optional packages are not the purpose of this standard and may be expected to appear in future documents.

Any medical device within the scope of this standard and claiming conformance to this draft standard shall do the following :

1. Shall conform to either a polling mode profile or a baseline profile and including any optional packages. Further conformance information is given in clause 9.
2. Shall provide a conformance statement in the format given which states what optional packages and what profile is used by the medical device.
3. Shall provide MDIB codes conforming to ENV 13734 'Vital Signs Information Representation'.
4. Should make use of the object models contained in this standard and ENV 13734 'Vital Signs Information Representation'.
5. Shall make references to object identifiers using codes as defined in ENV 13734 'Vital Signs Information Representation'.
6. Should use standards for layers 1-4 of the OSI Reference Model which are recognised as applicable to patient connected medical device communications.

6 Models

The Domain Information Model (DIM) for the Communication Controller extends the object-oriented model defined in the standard for Vital Signs Information Representation (VITAL ENV 13734). VITAL structures the problem domain using Coad- Yourdon- style subjects.

The model in this standard fully defines the VITAL Communication Subject. This subject is a static object model that represents information about the communication sub- system of medical devices.

Similar to the approach followed in VITAL, the modelling focuses on the interface aspects between devices only, it does not try to define a model for the implementation of any particular device.

This means that the static object model represents only information that is actually exchanged between devices. Conceptually, object instances defined in the model are contained in the MDIB (Medical Data Information Base) as defined in VITAL. The objects are accessed by the services defined in the VITAL service model.

Interfaces within a device (for example, communication layer API functions, internal programming interfaces to measurement or display subsystems etc.) are out of the scope of this standard.

The model considers general concepts that are suitable for several variations of communication profiles. It is for example independent of any specific lower layer implementation.

The model does not define any communication profiles with device type specific properties.

The Unified Modelling Language (UML) is used as the modelling notation. The Communication Controller subject as a Coad-Yourdon- style structuring element that is defined in VITAL is considered as a UML package in this standard.

The object definitions use several generic ASN.1 data type definitions from VITAL.

6.1 Domain Information Model for the Communication Subject

The Communication Subject contains objects that represent the communication sub- system of medical devices.

The primary purpose of the Communication Subject within the overall model defined by VITAL is to model information that is required for the following:

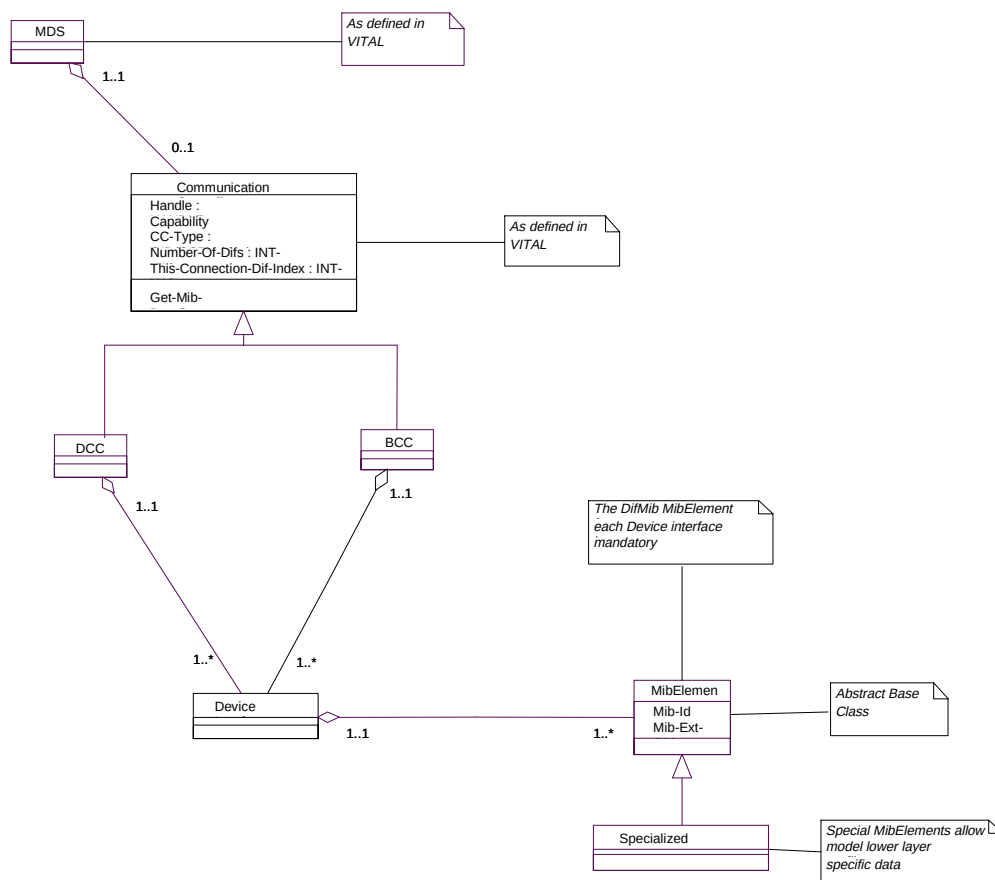
- Plug and play operation of medical devices.
- systems management of interconnected medical devices
- service and trouble shooting for the communication sub- systems of medical devices

NOTE This model does not preclude the existence of a lower layer specific or network specific management protocol. E.g. for a communication link with a TCP/IP transport system, the Simple Network Management Protocol (SNMP, RFC1441 et al) could be used as a management protocol. This may coexist with the definitions in the standard or it may be used instead of the definition in this standard.

Similar to the objects defined in the VITAL DIM, objects defined in this model represent information that is shared between devices through a point-to-point association between communicating systems. The objects are essentially used as information containers.

The object model focuses on generalized concepts for the representation of communication interface capabilities (ranging from e.g. link speeds to general Quality of Service related parameters), interface configuration aspects and statistical data (e.g. for trouble shooting). The model is independent of specific implementations of lower layer communication services.

Fig 2. shows the model for the Communication subject:



NOTE The object attributes shown in the figure are for visualization only. They are not complete. Refer to the object definitions for more details.

Figure 2 — Communication Subject Model

The model contains the following objects:

6.1.1 Medical Device System (MDS)

The Medical Device System object (MDS) is an abstraction of a device that provides medical information in the form of objects that are defined in the VITAL DIM.

The MDS object is the root of the Medical Data Information Base (MDIB) object containment hierarchy.

The object and its derived classes are defined in VITAL. It is included in the diagram to show the overall containment hierarchy.

6.1.2 Communication Controller

The Communication Controller represents the upper layer and lower layer communication profile of a medical device.

The object is the access point for retrieving Device Interface attributes and MIB Element attributes for obtaining management information related to data communications.

As a base class, the Communication Controller cannot be instantiated. There are two instantiable specializations defined (BCC and DCC). A medical device MDIB contains either no Communication Controller or one BCC or one DCC (dependent on its role).

6.1.3 Device Communication Controller (DCC)

The DCC is a Communication Controller object used by medical devices operating as agent systems (association responders).

The DCC shall contain one or more Device Interface objects.

6.1.4 Bedside Communication Controller (BCC)

The BCC is a Communication Controller object used by medical devices operating as manager systems (association requestors).

The BCC shall contain one or more Device Interface objects.

6.1.5 Device Interface

The Device Interface object represents a particular interface, or port. The port is either a logical or a physical end point of an association for which (e.g. statistical) data captured in the Management Information Base (MIB) Element objects can be independently collected.

Both an agent system and a manager system can have multiple logical or physical ports, dependent on the selected implementation of the lower layer communication system.

The Device Interface object is not accessible by CMDISE services. The object contains at least one MIB Element object (the Device Interface Mib Element, which represents device interface properties), which can be accessed by a special method defined by the Communication Controller object.

6.1.6 Management Information Base Element (MibElement)

The MIB (Management Information Base) object contains statistics and performance data for one Device Interface. The MibElement is a base class for specialized MibElement objects only. It cannot be instantiated.

Various MibElement types are defined to group management information in defined packages which can be generic or dependent on specific transport profiles.

The MibElement object is not directly accessible. Its attributes can only be accessed through a Communication Controller object. The object is not part of the device's MDIB.

6.1.7 Specialized MibElement

Management information for a communication link is sometimes very profile dependent. This standard defined two generic MibElements only:

- the mandatory Device Interface MibElement that describes properties of the device interface
- an optional general communication statistics MibElement that models typical communication statistics that are generally applicable

An informative annex defines additional MibElement objects for specific lower layer implementations that are relevant for this standard.

6.2 Object Definitions

6.2.1 Nomenclature Extensions

This standard defines extensions to the VITAL nomenclature. There are two types of extensions:

- New terms added to the existing Object Oriented Modelling Elements nomenclature table

Terms and corresponding code values are defined for new managed object classes and new attributes, events and actions defined by these managed object classes

— Definition of a new Infrastructure Elements nomenclature table

This new table contains terms and corresponding code values for infrastructure objects (which are not managed, e.g. the Device Interface object, the MIB Element object) and their attributes. The new table also contains terms and corresponding code values for profile identification and model extensions.

The new nomenclature table requires an extension of the NomPartition data type defined in VITAL as follows:

```
--
-- The following nomenclature partitions exist
--
NomPartition ::= INT-U16 {
    nom-part-unspec(0),
    nom-part-obj(1),          -- object oriented partition
    nom-part-metric(2),      -- metric (SCADA) partition
    nom-part-alert(3),       -- Alerts/ Events partition
    nom-part-dim(4),         -- Dimensions partition
    nom-part-vattr(5),       -- virtual attribute partition for operation objects
    nom-part-pgrp(6),        -- parameter group ID partition
    nom-part-sites(7),       -- measurement and body site locations
    nom-part-infrastructure(8), -- infrastructure elements partition
    nom-part-ext-nom(256),   -- Ids of other nomenclatures and dictionaries
    nom-part-priv(1024)     -- private partition
}
```

The nom-parts-infrastructure value was added to the values of the NomPartition data type as an extension.

The nomenclature table is defined in Annex B of this standard.

