IEEE P11073-10201™/D2.1.109 Draft Standard for Health informatics — Point-of-care medical device communication — Part 10201: Domain information model

Sponsor

IEEE 11073 Standards Committee
of the
IEEE Engineering in Medicine and Biology Society

Approved <XX MONTH 20XX>

IEEE-SA Standards Board

content. Other entities seeking permission to reproduce this document, in whole or in part, must also obtain permission from the IEEE Standards Department.

IEEE Standards Activities Department
445 Hoes Lane
Piscataway, NJ 08854, USA

Abstract: Within the context of the ISO/IEEE 11073 family of standards for point-of-care (POC) medical device communication (MDC), this standard provides an abstract, object-oriented domain information model that specifies the structure of exchanged information, as well as the events and services that are supported by each type of object. All data structure elements are specified using abstract syntax (ASN.1) and may be applied to many different implementation technologies, transfer syntaxes, and application service models. Core subjects include medical, alert, system, patient, control, archival, communication, and extended services. Model extensibility is supported, and a conformance model and statement template is provided.

Keywords: abstract syntax, alarm, alert, ASN.1, information model, medical device communications, medical information bus, MIB, point-of-care, POC, object-oriented, patient, remote control

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

PDF: ISBN 978-0-XXXX-XXXX-X STDXXXXX
Print: ISBN 978-0-XXXX-XXXX-X STDPDXXXXX

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page, appear in all standards and may be found under the heading "Important Notices and Disclaimers Concerning IEEE Standards Documents." They can also be obtained on request from IEEE or viewed at http://standards.ieee.org/IPR/disclaimers.html.

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents (standards, recommended practices, and guides), both full-use and trial-use, are developed within IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association ("IEEE-SA") Standards Board. IEEE ("the Institute") develops its standards through a consensus development process, approved by the American National Standards Institute ("ANSI"), which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE Standards are documents developed through scientific, academic, and industry-based technical working groups. Volunteers in IEEE working groups are not necessarily members of the Institute and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE Standards do not guarantee or ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to: results; and workmanlike effort. IEEE standards documents are supplied "AS IS" and "WITH ALL FAULTS."

Use of an IEEE standard is wholly voluntary. The existence of an IEEE standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person

utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, or be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to comments or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in revisions to an IEEE standard is welcome to join the relevant IEEE working group.

Comments on standards should be submitted to the following address:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854 USA

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under U.S. and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate fee, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. A current IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every ten years. When a document is more than ten years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit IEEE

Xplore at http://ieeexplore.ieee.org/ or contact IEEE at the address listed previously. For more information about the IEEE-SA or IEEE's standards development process, visit the IEEE-SA Website at http://standards.ieee.org.

Errata

Errata, if any, for all IEEE standards can be accessed on the IEEE-SA Website at the following URL: http://standards.ieee.org/findstds/errata/index.html. Users are encouraged to check this URL for errata periodically.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE-SA Website at http://standards.ieee.org/about/sasb/patcom/patents.html. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Participants

At the time this draft standard was submitted to the IEEE-SA Standards Board for approval, the Point-of-Care Devices Working Group had the following membership:

John Rhoads, *Chair*
, *Vice Chair*

8

The following members of the individual balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

[To be supplied by IEEE]

When the IEEE-SA Standards Board approved this standard on <Date Approved>, it had the following membership:

<Name>, *Chair*
<Name>, *Vice Chair*
<Name>, *Past Chair*
Konstantinos Karachalios, *Secretary*

[To be supplied by IEEE]

Introduction

This introduction is not part of IEEE P11073-10201/D2.1.8, Draft Standard for Health informatics — Point-of-care medical device communication — Part 10201: Domain information model.

ISO/IEEE 11073 standards enable communication between different medical devices and between medical devices and other IT systems for information and for command and control. The primary goals are to:

- Provide real-time plug-and-play interoperability for patient-connected medical devices
- Facilitate the efficient exchange of patient related data and medical device related data, acquired at the point-of-care, in all health care environments

"Real-time" means that data from multiple devices can be retrieved, time correlated, and displayed or processed in fractions of a second.

"Plug-and-play" means that when a device or system is connected to another device or system, detection, configuration, and the initiation of communication all occur automatically andwithout any other human interaction.

"Efficient exchange of medical device data" means that information that is captured at the point-of-care (e.g., patient vital signs data) can be archived, retrieved, and processed by many different types of applications without extensive software and equipment support, and without needless loss of information. This standard is especially targeted at acute and continuing care devices, such as patient monitors, ventilators, infusion pumps, ECG devices, etc. It is a member of a family of standards that can be layered together to provide connectivity optimized for the specific devices being interfaced.

Contents

Draft Standard for Health informatics — Point-of-care medical device communication — Part 10201: Domain information model

1. Scope

The scope of this project is to define a general object-oriented information model that may be used to structure information and identify services used in point-of-care medical device communications. The scope is primarily focused on acute care medical devices and the communication of patient vital signs information.

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

.[1]

[1] IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://www.standards.ieee.org/).

ISO/IEC 8824-1, Information technology — Abstract Syntax Notation One (ASN.1) — Part 1: Specification of basic notation. [2]

ISO/IEEE 11073-10101, Health informatics — Point-of-care medical device communication — Part 10101: Nomenclature.

ISO/IEEE 11073-20101, Health informatics — Point-of-care medical device communication — Part 20101: Application profiles – Base standard.

OMG® Unified Modeling Language® (OMG UML®) 2.5.1[3]

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary: Glossary of Terms & Definitions* should be referenced for terms not defined in this clause.[4]

agent: Device that provides data in a manager-agent communicating system.

alarm: Signal that indicates abnormal events occurring to the patient or the device system.

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

alert condition: The active (true) state of a physiologic alarm (primarily related to the patient), technical alarm (primarily related to a device), or an advisory that is typically reported to clinicians, physicians, or other healthcare staff, for responding to patient needs or related workflows.

alert monitor: Object representing the output of an alert system that considers multiple alert conditions in a scope defined by objects that are contained by a single MDS object. An alert monitor is able to report individual, concurrent alert conditions as well as the overall system alert condition.

alert status: Object representing the output of an alert system that considers multiple alert conditions in a scope defined by objects that are contained by either a single VMD object or a single MDS object. An alert status is able to report concurrent alert conditions.

archival: Relating to the storage of data over a prolonged period.

[2] ISO/IEC documents can be obtained from the ISO office, 1 rue de Varembé, Case Postale 56, CH-1211, Genève 20, Switzerland/ Suisse (http://www.iso.ch/) and from the IEC office, 3 rue de Varembé, Case Postale 131, CH-1211, Genève 20, Switzerland/Suisse (http://www.iec.ch/). ISO/IEC publications are also available in the United States from the Sales Department, American National Standards Institute, 25 West 43rd Street, 4th Floor, New York, NY 10036, USA (http://www.ansi.org/).
[3] The OMG UML standard can be freely downloaded at https://www.omg.org/spec/UML/
[4] *The IEEE Standards Dictionary: Glossary of Terms & Definitions* is available at http://shop.ieee.org/.

association control service element (ACSE): Method used to establish logical connections between medical device systems (MDSs).

attribute: The definition of a property of an object.

channel: An object that groups together physiological measurement data and any derived data that have a contextual relationship with each other..

class: A model which describes the properties and behaviors of a type of entity found within a problem domain.

class diagram: Diagram showing connections between classes in a system.

communication controller: Part of a medical device system (MDS) responsible for communications.

communication party: Actor of the problem domain that participates in the communication in that domain.