6.2.2 Communication Controller

Object: Communication Controller (CC)
Description: "The CC object represents the upper layer and lower layer communication profile (the application profile, the format profile and the transport profile) and provides access methods for obtaining management information related to data communications."
Derived From: Top
Name Binding: Handle
Registered as: NOM_MOC_CC (from object oriented nomenclature table)

6.2.2.1 Attributes

This object class defines the following attributes:

Table 1

Attribute Name	Attribute ID	Attribute Type	Remark	Qual.
Handle	NOM_ATTR_ID_HANDLE	Handle	ID for referencing the object	M
Capability	NOM_ATTR_CC_CAPAB	CcCapability	Bit field indicating specific capabilities of the CC implementation	M
CC-Type	NOM_ATTR_CC_TYPE	CC-Oid	could be used to specify variants, e.g. P1073, LAN, combinations,	O
Number-Of-Difs	NOM_ATTR_CC_NUM_DIFS	INT-U16	Number of device interfaces, defaults to 1 if not present. Device Interface objects are identified by their index. The index is a 16bit number between 1 and Number-Of-Difs. The list is statically configured at CC	O

			configuration time.	
This-Connection-Dif-Index	NOM_ATTR_CC_THIS_DIF_INDEX	INT-U16	Device Interface used for the current connection. 0 or not present if this cannot be determined/ specified by the implementation	O
Cc-Ext-Mgmt-Proto-Id	NOM_ATTR_CC_EXT_MNG_PROTOCOL	CcExtMgmtProto	specifies ID for an external management protocol, like e.g. SNMP or CMIP	O

The following data type definitions apply:

```
--
-- Capability specification of the Communication Controller
--
CcCapability ::= BITS-32 {
    cc-sup-ext-mgmt-protocol(0)
}
-- needs more definition
-- CC supports an external management protocol (like
-- e.g. SNMP). This requires the presence of the
-- CC-Ext-Mgmt-Proto-Id attribute.

--
-- the CC-OID data type is a regular 16bit wide OID from the infrastructure elements nomenclature
table
--
CC-Oid ::= OID-Type

--
-- The following is a list of network management protocols
-- The value range from 32768 to 65535 is reserved for manufacturer specific protocols
--
CcExtMgmtProto ::= INT-U16 {
    mgmt-protocol-snmp-v1(1),
    mgmt-protocol-snmp-v2(2),
    mgmt-protocol-snmp-v3(3),
    mgmt-protocol-cmip(16)
}
-- Simple Network Management Protocol Vers. 1
-- Simple Network Management Protocol Vers. 2
-- Simple Network Management Protocol Vers. 3
-- Common Management Information Protocol
```

This object class defines the following attribute group:

Table 2

Attribute Group	Attribute Group ID	Group Elements
Communication Controller Attribute Group (extensible attribute group)	NOM_ATTR_GRP_CC	from CC: (all attributes),

6.2.2.2 Behaviour

The Communication Controller object defines the following special methods:

Table 3

Action	MODE	Action ID	Action Parameter	Action Result
Get-Mib-Data	confirmed	NOM_ACT_GET_MIB_DATA A	GetMibDataRequest	GetMibDataResult

The following additional data type definitions are needed:

```
--
-- Data type for the ACTION INVOKE. One request can retrieve data for one device interface only.
-- Note: If the mib-id-list is empty, no MIB Element data is returned in the response
--         Valid entries in the mib-id-list are defined in the Mib-Element-List of the
--         Device Interface MIB Element object
--

GetMibDataRequest ::= SEQUENCE {
    dif-index          INT-U16,
    mib-id-list        MibIdList
}

MibIdList ::= SEQUENCE OF CC-Oid

--
-- Data type for the ACTION result
--
GetMibDataResult ::= SEQUENCE {
    dif-index          INT-U16,
    mib-data-list      MibDataList
}

MibDataList ::= SEQUENCE OF MibDataEntry

MibDataEntry ::= SEQUENCE {
    mib-id             CC-Oid,
    mib-attributes     AttributeList
}
```

6.2.2.3 Notifications

No specific events are defined for the CC object.

6.2.3 Device Communication Controller (DCC)

Object: Device Communication Controller (DCC)
Description: "The DCC is a Communication Controller object used by medical devices operating as agent systems (association responders)."
Derived From: Communication Controller
Name Binding: Handle
Registered as: NOM_MOC_DCC (from object oriented nomenclature table)

6.2.3.1 Attributes

The DCC object does not define any additional attributes.

6.2.3.2 Behaviour

The DCC object does not define any special methods.

6.2.3.3 Notifications

No specific events are defined for the DCC object.

6.2.4 Bedside Communication Controller (BCC)

Object:	Bedside Communication Controller (BCC)
Description:	"The BCC is a Communication Controller object used by medical devices operating as manager systems (association requestors)."
Derived From:	Communication Controller
Name Binding:	Handle
Registered as:	NOM_MOC_BCC (from object oriented nomenclature table)

6.2.4.1 Attributes

The BCC object does not define any additional attributes.

6.2.4.2 Behaviour

The BCC object does not define any special methods.

6.2.4.3 Notifications

No specific events are defined for the BCC object.

6.2.5 Device Interface Object

Object:	Device Interface (DIF)
Description:	"The DIF object represents a BCC or DCC communication port which is an end point of a single association for which (e.g. statistical) data are independently collected by the Communication Controller. The DIF object is not accessible by CMDISE services"
Derived From:	-
Name Binding:	-
Registered as:	NOM_CC_DIF (from infrastructure elements nomenclature table)

6.2.5.1 Attributes

This object class does not define any attributes. All its properties are captured in the Device Interface MIB Element. This MIB Element is mandatory for each instance of the Device Interface object that is supported by the Communication Controller.

6.2.5.2 Behaviour

The Device Interface object does not define any special methods.

6.2.5.3 Notifications

No specific events are defined for the Device Interface object.

6.2.6 Management Information Base Element (MIB Element)

Object:	Management Information Base Element (MIB Element)
Description:	"The MIB Element object represents management information about a specific physical or logical port of a Device Interface. The MIB Element object is an abstract base class only."
Derived From:	--
Name Binding:	--
Registered as:	NOM_CC_MIB_ELEM (from infrastructure elements nomenclature table)

6.2.6.1 Attributes

This object class defines the following attributes:

Table 4

Attribute Name	Attribute ID	Attribute Type	Remark	Qual.
Mib-Ext-Oid	NOM_CC_MIB_DATA_EXT_OID	OCTET STRING	The OCTET STRING contains a fully BER encoded registered ISO Object Identifier if the MIB Element is a registered concept. The size of the OCTET STRING shall be even, which may require a padding byte. This allows including MIB definitions from other standards here.	O

The attributes can only be retrieved by the special Communication Controller object method.

No additional data type definitions are needed.

This object class does not define any attribute group.

6.2.6.2 Behaviour

The MIB object does not define any special methods.

6.2.6.3 Notifications

No specific events are defined for the MIB object.

6.2.7 Device Interface MIB Element

Object: Device Interface MIB Element
Description: "This MIB element describes the properties of the Device Interface (object). This MIB Element is mandatory for each Device Interface of the Communication Controller"
Derived From: MIB Element
Name Binding: --
Registered as: NOM_CC_MIB_ELEM_DIF (from infrastructure elements nomenclature table)

6.2.7.1 Attributes

This object class defines the following attributes:

Table 5

Attribute Name	Attribute ID	Attribute Type	Remark	Qual.
Dif-Id	NOM_CC_MIB_DATA_DIF_ID	INT-U16	between 1 and the NoOfDifs value in the CC object.	M
Port-State	NOM_CC_MIB_DATA_DIF_PORT_ST	DifMibPortState		M
Dif-Type	NOM_CC_MIB_DATA_DIF_TYPE	CC-Oid	assumes entries in infrastructure elements NOM table	O
Active-Profile	NOM_CC_MIB_DATA_PROFILE_ID	OID-Type	This ID should contain the Profile Support Attribute ID as used in the ACSE user info structure that was negotiated in the association phase for the active profile. If no profile is active, the field should be set to 0.	O
Supported-Profiles	NOM_CC_MIB_DATA_SUPP_PROFILES	SupportedProfileList	see below	O
MTU	NOM_CC_MIB_DATA_MTU	INT-U32	Maximum Transmit Unit, in bytes	O
Link-Speed	NOM_CC_MIB_DATA_LINK_SPEED	INT-U32	in bits/s	O
Mib-Element-List	NOM_CC_MIB_DATA_MIB_ELEMENT_LIST	MibElementList	this is a list of MIB Elements supported by the Dif (in addition to this mandatory MIB Element). This assumes that the MIB elements are registered in the nomenclature.	M

The attributes can only be retrieved by the special Communication Controller object method.

The following data type definitions apply:

```
--
-- The Supported Profiles List defines which profiles are supported by the Device Interface
-- Entries in the list are Profile Support Attribute Ids as used in the ACSE user info structure
-- used for negotiating the active profile (see definition of application profiles).
-- (entries in the list are from the infrastructure elements nomenclature table)
--
SupportedProfileList ::= SEQUENCE OF CC-Oid

--
-- The Mib Element List defines which MIB elements are supported by the Device Interface
-- (entries in the list are from the infrastructure elements nomenclature table)
--
MibElementList ::= SEQUENCE OF CC-Oid

-- State of the communication port
DifMibPortState ::= BITS-16 {
    difmib-port-enabled(0),
    difmib-port-connected(1),      -- port physically connected to line/ network
    difmib-port-associated(2),     -- logical connection active on port
    difmib-port-failure(15)        -- port is in a hardware failure state
}
```

This object class does not define any attribute group:

6.2.7.2 Behaviour

This MIB Element object does not define any special methods.

6.2.7.3 Notifications

No specific events are defined for this MIB Element object.

6.2.8 General Communication Statistics MIB Element

Object: General Communication Statistics MIB Element
Description: "This MIB element represents generic communication statistics for one device interface"
Derived From: MIB Element
Name Binding: --
Registered as: NOM_CC_MIB_ELEM_GEN_COMM_STATS (from infrastructure elements nomenclature table)

6.2.8.1 Attributes

This object class defines the following attributes:

Table 6

Attribute Name	Attribute ID	Attribute Type	Remark	Qual.
Packets In	NOM_CC_MIB_DATA_PACK_IN	MibCcCounter	The number of packets received.	O
Packets Out	NOM_CC_MIB_DATA_PACK_OUT	MibCcCounter	The number of packet issued	O
Octets In	NOM_CC_MIB_DATA_OCT_IN	MibCcCounter		O
Octets Out	NOM_CC_MIB_DATA_OCT_OUT	MibCcCounter		O
Discarded-Packets-In	NOM_CC_MIB_DATA_DISC_PACKET_IN	MibCcCounter	Received Packets not delivered to upper layers	O
Discarded-Packets-Out	NOM_CC_MIB_DATA_DISC_PACKET_OUT	MibCcCounter	Packets from upper layers not sent to network interface	O
Unknown-Protocol-Packets-In	NOM_CC_MIB_DATA_UNK_PROTOCOL_PACKET_IN	MibCcCounter	Received packets with unknown protocol	O
Queue-Len-In	NOM_CC_MIB_DATA_QUEUE_LEN_IN	MibCcGauge	Size of output packet queue in bytes	O
Queue-Len-Out	NOM_CC_MIB_DATA_QUEUE_LEN_OUT	MibCcGauge	Size of input packet queue in bytes	O
Dif-Admin-Status	NOM_CC_MIB_DATA_DIF_STATE	OperationalState	Desired interface state	O
Dif-Oper-Status	NOM_CC_MIB_DATA_CUR_DIF_STATE	OperationalState	Current interface status	O
Dif-Last-Change	NOM_CC_MIB_DATA_TIME_DIF_LAST_CHANGE	AbsoluteTime	The time when the interface last changed state	O
Errors-In	NOM_CC_MIB_DATA_ERRS_IN	MibCcCounter	Corrupt received packets	O
Errors-Out	NOM_CC_MIB_DATA_ERRS_OUT	MibCcCounter	Corrupt sent packets	O
Generic-Mode	NOM_CC_MIB_DATA_COMM_MODE	MibCcCommMode	The mode of the communication.	O
Average Speed	NOM_CC_MIB_DATA_AVG_SPEED	INT-U32	In bits/s	O
Maximum Speed	NOM_CC_MIB_DATA_MAX_SPEED	INT-U32	In bits/s	O

The attributes can only be retrieved by the special Communication Controller object method.

The following data type definitions apply:

```
--
-- The Gauge type (from RFC1155) represents a non negative integer which may
-- increase or decrease, but which latches at a maximum value.
--
MibCcGauge ::= INT-U32

--
-- The Counter type (from RFC1155) represents a non negative integer with
-- monotonically increases until it reaches a maximum value, when it wraps around
-- and starts increasing again from zero.
--
```

```
MibCcCounter ::= INT-U32
```

```
--  
-- The CommMode type represents the modes of the communication that is supported  
-- by the DIF.
```

```
--  
MibCcCommMode ::= BITS-32 {  
    comm-mode-simplex(0),  
    comm-mode-half-duplex(1),  
    comm-mode-full-duplex(2)  
}
```

This object class does not define any attribute group.

6.2.8.2 Behaviour

This MIB Element object does not define any special methods.

6.2.8.3 Notifications

No specific events are defined for this MIB Element object.

6.3 Dynamic Model

6.3.1 General

Dynamic modelling is necessary to define the actual interactions between communicating medical devices.

In particular the dynamic model defines:

- The device connection state machine with all message interchanges
- Object specific dynamic behaviour, for instance the access to Communication Controller data

The dynamic models defined here ensure device interoperability regarding dynamic communication aspects.

6.3.2 Connection State Machine

The following state transition diagram shows the start-up state machine of the communication sub-system of a medical device (compare to the state machine defined in the ENV 13734 dynamic model in section 7.11.2, which used COAD-YOURLDON methodology):

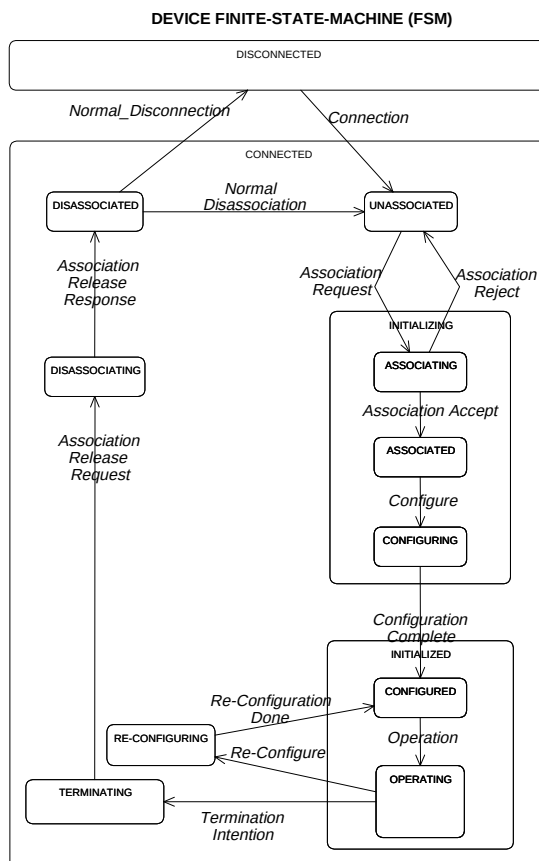


Figure 3 — Device State Machine

This state machine reflects the state transitions defined in VITAL. Specific application profiles (e.g. as defined in clause 9 of this standard) shall use this as a generic state machine, but they may define specific requirements or assumptions for the individual state transitions.

NOTE The normative profile dependent state transition tables are defined in Annex G.

6.3.3 Start- Up Object Interaction Diagram

The following object interaction diagram visualizes the start-up phase after connecting two devices (compare to the VITAL document, appendix C - the communication examples).

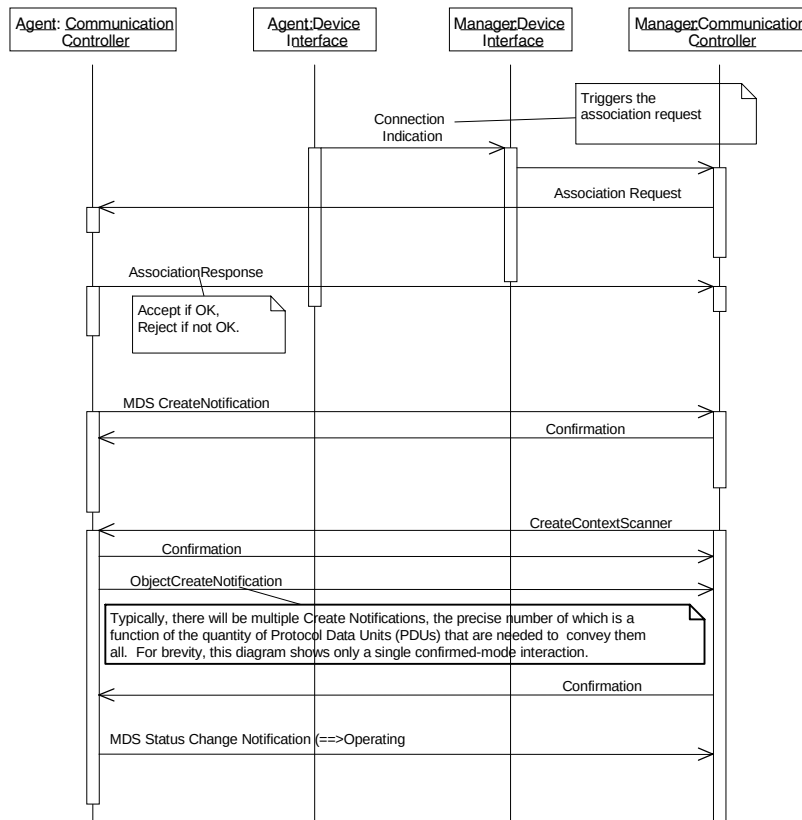


Figure 4 — Start-Up after Connection

It is assumed here, that conceptually messages are exchanged between the Communication Controller objects (using the device interface).

Some form of connection indication is necessary to make the manager system aware of a new agent on the network. This mechanism is dependent on the specific lower layer implementation and therefore it is not further defined in this standard.

This interaction diagram is applicable to the baseline application profile defined in this standard. Additional application profiles may modify this interaction diagram if required for their specific purpose.

6.3.4 Communication Subject - MIB Element Data Access

The following object interaction diagram shows how a manager system accesses the Management Information Base (MIB) data using the objects defined in this clause.

The diagram assumes that:

- an association is established between the agent and the manager
- the configuration phase is finished, the manager has an image of the agent's MDIB
- the DCC object is part of the agent's MDIB

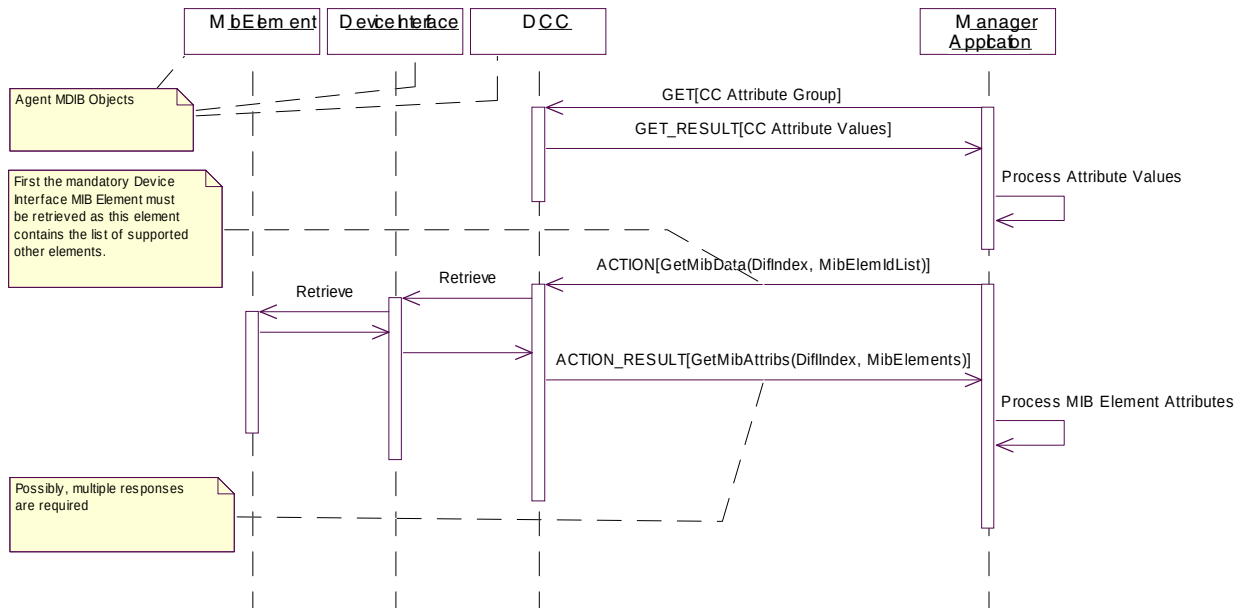


Figure 5 — MIB Element Data Access

The manager first uses the GET service to retrieve all DCC object attributes and their values. The attributes specify how many Device Interface objects exist.

The manager uses the ACTION service with the CC object defined method to retrieve the attributes of the mandatory Device Interface MIB Element. The MIB attribute variables specify if any additional MIB Elements are available for the interface.

If so, the manager can use the same ACTION command to retrieve the additional management information represented in the MIB Elements.

6.3.5 Association Negotiation

The following clauses define the behaviour of the Association Requestor and the Association Responder during the association negotiation phase. During association, in particular the use of specific application profiles and optional packages is negotiated.