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

data agent: As a medical device, a patient data acquisition system that provides the acquired data for other devices.

data format: Arrangement of data in a file or stream.

data logger: A device that is functioning in its capacity as a data storage and archival system.

NOTE—There may be several different types of data loggers; clinical, technical, forensic, alarm condition, user logs

data structure: A data organization format that is implemented by an application.

dictionary: Description of the contents of the medical data information base (MDIB) containing vital signs information, device information, demographics, and other elements of the MDIB.

discrete parameter: Measured, calculated, or manually entered value that can be expressed as a single numeric or textual value.

Example: a non-invasive systolic blood pressure (measured), cardiac index (calculated), gender male or female

domain information model (DIM): The model describing common concepts and relationships for a problem domain.

event: A change in device status that is communicated by a notification reporting service.

event report: Service [provided by the common medical device information service element (CMDISE)] to report an event relating to a managed object instance.

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

graphic parameter: Parameter that requires multiple regularly sampled data points in order to be expressed properly.

Example: a single ECG waveform snippet

host system: Term used as an abstraction of a medical system to which measurement devices are attached.**information service element**: Instances in the medical data information base (MDIB).

instance: The realization of an abstract concept or specification, e.g., class instance, application instance, information service element instance, virtual medical device (VMD) instance, operating instance.

instance method: A procedure or process that defines a behavior exhibited by the instances of a class (i.e. objects). Instance methods provide the interface by which the properties of an object may be accessed or modified.

intensive care unit (ICU): The unit within a medical facility in which patients are managed using multiple modes of monitoring and therapy.

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.

interoperability:The ability of two or more devices or systems to exchange information in a format that is usable by the receivers of the information.

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

lower layers: Layer 1 to Layer 4 of the International Organization for Standardization (ISO)/open systems interconnection (OSI) reference model. These layers cover mechanical, electrical, and general communication protocol specifications.

manager: Device that receives data in a manager-agent communicating system.

manager-agent model: Communication model where one device (i.e., agent) provides data and another device (i.e., manager) receives data.

medical data information base (MDIB): The concept of an object-oriented database storing (at least) vital signs information.

medical device: A device, apparatus, or system used for patient monitoring, treatment, or therapy, which does not normally enter metabolic pathways. For the purposes of this standard, the scope of medical devices is further limited to patient-connected medical devices that provide support for electronic communications.

medical device system (MDS): Abstraction for system comprising one or more medical functions. In the context of this standard, the term is specifically used as an object-oriented abstraction of a device that provides medical information in the form of instances of the classes that are defined in this standard.

monitor: A medical device designed to acquire, display, record, and/or analyze patient data and to alert caregivers of events needing their attention.

object: A particular instance of a class. An object represents a physical or logical occurrence of an actual medical device or medical device component.

object instance: synonym of object

object-oriented analysis: Method of analysis where the problem domain is modelled in the form of classes and the relationships between those classes.

open system: A set of protocols allowing computers of different origins to be linked together.

operation: A function or transformation that may be applied to or by objects (sometimes also called service).

problem domain: The field of health care under consideration in a modeling process.

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

real time: At the time when an event or process occurs.

scanner: An observer and "summarizer" of attribute values.

scenario: A formal description of a class of business activities including the semantics of business agreements, conventions, and information content.

service: A specific behavior that a communication party in a specific role is responsible for exhibiting.

syntax (i.e., of an interchange format): The rules for combining the construction elements of the interchange format.

system: The 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.

timestamp: An attribute or field in data that denotes the time of data generation.

top class: The ultimate base class for most of the classes belonging to the domain information model defined in this standard.

upper layers: Layer 5 to Layer 7 of the International Organization for Standardization (ISO)/open systems interconnection (OSI) reference model. These layers cover application, presentation, and session specifications and functionalities.

virtual medical device (VMD): An abstract representation of a medical-related subsystem of a medical device system (MDS).

virtual medical object (VMO): An abstract representation of an object in the Medical Package of the domain information model (DIM).

vital sign: Clinical information, relating to one or more patients, that is measured by or derived from apparatus connected to the patient or otherwise gathered from the patient.

waveform: Graphic data, typically vital signs data values varying with respect to time, usually presented to the clinician in a graphical form.

3.2 Abbreviations and acronyms

ACSE	association control service element
ASN.1	Abstract Syntax Notation One
BCC	bedside communication controller
BER	basic encoding rules
BMP	basic multilingual plane
CMDIP	Common Medical Device Information Protocol
CMDISE	common medical device information service element
CMIP	Common Management Information Protocol
CMISE	common management information service element
DCC	device communication controller
DICOM	digital imaging and communications in medicine
DIM	domain information model
ECG	electrocardiogram

1	EEG	electroencephalogram
2	EBWW	eyeball and wristwatch
3	FSM	finite state machine
4	GMDN	Global Medical Device Nomenclature
5	GMT	Greenwich mean time
6	IANA	Internet Assigned Numbers Authority
7	ICS	implementation conformance statement
8	ICU	intensive care unit
9	ID	identifier or identification
10	LAN	local area network
11	LSB	least significant bit
12	MDIB	medical data information base
13	MDS	medical device system
14	MEDICOM	medical imaging communication
15	MIB or Mib	management information base
16	MOC	managed object class
17	OID	object identifier
18	OR	operating room
19	OSI	open systems interconnection
20	PC	personal computer
21	PDU	protocol data unit
22	PM	persistent metric
23	SCADA	supervisory control and data acquisition
24	SCP ECG	Standard Communications Protocol [for computer-assisted]
25	Electrocardiography	
26	SNMP	Simple Network Management Protocol
27	SNTP	Simple Network Time Protocol
28	UDI	Unique Device Identifiaction
29	UML	unified modeling language
30	UTC	coordinated universal time
31	VMD	virtual medical device
32	VMO	virtual medical object
33	VMS	virtual medical system

4. General requirements

The ISO/IEEE 11073 family of standards is intended to enable medical devices to interconnect and inter-operate with other medical devices and with computerized healthcare information systems in a manner suitable for the clinical environment.

The ISO/IEEE 11073 family is based on an object-oriented systems management paradigm. Data (e.g., measurement, state) is modeled in the form of classes, instances of which can be accessed and manipulated using an object access service protocol.

The domain information model (DIM) defines the overall set of classes and their attributes, methods, and access functions needed for medical device communication.

The top-level user requirements are defined in IEEE 11073-20101,[5] which also defines the user scenarios covered by the ISO/IEEE 11073 family of standards.

As a part of the ISO/IEEE 11073 family of standards, the primary requirements for the DIM are as follows:

- Define an object-oriented model representing the relevant information (i.e., data) and functions (e.g., device controls) encountered in the problem domain of medical device communication, including measurement information, contextual data, device control methods, and other relevant aspects.
- Provide detailed specification of the classes defined in the object-oriented model, including their attributes and methods.
- Define a service model for communicating medical devices that provides access to the object instances, their attributes, and their methods.
- Use the nomenclature defined in ISO/IEEE 11073-10101 to identify all data elements in the model.
- Be usable for the definition of data communication protocols as well as for the definition of file storage formats.
- Define conformance requirements.
- Be extensible and expandable to incorporate future needs in the defined modeling framework.

[5] Information on references can be found in Clause 2.

5. DIM

5.1 General

5.1.1 Modeling concept

The DIM is an object-oriented model that consists of classifiers, their attributes, and their methods, which are abstractions of real-world entities in the domain of (vital signs information communicating) medical devices.

The information model and the service model for communicating systems defined and used in this standard are conceptually based on the International Organization for Standardization (ISO)/open systems interconnection (OSI) system management model (ISO/IEC 7498, parts 1-4). Classes defined in the information model are considered to model managed (here, medical) objects. For the most part, the objects are directly available to management (i.e., access) services provided by the common medical device information service element (CMDISE) as defined in this standard.

For communicating systems, the set of object instances available on any medical device that complies with the definitions of this standard forms the medical data information base (MDIB). The MDIB is a structured collection of managed medical objects representing the vital signs information provided by a particular medical device. Attribute data types, hierarchies, and behavior of objects in the MDIB are defined in this standard.

The majority of classes defined here represent generalized vital signs data and support information. Specialization of these classes is achieved by defining appropriate attributes. Class hierarchies and associations between classes are used to express device configuration and device capabilities.

> **Example**: A generalized object is instantiated to represent vital signs in the form of a real-time waveform. A set of attributes is used to specify a particular waveform as an invasive arterial blood pressure. The position in the hierarchy of all objects defines the subsystem that derives the waveform.

Figure 1 shows the relation between managed medical objects, MDIB, CMDISE, application processes, and communication systems.

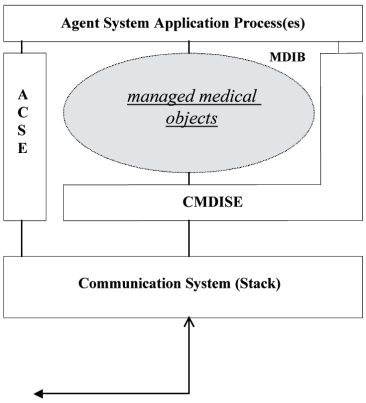

Figure 1 **—MDIB in communicating systems**

In the case of communicating systems, managed medical objects are accessible only through services provided by CMDISE. The way that these objects are stored in the MDIB in any specific system and the way applications and the CMDISE access these objects are implementation issues and as such not normative.

In the case of a vital signs archived data format that complies with the definitions in this standard, object instances are stored, together with their dynamic attribute value changes, over a certain time period on archival media. Attribute data types and hierarchies of objects in the archival data format are again defined in this standard.

Figure 2 shows the relationship between managed medical objects, the data archive, and archive access services.

In the case of the archived data format, the way managed medical objects are stored on a medium is the subject of standardization. The access services are a local implementation issue and as such are not intended to be governed by this standard.

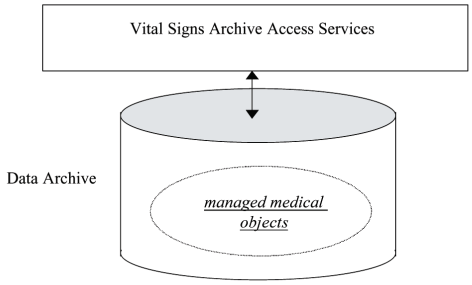

Figure 2 **—Managed medical objects in a vital signs archive**

5.1.2 Scope of the DIM

5.1.2.1 General

Vital signs information classes that are defined in this standard encompass digitized biosignals that are derived by medical measurement devices used, for example, in anaesthesia, surgery, infusion therapy, intensive care, and obstetrical care.

Biosignal data within the scope of this standard include direct and derived, quantitative and qualitative measurements, technical and medical alarms, and control settings. Patient information relevant for the interpretation of these signals is also defined in the DIM.

5.1.2.2 Communicating systems

Communicating systems within the scope of this standard include physiological meters and analysers, especially systems providing real-time or continuous monitoring. Data processing capabilities are required for these systems.

Information management objects that provide capabilities and concepts for cost-effective communication (specifically data summarization objects) and objects necessary to enable real-time communication are also within the scope of the information model in this standard.

Interoperability issues, specifically lower communication layers, temporal synchronization between multiple devices, etc., are outside the scope of this standard.

5.1.2.3 Archived vital signs

Context information classes that describe the data acquisition process and organize a vital signs archive are within the scope of this standard.

5.1.3 Approach

For the object-oriented modeling, the Unified Modeling Language (UML) is used. The domain is first subdivided into different packages, and this subdivision permits the organization of the model into smaller packages. Each package is then defined in the form of UML class diagrams. Classes are briefly introduced, and their relations and hierarchies are defined in the class diagrams. No single diagram shows all of the classes and relationships defined in the DIM. The diagrams must be considered as a set that, together, define the structure of the DIM.

For the class definitions, a textual approach is followed. Attributes are defined in attribute definition tables. Attribute data types are defined using Abstract Syntax Notation One (ASN.1) as defined by ISO/IEC 8824, parts 1-2. The behavior of and notifications generated by objects are also defined in definition tables. These definitions directly relate to the service model specified in Clause 7.

5.1.4 Extension of the model

It is expected that over time extensions of the model may be needed to account for new developments in the area of medical devices. Also, in special implementations, there may be a requirement to model data that are specific for a particular device or a particular application (and that are, therefore, not covered by the general model).

In some cases, it may be possible to use the concept of external object relations. Most classes defined in this standard provide an attribute group (e.g., the Relationship Attribute Group) that can be used to supply information about related objects that are not defined in the DIM. Supplying such information can be done by specifying a relation to an external object and assigning attributes to this relation (see6.1.3.20).

In other cases, it may be necessary to define completely new classes or to add new attributes, new methods, or new events to already defined classes. These extensions are considered private or manufacturer-specific extensions. Dealing with these extensions is primarily a matter of an interoperability standard that is based on this standard on vital signs representation.

In general, in an interoperability format, classes, attributes, and methods are identified by nomenclature codes. IEEE/ISO 11073 part 10101 defines nomenclature that should be preferred when implementing an 11073 compliant device. The nomenclature code space (i.e., code values) leaves room for private extensions. As a general rule, an interoperability standard that is based on this DIM should be able to deal with private or manufacturer-specific extensions by ignoring classes, attributes, etc., with unknown identifiers (i.e., nomenclature codes).

5.2 Package diagram–overview

The package diagram organizes the problem domain into separate groups. It shows key classes inside each package and the relationships between them.

The package diagram depicted in 0 contains only a small subset of all classifiers defined in the DIM. Not all relations are shown between the shown classes. Refer to the detailed UML class diagrams for each package for more information.