6.3.5.1 Association Requestor Behaviour

- The Association Requestor (typically the manager system) prepares the AARQ ACSE PDU with the fields (e.g. the Object Ids) as defined in IEEE P1073.2.0 (MDAP)
- The Association Requestor fills out the generic ACSE User Info fields as follows:
 - For each supported protocol version the corresponding bit is set in the ProtocolVersion field
 - For each supported nomenclature version the corresponding bit is set in the NomenclatureVersion field
 - For each supported functional unit the corresponding bit is set in the FunctionalUnits field (no defined functional units exist in the current version of the standard)
 - The appropriate SystemType bits are set
 - The appropriate StartupMode bits are set
 - The optionList attribute is appended (no entries for this list are defined in this version of the standard)

- 7) For each supported application profile, the corresponding ProfileSupport attribute is added to the supportedAProfiles List. This list is sorted. The most desirable application profile is the first one in this list.
 - i) Within the structure of each ProfileSupport attribute, the list of supported optional packages is added in the form of Optional Package Specification attributes. Dependent on the role of the system, the manager or the agent Optional Package Specification attributes shall be added for each package.
- c) The Association Requestor encodes the complete AARQ PDU (the standard ACSE part in BER, the user information in MDER) and sends it to the Association Acceptor
- d) The Association Requestor now waits for the AARE message.
- e) If a time-out condition occurs, the Association Requestor sends an Association Abort message and starts over again.
- f) If an AARJ (Reject) message is received, the Association Requestor terminates.
- g) If the AARE message is received, the Association Requestor decodes the message
- h) The Association Requestor checks the standard part of the message, in particular the Object Identifier fields (see IEEE P1073.2.0 for details)
- i) The Association Requestor checks the ACSE User Info field as follows
 - 1) The ProtocolVersion and NomenclatureVersion fields shall each have one (and only one) bit set. This bit shall represent a supported version. If any of the flagged versions are not supported, the Association Requestor sends an ABORT message and terminates. Otherwise, the Association Requestor uses the flagged versions to continue processing.
 - 2) The supported Functional Units are extracted and used for this association. If a Functional Unit that is required by the Association Requestor is not supported, it sends an ABORT message and terminates.
 - 3) The SystemType and StartupMode fields are processed and stored, if required.
 - 4) The supportedAProfiles list shall contain one (and only one) entry. It is checked that this is a supported profile (versions and specific fields shall match). If the profile is not supported, the Association requestor sends an ABORT message and terminates.
 - 5) For the negotiated profile, the optional package list is parsed. For each entry
 - i) It is checked that the specific details of the optional package as defined in the Optional Package Specification attributes match the requirements
 - ii) It is checked that all required (by the end application) optional packages are present
 - iii) If any of these checks fails, the Association Requestor sends an ABORT message. It then either terminates or it re- tries, not requesting any optional packages with mismatches.
- j) The Association Requestor waits for the MDS Create Event Report message. This message may contain additional information (profile dependent and/or optional package dependent) that shall be verified before continuing the communication.

At any time the Association Requestor may receive an association abort message. In this case it has to re- start the connection sequence.

6.3.5.2 Association Responder Behaviour

- a) The Association Responder (typically the agent system) receives an AARQ ACSE PDU and decodes the standard, BER encoded ACSE defined part.

- 1) It is checked that the object identifier fields contain supported values (see IEEE P1073.2.0). Valid presentation contexts, valid application context Ids)
- 2) It is checked that a user information field is appended
- 3) If any mismatches are detected, an AARJ (reject) ACSE PDU is sent back
- b) The Association Responder prepares the standard part of an AARE (response) PDU with the fields copied from the AARQ ACSE PDU, as defined in IEEE P1073.2.0
- c) The Association Responder fills out the generic ACSE User Info fields as follows:
 - 1) The supported protocol version bits in the AARQ message are compared to the supported protocol versions of the implementation. The bit corresponding to the highest common version is set in the ProtocolVersion field. If no common versions exist, no bit is set.
 - 2) The nomenclature version bits in the AARQ message are compared to the nomenclature versions of the implementation. The bit corresponding to the highest common version is set in the NomenclatureVersion field. If no common versions exist, no bit is set. In this case, the rest of the user information, esp. the optionList and the supportedAProfiles list in the received AARQ message shall not be interpreted (and they can be left empty in the response).
 - 3) The functional unit bits in the AARQ message are compared to the supported functional units of the implementation. All bits corresponding to common functional units are set in the functionalUnits field (that translates to a bitwise logical AND operation).
 - 4) The optionList in the AARQ PDU is parsed and processed, the corresponding field in the AARE PDU is filled out.
 - 5) The supportedAProfiles list is parsed in the sequence of the entries in the list. The first entry that is found and that matches a supported profile represents the common negotiated profile. An entry for this profile is added to the supportedAProfiles list in the AARE PDU. This is the only entry in the list. If no common profiles exist, the list in the response is empty.
 - 6) The Optional Package Specification attributes for the commonly supported profile in the AARQ message is parsed.
 - i) For each *manager* optional package specification attribute in the request message checks as defined for the optional package are performed. If positive, the corresponding agent optional package specification attribute is added to the list in the response.
 - ii) If the implementation supports any optional packages in the role of a manager system, the appropriate optional package specification attributes are added as well.
- d) The Association Responder encodes the complete AARE PDU (BER and MDER) and sends it to the Association Requestor
- e) The Association Responder prepares an MDS Object Create event report and sends it to the Association Requestor
- f) The Association Responder waits for further commands from the Association Requestor.

At any time the Association Responder may receive an association abort message. In this case it has to re- start the connection sequence.

7 Messages

7.1 General

The following diagram shows the upper layer communication stack that is defined in the Vital Signs Information Representation standard (VITAL ENV 13734).

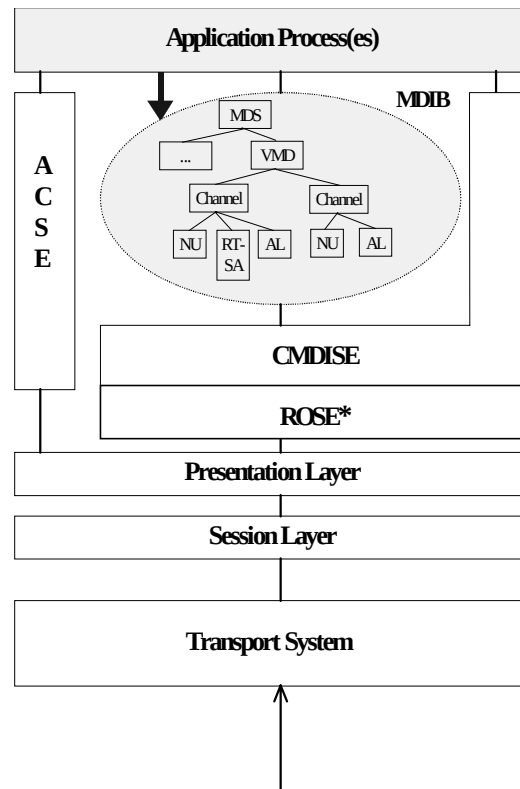


Figure 6 — VITAL Communication Stack

The diagram shows the components of the communication stack that are specified:

- ACSE is the ISO/OSI standard for association control. This standard shall be used for association and control service elements with provisions defined below.
- CMDISE is the object management service defined in VITAL. In principle, it is a lightweight version of the ISO/OSI CMISE (which is also a valid implementation of the CMDISE). The communication stack shall use either ISO/OSI standard CMISE or the CMDISE defined in VITAL as detailed below.
- The Remote Operation Service Element, ROSE, provides basic services used by the CMDISE (invoke an Operation, return the result of an operation, return an error, reject an operation). To comply with the use of optimised encoding rules, a modified version of the ROSE shall be used to work with the CMDISE. If a device uses entirely Basic Encoding Rules then ISO/OSI standard ROSE may be used as detailed below.

7.2 ACSE Protocol

General

For association control the standard Association Control Service Element as defined in X.227/ ISO/IEC 8650 shall be used.

In addition to the standard ACSE a set of application specific user information fields are given below, as well as the minimum (and mandatory) set of optional elements in the standard ACSE PDUs.

ACSE ASN.1 Message Definition

Using only the mandatory fields and the user information fields an example ASN.1 definition is given in annex C.

The ACSE messages shall be BER encoded. Furthermore, they must be wrapped in the corresponding Presentation PDU (CP: Connect Presentation, CPA: Connect Presentation Accept) and Session PDU (CN: Session Connect, AC: Session Accept).

ACSE User Information Fields

The user specific information fields in the ACSE message definition for use in the interoperability communication stack are defined in the alternative profiles sub-clauses of clause 9.

7.3 Session Layer Protocol

General

To support the standard ACSE, a minimal set of ISO/OSI standard session services is needed at least during association. Further session layer protocol details are given in the application profiles of clause 9.

7.4 Presentation Layer Protocol

General

In general, the presentation protocol allows to negotiate the abstract syntax (e.g. VITAL over CMDISE ASN.1) and transfer syntax (e.g. optimised encoding rules) between systems.

For the normal communication between medical devices (after association) the processing and messaging requirements for the presentation layer should be as small as possible.

Presentation Layer Services

The Presentation provides the following services:

- Connect Presentation (CP)
- Connect Presentation Accept (CPA)
- Presentation Data (TD)

Presentation Layer Message Definitions

Further presentation layer protocol details are given in the application profiles of clause 9.

7.5 ROSE Protocol

General

The Remote Operation Service Element is used by the CMDIP protocol. It provides a linkage between invoke messages and result messages (requests and responses) by means of invoke identifier fields. It also contains a field to distinguish the various remote operations (here: CMDISE services).

The ROSE protocol uses the same abstract syntax that is negotiated for CMDIP and the data structures from the VITAL standard. As a result, modifications to the ISO/OSI ROSE are needed to comply with the restrictions of optimised ASN.1 encoding rules.

ROSE Services

The ROSE Protocol is a relatively simple protocol defining the following services:

- Remote Operation Invoke
- Remote Operation Result
- Remote Operation Error
- Remote Operation Reject

ROSE Message Definitions

Further ROSE protocol details are given in the application profiles of clause 9.

7.6 CMDISE Protocol (CMDIP)

General

CMDISE Services

The requirements for the CMDIP protocol were already defined in the VITAL service model. The following basic services were defined in that document:

- retrieve object attribute value;
- modify object attribute value;
- invoke object defined functions;
- create and delete object instances.
- report events that occurred within an object.

The VITAL document contains also a list of necessary parameters for each of the services and the related results.

CMDIP Message Definitions

Further CMDIP protocol details are given in the application profiles of clause 9

8 Encoding rules

8.1 Overview

The purpose of the presentation layer is to assure that end systems may successfully communicate even if they use different representations. It does this by providing a common representation to be used in communication and by converting from the local representation to this common representation.