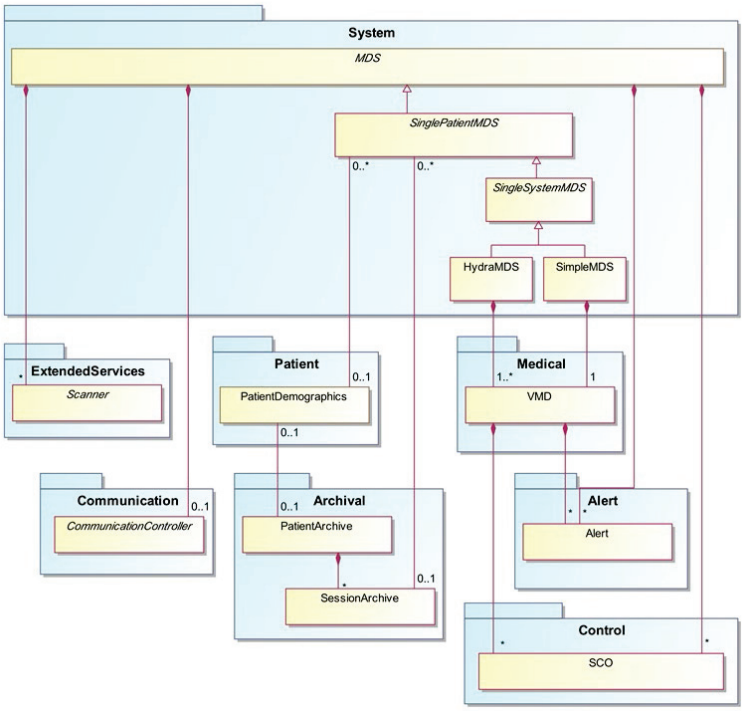

Figure 3 **—Packages of the DIM**

The Top class is an abstract class and at the same time the ultimate base for the majority of classes defined in the model. The only classes for which the Top class does not serve as a base class are the DeviceInterface class and the MibElement classes. Figure 4 shows the Top class and all of the class that are immediate subclasses of class Top. Many of the subclasses shown are themselves subclassed. This secondary subclassing is not shown in Figure 4.

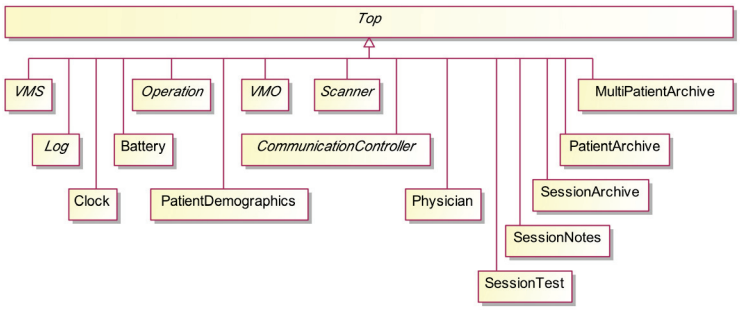

Figure 4 **—Top class inheritance**

The more detailed UML class diagrams for these packages are contained in 5.3 through 5.10. These diagrams show relationships between classes defined in the package as well as key relationships between classes defined in the package and classes defined in other packages. Some relationships appear in one diagram and are absent in other diagrams where the two associated classes are shown. The inclusion of a relationship in any UML class diagram indicates that it is part of the complete UML model.

The numbers in the packages refer to the corresponding subclauses in this clause about models and in Clause 6 about the class definitions.

5.3 Model for the Medical Package

5.3.1 General

The Medical Package deals with the derivation and representation of biosignals and contextual information that is important for the interpretation of measurements.

Figure 5 shows the UML class diagram for the Medical Package.

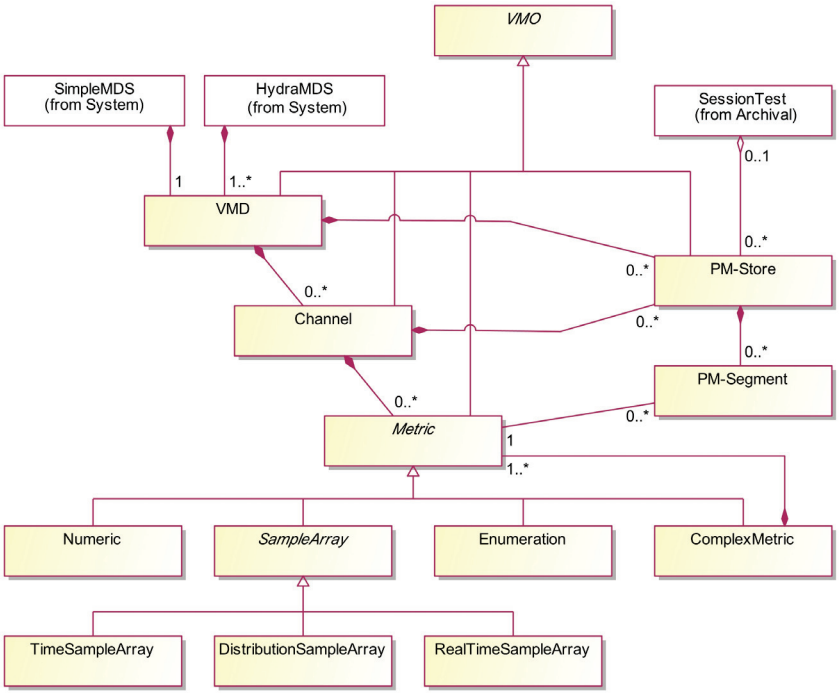

Figure 5 **—Medical Package model**

NOTE—Instances of any class that are modeled with a containment association may be contained by exactly one instance of a container class.[6]

The Medical Package model contains the classes described in 5.3.2 through 5.3.14.

5.3.2 VMO (i.e., virtual medical object) class

VMO is the abstract base class for all medical-related classes in the model. It provides consistent naming and identification across the Medical Package model.

The VMO class is abstract and cannot be instantiated.

5.3.3 VMD (i.e., virtual medical device) class

The VMD class models medical-related subsystems (e.g., hardware or even pure software) of medical devices. The characteristics of a subsystem (e.g., modes, versions) are captured

[6] Notes in text, tables, and figures are given for information only, and do not contain requirements needed to implement the standard.

by instances of this class. At the same time, a VMD object is a container for objects representing measurement and status information.

Example: A modular patient monitor provides measurement modalities in the form of plug-in modules. Each module is represented by a VMD object.

5.3.4 Channel class

Channel objects are used for grouping Metric objects that have a contextual relationship with each other.

Example ~~1~~: A blood pressure VMD may define a Channel object to group together all metrics that deal with the blood pressure (e.g., pressure value, pressure waveform). A second Channel object can be used to group together metrics that deal with heart rate.

~~**Example 2:** A pulse oximeter has two physical interface ports, one that provides waveform data and one that provides numeric values.~~

The Channel object is mandatory for representation of Metric objects in a VMD. In some cases a VMD may contain a single Channel and that Channel may be functionally superfluous. This may be likely if the attribution of the Channel object consists only of the mandatory *type* and *handle* attributes. In this case it is recommended that the value for the *type* attribute be initialized using the values corresponding to the term having the reference ID *MDC_DEV_CHAN*, which is defined in IEEE11073:10101. Usage of this term in this context may ~~serve as a signal~~provide an indication that the Channel object in question may not require representation in other messaging formats (e.g. IHE PCD-01).

5.3.5 Metric class

The Metric class is the abstract base class for all classes representing direct and derived, quantitative and qualitative biosignal measurement, status, and context data.

Specializations of the Metric class are provided to deal with common representations (e.g., single values, array data, status indications) and presentations (e.g., on a display) of measurement data.

The Metric class is abstract and cannot be instantiated.