ISO Presentation Service

The presentation service provides two general categories of service. Its prime service is to handle the representation of application data so that two application entities can successfully exchange data even if they use different local representation of that data. To provide this service the presentation layer has two functions that it carries out for presentation service users.

Negotiation of transfer syntaxes:

For any type of user data, a transfer syntax is negotiated that can be used by each presentation entity for converting to or from its user's data representation.

Transformation:

The data that the user provides is converted into transfer syntax representation for transmission. Data received for delivery to the user is transformed from the transfer syntax representation to the user's representation.

Abstract Syntax and Transfer Syntax

Application exchanges involve transmitting data structures, which may be quite complex. There is a need for an abstract syntax and encoding rules to specify the representation for data types defined in the abstract syntax. ISO has issued separate documents for these two concepts:

- ISO 8824 - Abstract Syntax Notation One (ASN.1) – specifies the abstract syntax
- ISO 8825-1 : 1995 – ASN.1 Encoding Rules - Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER), Distinguished Encoding Rules (DER)
- ISO 8825-2 : 1995 – ASN.1 Encoding Rules – Specification of Packed Encoding Rules (PER)

General

The use of abstract syntax separates data types as they appear in the application protocol from how they physically appear in an application. The task of a transfer syntax notation is to represent unambiguously the values of data types as they are transmitted on the network.

There are a number of alternative transfer syntax's that may be applicable to the field of Medical Device Communications, among them are BER and PER which are ISO standards as well. The IEEE committee spent considerable effort analyzing these alternatives and decided they were inappropriate for device communication applications due to a number of special requirements:

It must be bandwidth efficient – low end medical devices may be communicating over channels of 100 bytes per second, putting a high value on the efficiency of the protocol

It must be processing efficient – the IEEE 1073 set of standards applies to devices that have very limited processing power and cannot dedicate much computational power to the communications aspect.

It must be able to encode messages in a fixed length format – this would allow sending applications to change only the value fields between successive transmissions.

These requirements are derived from the fact that potentially very simple and inexpensive devices with very limited processing power will need to comply with the standard. Unfortunately the existing encoding rules do not comply with these requirements. Both BER and PER use variable length expressions to represent integers of various sizes, this does not allow for fixed size messages. In addition, the IEEE committee prototyped some implementations using BER encoding and found that considerable processing power was required.

8.2 Medical Data Encoding Rules (MDER)

Consistent with the tradition of defining specialized encoding rules with which to meet special requirements, the IEEE 1073 committee has defined a set of encoding rules termed Medical Data Encoding Rules (MDER) within the standard IEEE 1073.2.0 (MDAP).

The MDER simplifies the transfer syntax and meets the requirements established above. In general it differs from BER and other ISO encoding rules in that:

1. No alignment is used in MDER; that is, additional bytes are not added to byte strings, for example, to obtain lengths that are divisible by two or four. However, variable length data items, i.e. strings, should have an even length for performance reasons; for example, since most data elements are 16-bit, they will not be misaligned if strings are even length.
2. Communicants are restricted to using the Network byte order (big endian) convention.
3. Data is represented in fixed size octets, rather than variable size octets as in BER or PER.

Prototyping efforts have demonstrated that the MDER is very efficient and allows for fixed size messages, which allow the device to pre-define the bulk of the message and only fill-in for the content that is variable, such as parameter values.

8.3 Abstract Syntax

The Abstract Syntax shall be ASN.1. This is a well established ISO Standard (ISO 8824) which has gained wide acceptance.

8.4 Transfer Syntax

The association protocol shall use ISO BER to provide for universal interoperability during negotiation of MDER conventions. All other PDU's exchanged in the life cycle of device-host communication should be based in MDER if possible and shall be based in either MDER or BER.

9 Application Profiles

9.1 General

The standard for Vital Signs Information Representation (VITAL ENV 13734) defines:

- an object model for the representation of data provided by medical devices that are directly connected to patients in operating rooms, intensive care rooms and labs for examination and treatment
- a service model that allows efficient access to the object instances using a data communication link between a manager system and an agent system

VITAL does not define any restrictions or recommendations on how these definitions are used in an implementation.

This chapter defines application profiles that impose restrictions for the use of the object model and the service model to facility interoperability of actual implementations.

The application profiles are independent of specific device types (device specializations, as e.g. defined in the IEEE P1073 series of standards) or specific lower communication layers (transport profiles).

This standard defines two fundamentally different application profiles:

- A Baseline Application Profile is defined for devices that support dynamic exchange of object containment trees. This profile is the recommended profile to ensure interoperability of medical devices conforming to this standard.
- A Polling Mode Application Profile is defined for devices that only support a mostly context free polling mode to access the medical data. This profile is intended for small-scale devices and special application purposes.

Optional packages defining additional capabilities are necessary for both types of profiles. The guideline that should be followed when defining optional packages is as follows:

- The optional package provides a significant additional functionality
- The objects, services and behaviour necessary for the implementation of the package can be cleanly separated from the application profile and from other optional packages
- The optional package needs some level of additional specification that should be negotiated at association time or before the configuration phase between the manager system and the agent system

The application profiles define a level of interoperability that allows conforming implementations to communicate and exchange data. This interoperability is at the level of the upper layers protocol engine only. Any application using such a protocol engine needs to verify separately if the data available from a connected device is suitable for its intended purpose.

9.2 Baseline Application Profile

9.2.1 General

The purpose of the Baseline Application Profile is to define a minimum set of requirements that an implementation has to fulfil in order to be considered a valid implementation of the context sensitive communication model as outlined in VITAL.

The level of specification provided here supports the implementation of interoperable (at the upper layers) medical devices.

Interoperable means in this context that an implementation can successfully proceed through the communication state machine to a configured or operable state and can start to exchange data. This does not imply that the data provided by a medical device is suitable for any specific end user application.

An explicit containment tree (Medical Data Information Base, MDIB as defined in VITAL) is exchanged between the manager system and the agent system during configuration.

Scanner objects that are instantiated by the agent device itself provide data access. Scanners send instantaneous object attribute updates, the manager device does not have to actively poll the agent device.

The Baseline Application Profile defines the minimum requirements for the full, context sensitive communication model defined in VITAL. This profile is the recommended profile for compliant implementations.

Specific extensions to this profile can be made by negotiating the use of well- defined optional packages for additional levels of functionality.

9.2.2 Services

9.2.2.1 ACSE Services

The Baseline Application Profile requires the Association Control Service Element (ACSE, ISO/IEC 8650).

Table 7 specifies the service requirements for the ACSE services for agent and manager systems. Note that both the agent and the manager may optionally request an orderly association release or may abort the association.

Table 7

Service	Invoke/ Provide	Support (Manager)	Support (Agent)	Comment
Association Request	Invoke	M	N/A	Agent does not request an association
	Provide	N/A	M	Manager does not reply to an association request
Association Response	Invoke	N/A	M	
	Provide	M	N/A	
Association Release Request	Invoke	O	O	
	Provide	M	M	
Association Release Response	Invoke	O	O	
	Provide	M	M	Both systems shall be able to respond to an association release request
Association Abort - Service User	Invoke	O	O	
	Provide	M	M	
Association Abort - Service Provider	Invoke	O	O	
	Provide	M	M	

To ensure interoperability, an implementation shall use the ISO registered object identifiers assigned in IEEE P1073.2.0 (MDAP) in all association messages , e.g. for the application name, the abstract syntax and the transfer syntax.

Further specifications for association messages (e.g. usage of mandatory fields) defined in P1073.2.0 shall be followed by an implementation.

An implementation shall use the associate user information data that is defined in IEEE P1073.2.0 and the specific data content defined in the supported application profile as defined in this standard.

It is specifically noted here that in the case of a disconnection event an agent device or a manager device may not be able to invoke an abort or release service to inform the peer system.

9.2.2.2 CMDISE Services

The CMDISE services are defined in VITAL. They are required for the communication state machine and to communicate MDIB data.

Table 8 specifies the service requirements for the CMDISE services for agent and manager systems.

Table 8

Service	Invoke/ Provide	Support (Manager)	Support (Agent)	Comment
GET Service	Invoke	O	N/A	
	Provide	N/A	M	At least the MDS attributes can be accessed with GET service
SET Service	Invoke	O	N/A	
	Provide	N/A	C	See notes
Confirmed SET Service	Invoke	O	N/A	
	Provide	N/A	C	See notes
Event Report Service	Invoke	N/A	O	Dependent primarily on scanner set-up
	Provide	M	N/A	
Confirmed Event Report Service	Invoke	N/A	M	needed for the MDS Notifications and episodic scanner events
	Provide	M	N/A	
ACTION Service	Invoke	O	N/A	
	Provide	N/A	C	See note 5
Confirmed ACTION Service	Invoke	O	N/A	
	Provide	N/A	C	See note 5
CREATE Service	Invoke	M	N/A	The CREATE service is mandatory to create a context scanner object instance.
	Provide	N/A	M	
DELETE Service	Invoke	O	N/A	
	Provide	N/A	O	

Invoking a CMDISE service includes sending the service invoke message and receiving the corresponding response message (response message only for confirmed mode services and GET service)

Providing the GET service or a confirmed CMDISE service includes receiving and processing the service invoke message and sending the appropriate response message.

The agent may support a DELETE service for the Context Scanner, which is the only object in this profile that can be dynamically created. However, deleting the Context Scanner is not recommended.

The manager may invoke a particular service in confirmed or in unconfirmed mode. Therefore, the agent system shall provide both services.

The agent support of the SET service is conditional (C). This means: In the case that the agent supports an object that has set-able attributes, the agent shall provide the SET service. Otherwise, the service support is not applicable.

NOTE Typically an agent implementation using the baseline profile has to support the SET service because the scanner operational state attribute is always settable.

The agent support of the ACTION service is conditional (C). This means: In the case that the agent supports an object that defines an object specific method, the agent shall provide the ACTION service. Otherwise, the service support is not applicable.

NOTE Optional packages may require additional services. This list represents the requirements for the baseline profile without optional packages only.

9.2.3 Support for MDIB Objects

In the agent object model, Metric objects shall be contained in Channel objects. This defines a fixed containment hierarchy below the VMD object (VMD - Channel - Metric) which reduces the variability and thus simplifies manager implementations

Table 9 lists the objects that are in the scope of the Baseline Application profile.

Table 9

Object	Support (Manager)	Support (Agent)	Comment
VMD	M	M	
Channel	M	M	
Numeric	M	O	
Real-Time Sample Array	M	O	see notes
Time Sample Array	M	O	
Distribution Sample Array	M	O	
Enumeration	M	O	
Alert	M	O	The agent shall support at most one type of alert related object class (multiple instances are allowed)
Alert Status	M	O	The agent shall support at most one type of alert related object class (multiple instances are allowed)
Alert Monitor	M	O	The agent shall support at most one type of alert related object class (multiple instances are allowed)
Simple MDS	M	O	An agent has to support at least one MDS Type
Hydra MDS	M	O	An agent has to support at least one MDS Type
Composite Single Bed MDS	M	O	An agent has to support at least one MDS Type
Event Log	M	O	
Battery	M	O	
Episodic Configurable Scanner	M	O	
Periodic Configurable Scanner	M	O	
Fast Periodic Configurable Scanner	M	O	see notes
Context Scanner	M	M	
Alert Scanner	M	C	Mandatory for the agent if alert related objects are supported
Communication Controller	M	O	Includes mandatory support for Device Interface and MIB Element objects.

Mandatory (M) elements shall be supported by a conforming implementation.

Optional (O) elements may be supported by a conforming application, dependent on its functionality and the data that it provides or processes.

Conditional (C) elements depend on the usage of specific optional packages.

Objects, which are not listed in this table, are not supported by the baseline profile. Optional packages as defined in the annex may support additional objects defined in VITAL.

Implementations should ignore any unknown objects in the MDIB, if possible without terminating the association. This behaviour provides a level of interoperability even in presence of future extensions of the object model.

NOTE 1 A manager application has to support all objects that could be part of an agent's MDIB in the baseline profile. Dependent on the end user functionality, this may mean that the manager ignores a certain sub- set of these objects. E.g. if the end user application just needs numerical measurements it may still need to deal with Time Sample array objects (at least as an entry in the context scanner object create event report) provided by a specific device but it can ignore the information provided by the TSA object.

NOTE 2 The handling of real-time waves (RT-SA objects and the fast periodic configurable scanner) is included in the baseline profile and it does not constitute a separate optional package. Reasoning: For the handling of real-time waves additional information is needed by the manager (sample frequencies, update periods, sample formats, compression etc.). All this information is included in the form of static attributes of the RT-SA objects in the containment tree. So instead of negotiating these at association time it is defined here that the manager has to check the RT-SA object static attributes in the containment tree. If the manager detects that it cannot support the agent's configuration it just does not enable the fast periodic scanner (and it notifies the end user as appropriate).

NOTE 3 The handling of alert related objects is included in the baseline profile. If a manager application does not require alert related data, it may ignore the objects in the MDIB. This manager application would not enable the alert scanner to avoid frequent attribute updates.

9.2.4 Extensions of MDIB Object Definitions

9.2.4.1 MDS Object

9.2.4.1.1 System Specification Attribute Contents

For the baseline application profile, it is recommended that the MDS object provides a standard entry in the System-Specification attribute (inherited from the VMS object), if it conforms to any medical device specialization standards (e.g. the P1073.1.3 series).

In this case, the System-Specification attribute should contain the MedDevSpec-Std-Support entry that contains a list of nomenclature Ids from the infrastructure elements nomenclature table representing the specific standard.

Table 10

Component Capability ID	Registered as	Data Type	Remark	Qual.
MedDevSpec-Std-Support	NOM_MED_DEV_SPEC_STD_SUPPORT	MedDevSpecStdSupport	This component contains the list of supported medical device specialization standards	O

This entry in the System-Specification attribute shall list all medical device specialization standards that the device conforms to.

The component capability ID (component_capab_id) field in the System-Specification MDS attribute is defined as a PrivateOid data type in VITAL. This standard overrides this definition. The value of this field shall be taken from the infrastructure elements nomenclature table, as defined in the nomenclature extensions in annex B.

If private (manufacturer specific) entries in the System-Specification attribute are needed by an implementation, the nomenclature IDs for these entries shall be from the private partition of the same nomenclature table as specified for the entry defined here.

The following additional data type is necessary:

```
--  
-- The MedDevSpecStdSupport is a list of nomenclature object Ids that identify any particular upper  
layer medical  
-- device specialization standards that the device conforms to  
-- The object IDs in the list shall be from the infrastructure elements nomenclature table.  
--  
MedDevSpecStdSupport ::= SEQUENCE OF OID-Type
```

If supported, the agent device shall provide the System-Capability attribute as part of the MDS Create Notification.

9.2.4.1.2 System Identification Attribute Contents

The System-Id MDS attribute is defined in VITAL as mandatory.

For devices that are not serialized or that cannot be electronically serialized (e.g. legacy devices), the System ID attribute shall use one of the following alternatives:

- An OCTET STRING containing 8 octets with the hexadecimal value 0
- An OCTET STRING containing 3 octets (24bits) with the IEEE assigned unique EUI-64¹⁾ manufacturer code followed by 5 octets with the hexadecimal value 0 to indicate that the device is not serialized
- An OCTET STRING with a length different than 8 octets to indicate that the identification does not conform to the EUI-64 rules

For all other devices the System ID attribute shall contain a unique 64bit (8 octet) EUI-64 identifier. The EUI-64 identifier is constructed from a 24bit manufacturer ID assigned by the IEEE and a 40bit unique identifier assigned by the manufacturer.

The unique identifier shall identify the medical device, independent from the communication port used to communicate with the device.

NOTE The EUI-64 documentation defines a method for mapping a EUI-48 identifier (a IEEE 802.3 MAC address) uniquely to a EUI-64 identifier.

9.2.5 Support for MDIB Object Attributes

For all MDIB objects that are supported by the agent system (as specified in the object containment tree/ the MDIB) the attribute and attribute group qualification as mandatory, conditional or optional that is defined in VITAL is applicable.

A manager implementation has to be able to process all potential attributes of the supported objects (which may mean that the application ignores unknown attributes and the corresponding attribute value). It is recommended that the implementation logs the occurrence of unsupported attributes.

9.2.6 Dynamic Behaviour

An informative state-transition table is given in annex G.

¹⁾ EUI-64 is a trademark of IEEE

9.2.6.1 Start-up System Interaction

The following figure shows the dynamic system interaction with respect to the exchange of upper layer communication messages for the baseline application profile in the start-up phase.

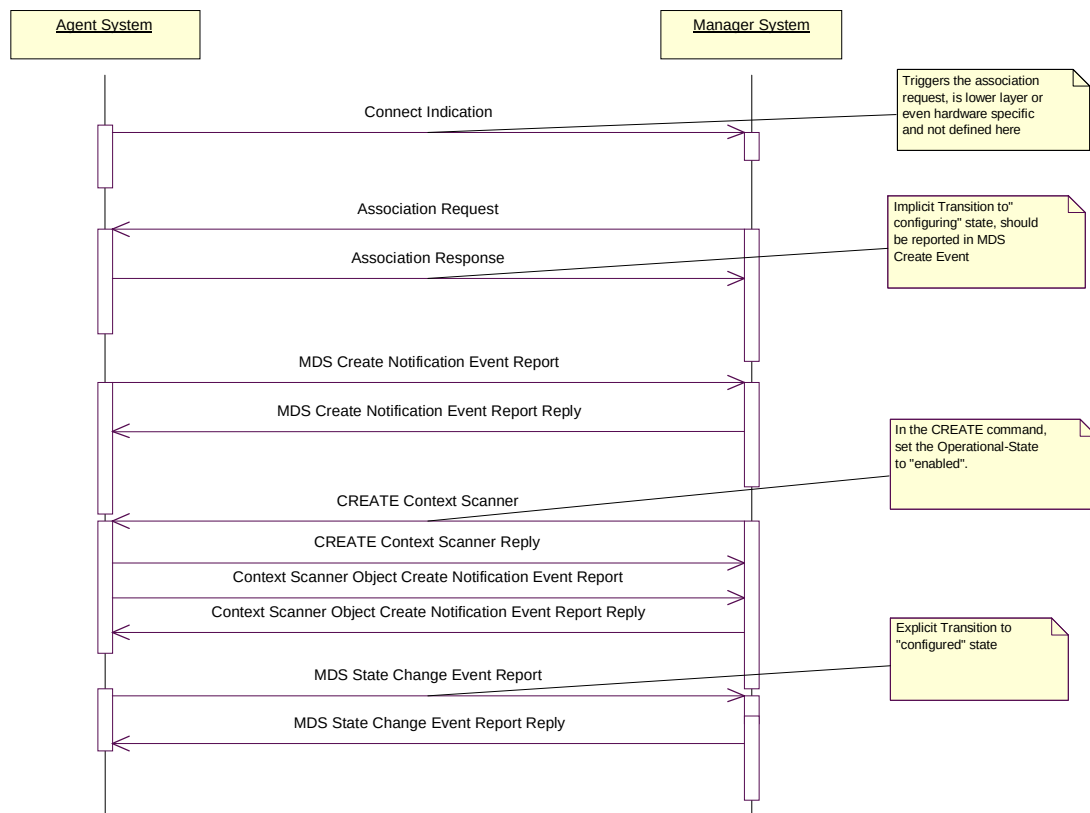


Figure 7 — Baseline Application Profile System Interaction

The negotiation procedure for the association (ACSE) messages is defined in the domain information model clauses of this standard. At the end of the association phase the system implicitly transitions to *Configuring* state, according to the MDS state machine. For the agent system it is recommended to report this state in the MDS-State attribute within the MDS Create Notification Event Report.

After start-up, the following state transitions shall be reported in form of confirmed MDS State Change notifications by the agent system:

- Transition to *Configured* state
- Transition to *Re- configuring* state (applicable only for devices supporting re- configuration during normal operation)

The agent system may optionally report any additional MDS state changes according to the state transition diagram defined in VITAL and the model clause in this standard.

Dependent on the size of the MDIB, the agent may use multiple Context Scanner Object Create Notification Event Reports to send the containment tree and static object attributes.

9.2.6.2 Confirmed Scanner Dynamic Behaviour

The following clauses specify the state handling for the confirmed scanner object implementation within an agent system and a manager system.

Confirmed scanners require explicit event report result messages from the manager system. The *transmit window size* scanner object attribute defines how many unconfirmed event reports might be

outstanding at any given time as a maximum. A *scan report number* field is included in every scanner event report as a sequence number. A *confirm time-out* attribute defines the time period before the scanner object re- sends an unconfirmed event report.

9.2.6.2.1 Agent Implementation

The following state diagram shows the dynamic behaviour of a confirmed scanner object within an agent implementation.