5.3.6 Numeric class

The Numeric class models numerical measurements and status information, e.g., amplitude measures, counters.

Example: A heart rate measurement is represented by a Numeric object.

NOTE— A compound Numeric object is defined as an efficient model, for example, for arterial blood pressure, which usually has three associated values (i.e., systolic, diastolic, mean). The availability of multiple values in a single Numeric (or other Metric) object can be indicated in a special structure attribute of the Metric class.

5.3.7 SampleArray class

The SampleArray class is the abstract base class for metrics that have a graphical, curve type presentation and, therefore, have their observation values reported as arrays of data points by communicating systems.

The SampleArray class is abstract and cannot be instantiated.

5.3.8 RealTimeSampleArray class

The RealTimeSampleArray class models a real-time continuous waveform. As such, it has special requirements in communicating systems, e.g., processing power, low latency, high bandwidth.

Example: An electrocardiogram (ECG) real-time wave is represented as a RealTimeSampleArray object.

5.3.9 TimeSampleArray class

The TimeSampleArray class models noncontinuous waveforms (i.e., a wave snippet). Within a single observation (i.e., a single array of sample values), samples are equidistant in time.

Example: Software for ST segment analysis may use the TimeSampleArray class to represent snippets of ECG real-time waves that contain only a single QRS complex. Within this wave snippet, the software can locate the ST measurement points. It generates a new snippet, for example, every 15 seconds.

5.3.10 DistributionSampleArray class

The DistributionSampleArray class models linear value distributions in the form of arrays containing scaled sample values. Each value within an observation array represents a measured value from a different point in the parameter space. The DistributionSampleArray shall not be used for measurements distributed in time.

Example: An electroencephalogram (EEG) application may use a Fourier transformation to derive a frequency distribution (i.e., a spectrum) from the EEG signal. It then uses a DistributionSampleArray object to represent that spectrum in the MDIB.

5.3.11 Enumeration class

The Enumeration class models status information and/or annotation information. Observation values may be presented in the form of normative codes (that are included in the nomenclature defined in IEEE 11073 part 10101 or in some other nomenclature scheme), bit string, or in the form of free text.

Example: An ECG rhythm qualification may be represented as an Enumeration object. A ventilator may provide information about its current ventilation mode as an Enumeration object.

5.3.12 ComplexMetric class

In special cases, a ComplexMetric object can be used to group a larger number of strongly related Metric objects in one single container object for performance or for modeling convenience. A ComplexMetric object is a composition of Metric objects, possibly recursive.

> **Example**: A ventilator device may provide extensive breath analysis capabilities. For each breath, it calculates various numerical values (e.g., volumes, I:E ratio, timing information) as well as enumerated information (e.g., breath type classification, annotation data). For efficiency, all this information is grouped together in one ComplexMetric object instance, which is updated upon each breath.

5.3.13 PM-Store (i.e., persistent metric) class

The PM-Store class provides long-term storage capabilities for metric data. A PM-Store contains a variable number of PM-Segment objects that can be accessed only through the PM-Store object. A PM-Store object is intended to store data of a single Metric object only.

> **Example**: A device stores the numerical value of an invasive blood pressure on a disk. It uses the PM-Store object to represent this persistent information. The attributes of the PM-Store object describe the sampling period, the sampling algorithm, and the storage format. When the label of the pressure measurement is changed (e.g., during a wedge procedure), the storage process opens a new PM-Segment to store the updated context data (here: the label).

5.3.14 PM-Segment class

The PM-Segment class models a continuous time period in which a metric is stored without any changes of relevant metric context attributes (e.g., scales, labels).

PM-Segment objects are accessible only through the PM-Store object that contains them (e.g., for retrieving stored data, the PM-Store object has to be accessed).

5.4 Model for the Alert Package

5.4.1 General

The Alert Package deals with classes that represent status information about patient condition and/or technical conditions influencing the measurement or device functioning. Alert-related information is often subject to normative regulations to which a device may be required to comply and, therefore, requires special consideration that is outside the scope of this standard.

In the model, all alarm-related object-oriented items are identified by the term alert. The term alert is used in this standard as a synonym for the combination of patient-related physiological alarms, technical alarms, and equipment user-advisory signals.

An alarm is a signal that indicates abnormal events occurring to the patient or the device system. A physiological alarm is a signal that either indicates that a monitored physiological parameter is out of specified limits or indicates an abnormal patient condition. A technical alarm is a signal that indicates a device system is either not capable of accurately monitoring the patient's condition or no longer monitoring the patient's condition.

The model defines three different levels of alarming. These levels represent different sets of alarm processing steps, ranging from a simple context-free alarm event detection to an intelligent device system alarm process. This process facilitates the prioritization of all device alarms, the latching of alarms if needed (a latched alarm does not stop when the alert condition goes away), and the production of audible and visual alarm indications for the user.

For consistent system-wide alarming, a particular medical device may provide either no alarming capability or exactly one level of alarming, which is dependent on the capabilities of the device. Each level is represented by one specific object class. In other words, either zero or one alarm object class (e.g., only Alert or only AlertStatus or only AlertMonitor; no combinations) is instantiated in the device containment tree. Multiple instances of a class are allowed.

NOTE— Medical device alarming is subject to various national and international safety standards (e.g., IEC 60601 series, ISO 9703 series). Considering requirements of current safety standards, objects in this standard define information contents only. Any implementation shall, therefore, follow appropriate standards for dynamic alarming behavior.

Figure 6 shows the UML class diagram for the Alert Package.

Instances of classes in the Alert Package area shall be contained in exactly one superior object.

The Alert Package model contains the classes described in 5.4.2 through 5.4.4.

5.4.2 Alert class

The Alert class models the status of a simple alert condition check. As such, an instance represents a single alarm only. The alarm can be either a physiological alarm or a technical alarm condition of a related object [e.g., MDS (i.e., medical device system), VMD, Metric]. In the case that neither an AlertStatus nor AlertMonitor object is used, a single Alert object is needed for each alert condition that the device is able to detect.

Each Alert object has a reference to an object defined in the Medical Package to which the alert condition relates.

NOTE—An Alert object is not dynamically created or deleted in cases where alert conditions start or stop. Rather, an existing Alert object's attribute values change (are updated) in these cases.

Example: An Alert object may represent the status of a process that checks for a limit violation physiological alarm of the heart rate signal. In the case of a violation of the limit, the object generates an event (i.e., attribute update) that represents this alert condition in the form of attribute value changes.

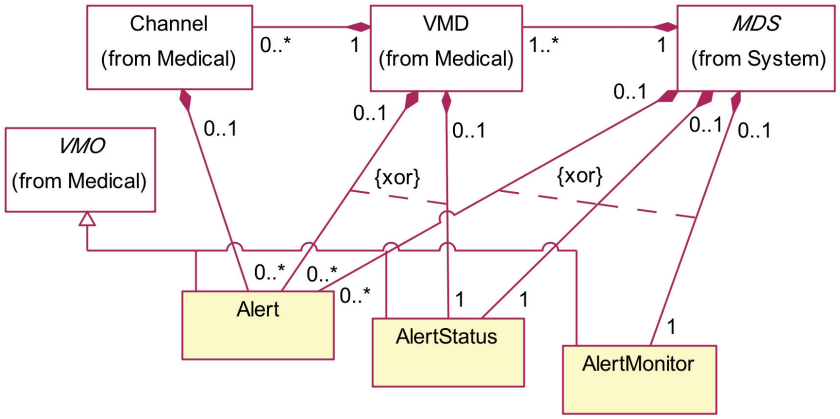

Figure 6 —**Alert Package model**

5.4.3 AlertStatus class

The AlertStatus class models the output of a process that considers all alert conditions in a scope that spans one or more objects. In contrast to an Alert object, an AlertStatus object collects all alert conditions related to a VMD object hierarchy or related to an MDS object and provides this information in list-structured attributes. Collecting all alarms together allows the implementation of first-level alarm processing where knowledge about the VMD or MDS can be used to prioritize alert conditions and to suppress known false alarm indications.

For larger scale devices without complete alarm processing, use of an AlertStatus object greatly reduces the overhead of a large number of Alert object instances.

If a device contains an AlertStatus object, it shall not contain any Alert or the AlertMonitor objects. Each VMD or MDS in the MDIB is able to contain at most one instance of the AlertStatus class.

> **Example**: An ECG VMD derives a heart rate value. As the VMD is able to detect that the ECG leads are disconnected from the patient, its AlertStatus object reports only a technical alarm and suppresses a heart rate limit violation alarm in this case.

5.4.4 AlertMonitor class

The AlertMonitor class models the output of a medical device system alert processor. As such, it represents the overall device or system alert condition and provides a list of all alert conditions of the system in its scope. This list includes global state information and individual alarm state information that allows the implementation of a safety-standard-compliant alarm display on a remote system.

If a device contains an AlertMonitor object, it shall not contain any Alert or AlertStatus objects. An MDS shall not contain more than one AlertMonitor object.

Example: A patient-monitoring system provides ~~alarm~~ alert information in the form of an AlertMonitor object to a central station. Alert information includes the current global maximum severity of audible and visual alert conditions on the monitor display as well as a list of active technical and physiological alarm conditions. The alert processor operates in a latching mode where physiological alarm conditions are buffered until they are explicitly acknowledged by a user.

5.5 Model for the System Package

5.5.1 General

The System Package deals with the representation of devices that derive or process vital signs information and comply with the definitions in this standard.

Figure 7 shows the UML class diagram for the System Package.

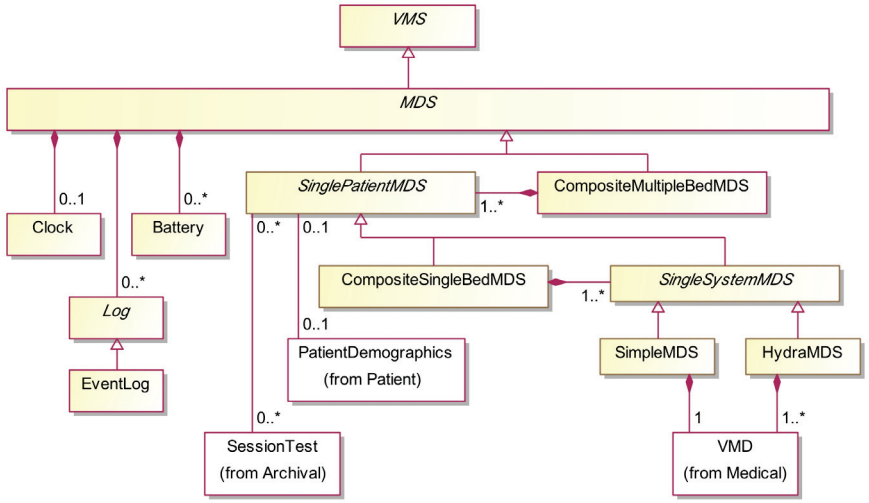

Figure 7 **—System Package model**

The System Package model contains the classes described in 5.5.2 through 5.5.13.

5.5.2 VMS (i.e., virtual medical system) class

The VMS class is the abstract base class for all System Package classes in this model. It provides consistent naming and identification of system-related objects.

The VMS class is abstract and cannot be instantiated.

5.5.3 MDS class

The MDS class is an abstraction of a device that provides medical information in the form of instances of classes that are defined in the Medical Package of the DIM.

An instance of the MDS class is the top-level object in the device's MDIB and represents the instrument itself. Composite devices may contain additional MDS objects in the MDIB.

Further specializations of this class are used to represent differences in complexity and scope.

The MDS class is abstract and cannot be instantiated.

5.5.4 CompositeMultipleBedMDS class

The CompositeMultipleBedMDS class models a device that contains multiple MDS objects at multiple locations (i.e., multiple beds).

5.5.5 SinglePatientMDS class

The SinglePatientMDS is an abstract class that models a medical device that is associated with a single patient.

The SinglePatientMDS class is abstract and cannot be instantiated.

5.5.6 CompositeSingleBedMDS class

The CompositeSingleBedMDS class models a device that contains one or more SimpleMDS or HydraMDS objects at one location (i.e., a bed).

5.5.7 SingleSystemMDS class

The SingleSystemMDS class models a medical device that contains VMD objects and that may be contained by a CompositeMDS object.

The SingleSystemMDS class is abstract and cannot be instantiated.

5.5.8 SimpleMDS class

The SimpleMDS class models a medical device that contains a single VMD instance only (i.e., a single-purpose device).

5.5.9 HydraMDS class

The HydraMDS class models a device that contains multiple VMD instances (i.e., a multipurpose device).

5.5.10 Log class

The Log class is an abstract base class that models a storage container for important local system notifications and events. It is possible to define specialized classes for specific event types.

The Log class is abstract and cannot be instantiated.

5.5.11 EventLog class

The EventLog class models a general Log object that stores system events in a free-text representation.

> **Example**: An infusion device may want to keep track of mode and rate changes by remote systems. When a remote operation is invoked, it creates an entry in its event log.

5.5.12 Battery class

For battery-powered devices, some battery information is contained in an MDS object in the form of attributes. If the battery subsystem is either capable of providing more information (i.e., a smart battery) or manageable, then a special Battery object is provided.

5.5.13 Clock class

The Clock class provides additional capabilities for handling date-related and time-related information beyond the basic capabilities of an MDS object. It models the real-time clock capabilities of an MDS object.

The Clock class is used in applications where precise time synchronization of medical devices is needed. This class provides resolution and accuracy information so that applications can synchronize real-time data streams between devices.

5.6 Model for the Control Package

5.6.1 General

The Control Package contains classes that allow remote measurement control and device control.

The model for remote control defined in this standard provides the following benefits:

- A system that allows remote control is able to explicitly register which attributes or features can be accessed or modified by a remote system.
- For attributes that can be remotely modified, a list of possible legal attribute values is provided to the controlling system.
- It is not mandatory that a remote-controllable item correspond to an attribute of a medical object.

— Dependence of a controllable item on internal system states is modeled.

— A simple locking transaction scheme allows the handling of transient states during remote control.

At least two different uses of remote control are considered:

— Automatic control may be done by some processes running on the controlling device. Such a process has to be able to discover automatically how it can modify or access the controllable items to provide its function.

— It is also possible to use remote control to present some form of control interface to a human operator. For this use, descriptions of functions, and possibly help information, need to be provided.