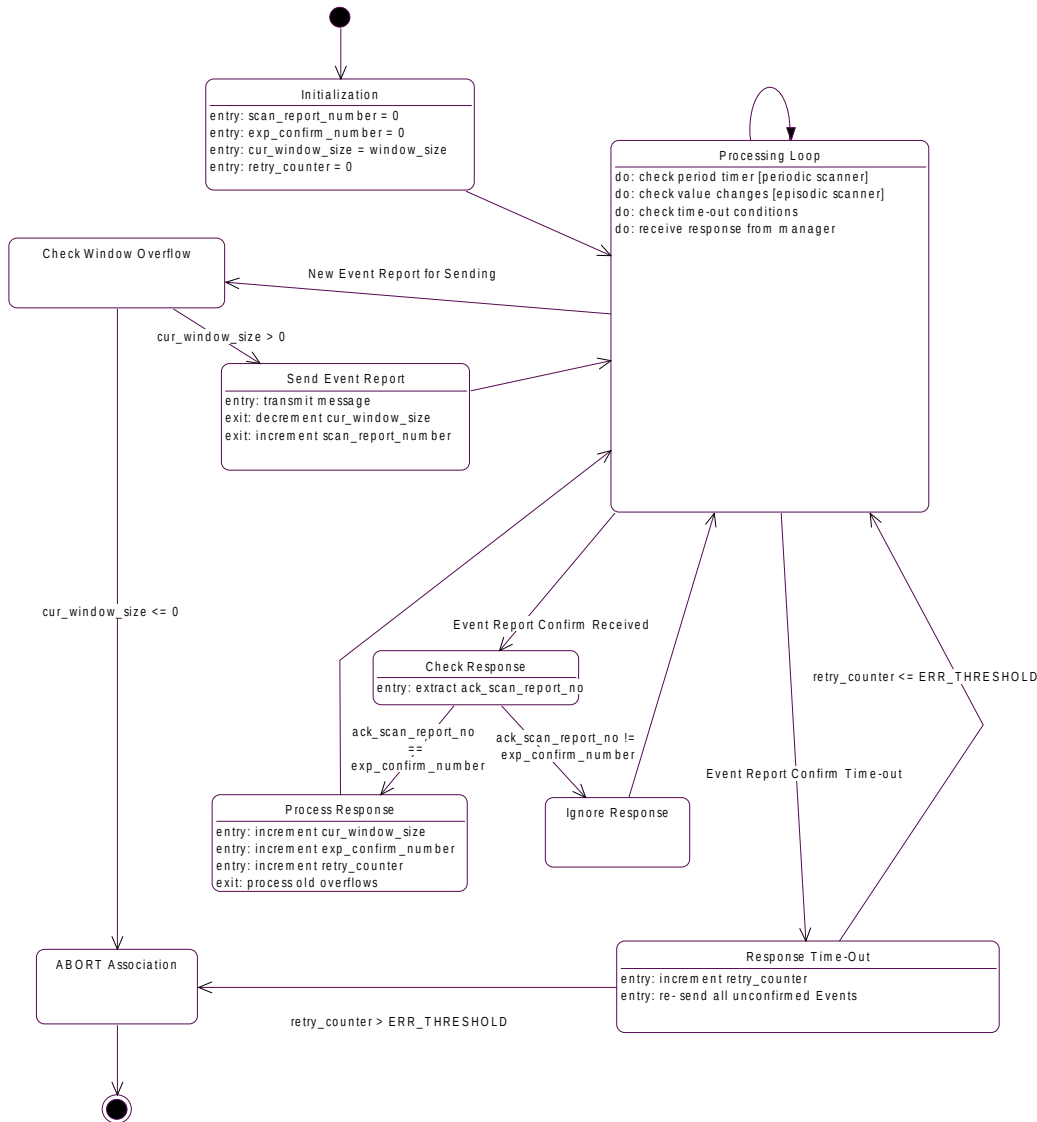


Figure 8 — Confirmed Scanner State Diagram

In a main processing loop the scanner object checks if either a new event report message shall be sent or a time-out condition (missing event report reply from the manager) occurs.

A new event report is only sent if the transmit window size is not exceeded. If the transmit window size is exceeded or the manager does not send an event report reply after some retries, the association between agent and manager is aborted because a reliable communication is no longer possible.

The scan report number field in new scanner event report messages shall be incremented by one for subsequent messages. Re- sent event reports shall keep their original scan report number.

NOTE 1 An agent may implement more fault tolerance in the case of a transmit window overflow condition, e.g. by using a buffer scheme for event report messages that cannot be send due to an empty transmit window.

NOTE 2 When processing the various counters in the state machine, an implementation should ensure that it handles counter wrap-around situations correctly (the *scan_report_number* e.g. is a 16bit counter only).

NOTE 3 Even a scanner operating in episodic mode should have an internal period that it uses to check for attribute value changes. This period defines the maximum frequency of event report messages that are sent by this scanner. An implementation shall specify this period.

9.2.6.2.2 Manager Implementation

The following state chart defines the behaviour of the manager that controls the confirmed scanner object in the agent (the scanner manager implementation).

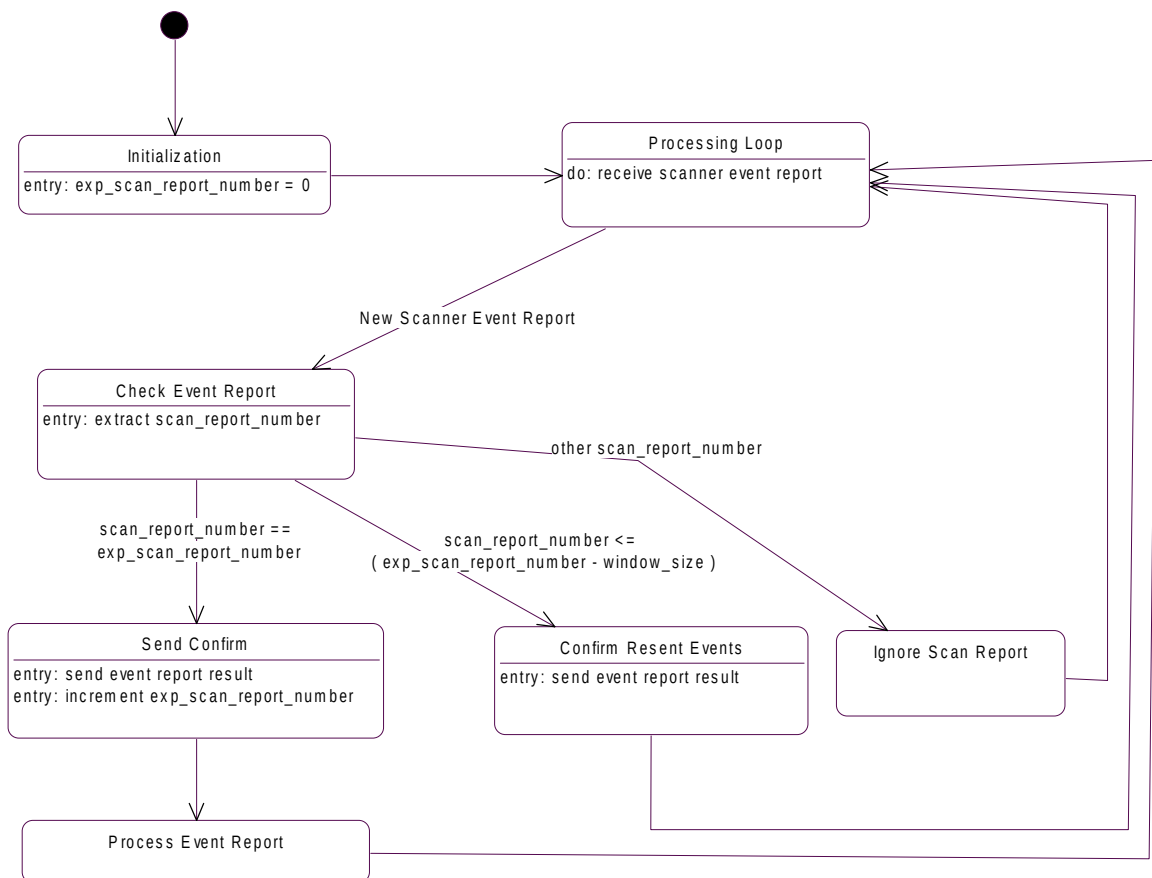


Figure 9 — Scan Manager State Chart (Confirmed Mode)

After initialization the scan manager waits for event report messages from the scanner object in the agent in a main processing loop. The scan manager receives scanner event reports in the correct sequence only. Other, reports with unexpected scan report numbers are discarded and not further processed.

The *Confirm Resent Events* state covers the case where an event report result message that was previously sent was lost on the communication link and caused a re- transmission of the original event report by the scanner (agent).

NOTE 1 When processing the various counters in the state machine, an implementation should ensure that it handles counter wrap-around situations correctly (the *scan_report_number* e.g. is a 16bit counter only).

NOTE 2 Instead of ignoring scanner event reports with a scan report number that does not match the manager may implement a flexible recovery mechanism (e.g. by buffering reports) as long as it sends out event report result messages in the correct sequence only.

9.2.6.3 Unconfirmed Scanner Dynamic Behaviour

For unconfirmed scanner event reports the manager system shall not send any event report result messages.

9.2.6.3.1 Agent Implementation

The agent implementation of an unconfirmed scanner object shall increment the scan report number by one for each subsequent event report.

NOTE When processing the various counters in the state machine, an implementation should ensure that it handles counter wrap-around situations correctly (the *scan_report_number* e.g. is a 16bit counter only).

9.2.6.3.2 Manager Implementation

The scanner manager implementation for unconfirmed scanner objects shall check the scan report number field in the received event report messages. It shall reject any duplicate event reports for further processing.

The scanner manager for unconfirmed scanner objects should abort the association if too many missing event reports are detected in a given time period.

9.2.7 Specific Restrictions

9.2.7.1 Encoding Rules

The Medical Device Encoding Rules (MDER) as defined in IEEE P1073.2.0 shall be used for the encoding of the data structures in VITAL and in this standard. The ACSE messages, with the exception of the user information field, shall be defined in the Basic Encoding Rules (BER). The ACSE user info field shall be encoded in MDER.

Specifically the encoding of the FLOAT-Type data type shall be the Medical Device Numeric Format (MDNF) defined in IEEE P1073.2.0. VITAL just maps this type to an opaque 32bit as a placeholder.

9.2.7.2 Usage of the OCTET STRING Data Type

To provide for an efficient encoding and decoding across different computing platforms the length of the OCTET STRING (the number of bytes in the structure) shall be even.

The format, the character set and the language of any user readable text contained in an OCTET STRING is not normative unless explicitly defined in VITAL. To improve interoperability between applications it is recommended to use a 16bit UNICODE character set with NULL character termination for any readable strings.

9.2.7.3 CMDIP/ ROSE Message Definitions

The CMDIP/ ROSE Message Definitions shall be as defined in IEEE P1073.2.0. The CMIP*/ ROSE* optimized versions shall be used.

9.2.7.4 Session Layer Protocol

The session layer protocol shall be as defined in IEEE P1073.2.0. The MDAP Session Layer extensions as defined in this standard shall be used.

The usage of the session layer coalescing service can be negotiated as an option in the baseline profile.