The basic concept presented here is based on Operation objects. An Operation object allows modification of a virtual attribute. This virtual attribute may, for example, be a measurement label, a filter state (on/off), or a gain factor. The attribute is called virtual because it need not correspond to any attribute in other objects instantiated in the system. A virtual attribute may correspond to multiple attributes and may be responsible for a series of actions.

Different specializations of the Operation class define how the virtual attribute is modified. A SelectItemOperation, for example, allows the selection of an item from a given list of possible item values for the attribute. A SetValueOperation allows the setting of the attribute to a value from a defined range with a specific step width (i.e., resolution).

The idea is that the Operation object provides all necessary information about legal attribute values. Furthermore, the Operation object defines various forms of text string to support a human user of the operation. It also contains grouping information that allows logical grouping of multiple Operation objects together when they are presented as part of a human interface.

Operation objects cannot directly be accessed by services defined in the service model in Clause 7. Instead, all controls shall be routed through the SCO (i.e., service and control object). Operation objects support a simple locking mechanism to prevent side effects caused by simultaneous calls.

The SCO groups together all Operation objects that belong to a specific entity (i.e., MDS, VMD). The SCO also allows feedback to a controlled device, for example, for a visual indication that the device is currently remote-controlled.

Figure 8 shows the UML class diagram for the Control Package:

The Control Package model contains the classes described in 5.6.2 through 5.6.10.

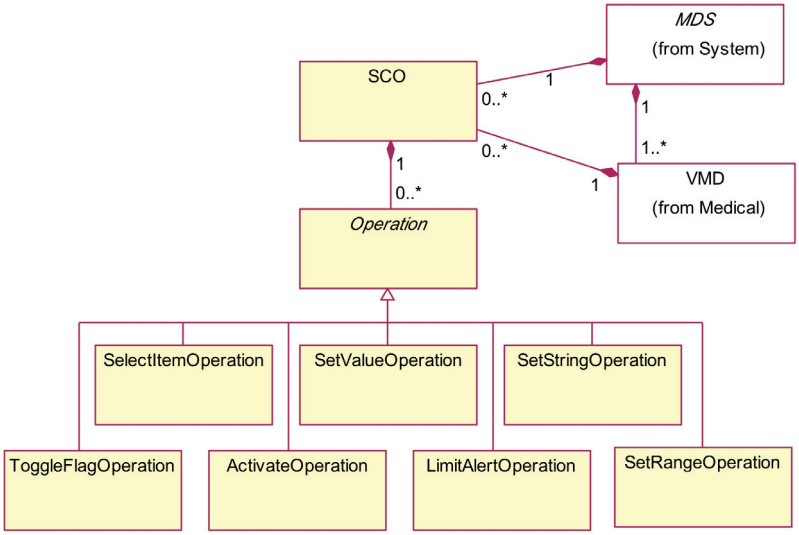

Figure 8 **—Control Package model**

5.6.2 SCO class

An instance of the SCO (service and control object) class is responsible for managing all remote-control capabilities that are supported by a medical device.

Remote control in medical device communication is sensitive to safety and security issues. The SCO provides means for the following:

1. Simple transaction processing, which prevents inconsistencies when a device is controlled from multiple access points (e.g., local and remote) and during the processing of control commands.
2. State indications, which allows local and remote indication of ongoing controls.

5.6.3 Operation class

The Operation class is the abstract base class for classes that model remote-controllable items. Each Operation object allows the system to modify some specific item (i.e., a virtual attribute) in a specific way defined by the Operation object. Operation objects are not directly accessible by services defined in the service model in Clause 7. All controls shall be routed through the SCO object (i.e., the parent) to allow a simple form of transaction processing.

The set of Operation objects instantiated by a particular medical device defines the complete remote control interface of the device. In this way a host system is able to discover the remote control capabilities of a device in the configuration phase.

The Operation class is abstract and cannot be instantiated.

5.6.4 SelectItemOperation class

An instance of the SelectItemOperation class facilitates the selection of one item out of a given list.

Example: The invasive pressure VMD may allow modification of its label. It uses a SelectItemOperation object for this function. The list of legal values supplied by the operation may be, for example, {ABP, PAP, CVP, LAP}. By invoking the operation, a user is able to select one value out of this list.

5.6.5 SetValueOperation class

An instance of the SetValueOperation class facilitates the adjustment of a value within a given range with a given resolution.

Example: A measurement VMD may allow adjustment of a signal gain factor. It uses a SetValueOperation object for this function. The SetValueOperation object provides the supported value range and step width within this range.

5.6.6 SetStringOperation class

An instance of the SetStringOperation class allows the system to set the contents of an opaque string variable of a given maximum length and format.

Example: An infusion device may allow a remote system to set the name of the infused drug in free-text form to show it on a local display. It defines an instance of the SetStringOperation class for this function. The SetStringOperation object specifies the maximum string length and the character format so that the device is able to show the drug name on a small display.

5.6.7 ToggleFlagOperation class

An instance of the ToggleFlagOperation class allows operation of a toggle switch (with two states, e.g., on/off).

Example: An ECG VMD may support a line frequency filter. It uses a ToggleFlagOperation object for switching the filter on or off.

5.6.8 ActivateOperation class

An instance of the ActivateOperation class allows a defined activity to be started (e.g., a zero pressure).

Example: The zero procedure of an invasive pressure VMD may be started with an ActivateOperation object.

5.6.9 LimitAlertOperation class

An instance of the LimitAlertOperation class allows adjustment of the limits of a limit alarm detector and the switching of the limit alarm to on or off.

5.6.10 SetRangeOperation class

An instance of the SetRangeOperation class allows the selection of a value range by the simultaneous adjustment of a low and high value within defined boundaries.

Example: A measurement VMD may provide an analog signal input for which the signal input range can be adjusted with a SetRangeOperation object.

5.7 Model for the ExtendedServices Package

5.7.1 General

The ExtendedServices Package contains classes that provide extended medical object management services that allow efficient access to medical information in communicating systems. Such access is achieved by a set of objects that package attribute data from multiple objects in a single event message.

The classes providing extended services are conceptually derived from ISO/OSI system management services defined in the ISO/IEC 10164 family of standards (specifically Part 5 and Part 13). The definitions have been adapted to and optimized for specific needs in the area of vital signs communication between medical devices.

Figure 9 shows the UML class diagram for the ExtendedServices Package.

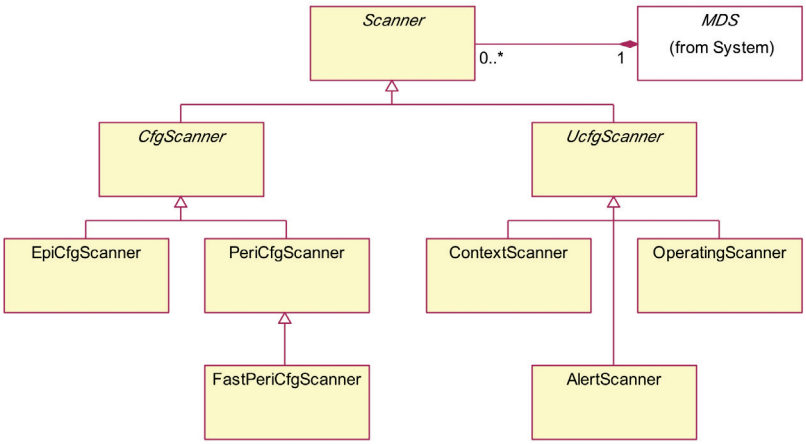

Figure 9 **—ExtendedServices Package model**

The ExtendedServices Package model contains the classes described in 5.7.2 through 5.7.10.