9.2.7.5 Presentation Layer Protocol

The presentation layer protocol shall be as defined in IEEE P1073.2.0. The MDAP Presentation Layer extensions (MDAP-TD PDU) shall be used.

9.2.7.6 Object Attribute Retrieval

The agent system provides read access to object attributes with a granularity of a complete attribute group.

NOTE This means the manager can e.g. invoke the GET service with an attribute group ID as a parameter. Scanner scan-list attributes contain attribute group IDs, not individual attribute IDs.

Table 11 defines the way in which the agent shall provide access to object attributes for the manager system for the various attribute groups defined in VITAL.

Table 11

Attribute Group	Access
VMO Static Context Group	Via Context Scanner only
VMO Dynamic Context Group	Via specific Episodic Scanner instance that can be configured or that is configured to scan this attribute group.
Metric Observed Value Group	As flagged in the MetricAccess field of the Metric-Spec static attribute of the metric object instance (scanner access or GET service). It is recommended for an agent device implementation to support scanner access.
Alert Group	Via Alert Scanner only
Alert Status Group	Via Alert Scanner only
Alert Monitor Group	Via Alert Scanner only
VMD Application Attribute Group	Accessible by the GET service (attribute group ID provided in the GET request) only.
VMD Production Attribute Group	Accessible by the GET service (attribute group ID provided in the GET request) only.
System Identification Attribute Group	Reported in the MDS-Create-Notification event report. Also accessible by the GET service (attribute group ID provided in the GET request)
System Application Attribute Group	At least the static attributes <i>System-Capability</i> and <i>System-Specification</i> shall be reported in the MDS-Create-Notification event report. The <i>Mds-Status</i> shall be reported in the form of an MDS Attribute Update Event Report as defined in clause 9.2.6.1 (Start-up System Interaction). It is recommended to report this attribute also in the MDS Create Event report at system start-up time. All group attributes are accessible by the GET service (attribute group ID provided in the GET request) [see also note 2]
System Production Attribute Group	Accessible by the GET service (attribute group ID provided in the GET request) only.
Battery Attribute Group	Accessible by the GET service (attribute group ID provided in the GET request) only.
Scanner Attribute Group	Via Context Scanner (static attributes, if applicable) and GET request
Communication Controller Attribute Group	Accessible by the GET service (attribute group ID provided in the GET request) only.

NOTE The MDS defined event Mds-Create-Notification contains attributes from the System Identification Attribute Group and the System Application Attribute Group that are necessary to determine the system capabilities and the system type.

It is not recommended for an implementation to send unsolicited event reports that cannot be configured or controlled by the peer system. The reporting of an MDS state change is an exception. Dependent on the specific application, other exceptions may exist, esp. for dynamic MDS attributes (e.g. to flag a transition from line power to battery power).

If the agent device does not support any attribute within the requested attribute group it shall respond to a GET request with an empty attribute list in the response message.

If the requested attribute group is not defined for the specified object the agent shall respond with a CMDIP error message as defined in P1073.2.0 (MDAP).

Access to attribute groups that are part of an Optional Package is defined as part of the specific Optional Package.

9.2.7.7 Object Attribute Modification

The agent system shall support modification of attributes using the SET service according to the following table:

Table 12

Object	Attribute	Remark	Qual.
Scanner	Operational-State	Required to enable/ disable the scanner object	M
Scanner	Scan-List	Only if indicated in scanner attributes	O
MDS	Soft-Id		O
MDS	Operation-Mode		O
MDS	Application-Area	Recommended if agent is dependent on application area	O
MDS	Patient-Type	Only if not present in Patient-Demographics	O
MDS	Date-and-Time	Recommended if not date and time cannot be set via special lower layer service	O
MDS	Altitude	Recommended if agent is dependent on altitude value	O
MDS	Line-Frequency	Recommended if agent is dependent on line frequency value	O

Mandatory (M) means that the attribute shall be present and SET access to the attribute shall be provided. Optional (O) means that the attribute may be present and SET access to the attribute may be provided.

An agent may support SET access to other attributes. All attributes that can be modified by the SET service shall be listed by an implementation as part of the conformance statement.

9.2.7.8 Composite MDS Object

The baseline profile supports the Composite Single Bed MDS object. In addition to the normal relations of MDS objects defined in VITAL, this object supports containment relations to other MDS objects.

For the baseline profile, the following restrictions apply:

9.2.7.8.1 A Composite Single Bed MDS object can contain (in form of a containment relation in the MDIB) Simple MDS objects and/ or Hydra MDS objects and/ or other Composite Single Bed MDS objects (in addition to other containment relations defined in VITAL).

9.2.7.8.2 Hydra MDS objects and Simple MDS objects cannot contain (in form of a containment relation in the MDIB) any MDS objects

9.2.7.8.3 An implementation shall specify the maximum MDS containment hierarchy depth in the Profile Support Attribute. It is recommended that the MDS containment tree depth does not exceed 3.

NOTE The MDS containment hierarchy depth is calculated as follows: The root of the object containment tree (MDIB) is an MDS object. Follow the longest path through the containment tree with just MDS objects. The

number of MDS objects in this path is the MDS containment hierarchy depth. Example: A Simple MDS or a Hydra MDS does not contain any other MDS objects, so the depth is 1. A Composite Single Bed MDS as a MDIB root that contains one Hydra MDS defines a hierarchy depth 2. A Composite Single Bed MDS as a MDIB root that contains another Composite Single Bed MDS which itself contains a Hydra MDS defines a hierarchy depth 3.

9.2.7.8.4 An MDIB with multiple MDS objects may use multiple naming contexts. A naming context is a set of objects within the MDIB for which the handle attribute can be used as unique object identification.

NOTE Objects within one MDIB that are in different naming contexts are distinguished by means of the context ID field in the object global handle attribute.

9.2.7.8.5 The agent system shall provide a consistent time stamping of information even if the multiple MDS objects represent multiple physical devices.

9.2.7.9 Alert Object Usage

An implementation deriving patient physiological alarm data or technical alarm data needs to provide either Alert Monitor objects or Alert Status objects or Alert objects in the MDIB. As defined in VITAL, an implementation shall only implement one type of alarm related object class.

If an implementation derives an overall alarm status for the medical device for user presentation (alarm lights, alarm sound, suspend and silence user interaction) it shall use the Alert Monitor object to represent the alarm status of the device.

NOTE The Alert Monitor has means to express the overall alarm status of a device related to visible alarm conditions, audible alarm conditions and user interactions. It also can express special alarm processing modes, e.g. *latching* alarm conditions. Therefore, the Alert Monitor object is the only suitable object for implementations with these capabilities.

In all other cases it is recommended to use the Alert Status object for representing alarm related data.

NOTE The Alert Status object represents active alarm conditions of a VMD sub- system. This allows some prioritization of alarm conditions at the VMD level, which is difficult to achieve in later processing steps without implementation specific knowledge. Therefore the Alert Status object is a suitable input for further alarm processing steps.

The usage of the Alert object is not recommended in the baseline profile.

9.2.7.10 Scanner Configuration

As long as the Configurable Scanners Optional Package is not used the following clauses shall be applicable:

9.2.7.10.1 The scanner objects are instantiated by the agent device itself. They are reported as part of the object containment tree (MDIB) by the context scanner, together with all implemented static attribute values²⁾.

An exception to this is the context scanner object. This object shall be explicitly created by the manager, using the CMDISE CREATE service.

9.2.7.10.2 The scanner instances by default shall be disabled (the Operational-State attribute contains the value *disabled*). The manager can enable the scanner by using the SET service to set the Operational-State attribute to *enabled*.

For the context scanner only, the manager can enable the context scanner in the CREATE command by setting the Operational-State attribute to *enabled*.

²⁾ For this purpose all scanner attributes which cannot be modified during an association are considered static.

While in the disabled state, the scanner shall not scan attributes, even when the scanner supports the *extensive* scan mode. This ensures that no outdated information is included in event reports when the scanner is enabled.

Upon the state transition from *disabled* state to *enabled* state, an unconfigurable or an episodic scanner object shall internally run a refresh cycle so that all available scanned attributes will automatically be updated (included in the first set of scan reports) without further intervention by the manager.

The manager can disable the scanner by using the SET service to set the Operational State attribute to *disabled*. The scanner shall stop sending event reports in this case. For scanners operating in confirmed mode the scanner state machine runs to completion for any not yet confirmed event reports. This may result in re- sent event reports in the case of time-out conditions.

9.2.7.10.3 The scanner instances shall support the auto- configuration modes (the Scan-Config-Limit attribute shall have the *auto-init-scan-list* and *auto-updt-scan-list* bits set).

9.2.7.10.4 Scanners shall support the Scan-List attribute, if it is defined for the specific scanner object type. The Scan-List should be accessible by the GET service (specifying the Scanner Attribute Group as a parameter).

9.2.7.10.5 Entries in the Scan-list (*ScanEntry* data type) shall contain the global handle of the scanned object and the attribute group ID. This means the granularity for scanning object attributes is the attribute group, not an individual attribute.

The scanners may support modification of the Scan-List attribute with the SET service. If this is not supported, the scanner shall set the *no-scan-list-mod* bit in the Scan-Config-Limit attribute.

Optionally, the scanner may emit an attribute update event report (inherited from the Top object) in case the Scan-List is internally changed (e.g. as a result of an internal re- configuration).

9.2.7.10.6 All Episodic scanners shall support the confirmed mode. Episodic scanners should support the Refresh object method.

9.2.7.10.7 All Periodic Scanners shall support the unconfirmed mode.

As an exception, an Alarm Scanner scanning Alert Status objects or Alert objects should support the confirmed mode.

NOTE Alert Status and Alert objects are both considered input to an alarm manager process. Here, no transition must be lost in order to allow the implementation of latching alarm conditions. So despite the communication overhead for confirming periodic (potentially high frequency) event reports, confirmed mode is recommended in this case (unless a reliable transport layer is used). For scanning an Alert Monitor object, the unconfirmed mode is acceptable.

9.2.7.10.8 The agent implementation shall guarantee that a scanner event report does not contain data about object instances in the MDIB that were not previously created by the Context Scanner.

NOTE This means that the agent must ensure that the manager has received and processed (has sent an event report reply) an object create notification from the context scanner before including information about this object in other scanner event reports.

9.2.7.10.9 The manager implementation shall discard any information in scanner event reports that relate to objects which are not in its current copy of the containment tree. The manager may send an error indication (ROSE error message), indicating the condition (*no such object instance* error code).

NOTE In contrast to the previous clause this clause deals with the case that a scanner event report contains object attributes for an object that was already deleted in the containment tree.

9.2.7.10.10 The baseline profile does not support settable scanner reporting intervals for periodic configurable scanner objects. This means that the *Reporting Interval* attribute cannot be modified by the manager system.