5.7.2 Scanner class

The Scanner class is an abstract base class that models an observer and "summarizer" of attribute values. A Scanner object observes attributes of managed medical objects and generates summaries in the form of notification event reports. These event reports contain data from multiple objects, which provide a better communication performance compared to separate polling commands (e.g., GET service) or multiple individual event reports from all object instances.

Subclasses of the Scanner class may be instantiated either by the agent system itself or by the manager system (e.g., dynamic scanner creation by using the CREATE service).

The Scanner class is abstract and cannot be instantiated.

5.7.3 CfgScanner (i.e., configurable scanner) class

The CfgScanner class is an abstract base class that defines a special attribute (i.e., the Scan-List attribute) that allows the system to configure ~~which of an object's~~ attributes of an object are scanned. The Scan-List attribute may be modified either by the agent system (i.e., auto-configuration or pre-configuration) or by the manager system (i.e., full dynamic configuration by using the SET service).

A CfgScanner object may support different granularity for scanning:

1. Attribute group (i.e., a defined set of attributes): The Scan-List attribute contains the identifiers (IDs) of attribute groups, and all attributes in the group are scanned.
2. Individual attribute: The Scan-List attribute contains the IDs of all attributes that are scanned.

In order to deal efficiently with optional attributes, the attribute group scan granularity is recommended for CfgScanner objects.

The CfgScanner class is abstract and cannot be instantiated.

5.7.4 EpiCfgScanner (i.e., episodic configurable scanner) class

An instance of the EpiCfgScanner class is responsible for observing attributes of managed medical objects and for reporting attribute changes in the form of unbuffered event reports.

The unbuffered event report is triggered only by attribute value changes. If an EpiCfgScanner object uses attribute group scan granularity, the event report contains all attributes of the scanned object that belong to this attribute group if one or more of these attributes changed their value.

Example: A medical device provides heart beat detect events in the form of an Enumeration object. A display application creates an instance of the EpiCfgScanner object and adds the observed value of the Enumeration object to the Scan-List attribute. The scanner instance afterwards sends a notification when the Enumeration object reports a heart beat.

5.7.5 PeriCfgScanner (i.e., periodic configurable scanner) class

An instance of the PeriCfgScanner class is responsible for observing attributes of managed medical objects and for periodically reporting attribute values in the form of buffered event reports. A buffered event report contains the attribute values of all available attributes that are specified in the scan list, independent of attribute value changes.

If the scanner operates in a special superpositive mode, the buffered event report contains all value changes of attributes that occurred in the reporting period; otherwise, the report contains only the most recent attribute values.

Example: A data logger creates an instance of the PeriCfgScanner class and configures the scanner so that it sends an update of the observed value attributes of all Numeric objects in the MDIB every 15 seconds.

5.7.6 FastPeriCfgScanner (i.e., fast periodic configurable scanner) class

The FastPeriCfgScanner class is specialized for scanning the observed value attribute of RealTimeSampleArray objects. This special scanner class is further optimized for low-latency reporting and efficient communication bandwidth utilization, which is required to access real-time waveform data.

Example: A real-time display application (e.g., manager system) wants to display ECG waveforms. It instantiates a FastPeriCfgScanner object on the agent system (e.g., server device) and requests periodic updates of all ECG leads.

5.7.7 UcfgScanner (i.e., unconfigurable scanner) class

The UcfgScanner class is an abstract base class. An UcfgScanner object scans a predefined set of managed medical objects that cannot be modified. In other words, an UcfgScanner object typically is a reporting object that is specialized for one specific purpose.

The UcfgScanner class is abstract and cannot be instantiated.

5.7.8 ContextScanner class

An instance of the ContextScanner class is responsible for observing device configuration changes. After instantiation, a ContextScanner object is responsible for announcing the object instances in the device's MDIB. A ContextScanner object provides the object instance containment hierarchy and static attribute values.

In case of dynamic configuration changes, a ContextScanner object sends notifications about new object instances or deleted object instances.

Example: A data logger creates a ContextScanner instance in an agent MDIB to receive notifications about MDS configuration changes when new measurement modules are plugged in (i.e., new VMD instance) or when such a module is unplugged (i.e., VMD instance deleted).

5.7.9 AlertScanner class

An instance of the AlertScanner class is responsible for observing the alert-related attribute groups of objects modeled in the Alert Package. As alarming in general is security-sensitive, AlertScanner objects are not configurable (i.e., all or no Alert objects are scanned).

An AlertScanner object sends event reports periodically so that timeout conditions can be checked.

5.7.10 OperatingScanner class

An instance of the OperatingScanner class is responsible for providing all information about the operating and control system of a medical device.

In other words, an OperatingScanner instance maintains the configuration of Operation objects contained in SCOs (by sending CREATE notifications for Operation objects), it scans transaction-handling-related SCO attributes, and it scans Operation attributes. Because SCOs and Operation objects may have dependencies, the OperatingScanner is not configurable.

5.8 Model for the Communication Package

5.8.1 General

The Communication Package deals with objects that enable and support basic communication.

Figure 10 shows the UML class diagram for the Communication Package.

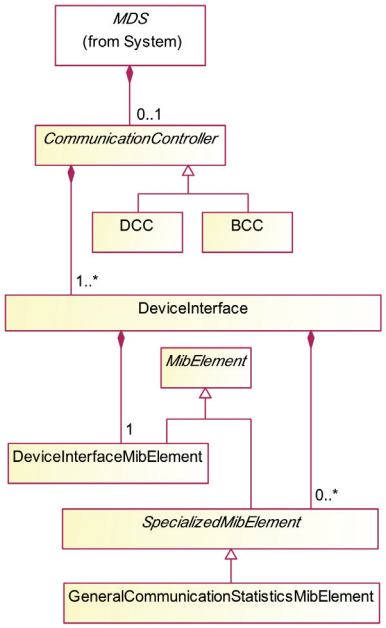

Figure 10—**Communication Package model**

The Communication Package model contains the classes described in 5.8.2 through 5.8.7.

5.8.2 CommunicationController class

The CommunicationController class models the upper layer and lower layer communication profile of a medical device.

CommunicationController objects are the access points for retrieving DeviceInterface attributes and management information base element (MibElement) attributes for obtaining management information related to data communications.

The CommunicationController class is abstract and cannot be instantiated. Two instantiable subclasses are defined: BCC and DCC. A medical device MDIB contains either

no CommunicationController object or one BCC object or one DCC object (depending on its role).

5.8.3 DCC (i.e., device communication controller) class

The DCC class is the CommunicationController subclass used by medical devices operating as agent systems (i.e., association responders).

DCC objects shall contain one or more DeviceInterface objects.

5.8.4 BCC (i.e., bedside communication controller) class

The BCC class is the CommunicationController subclass used by medical devices operating as manager systems (i.e., association requestors).

BCC objects shall contain one or more DeviceInterface objects.

5.8.5 DeviceInterface class

The DeviceInterface class models a particular interface, i.e., port. The port is either a logical or a physical end point of an association for which (e.g., statistical) data captured in the MibElement objects can be independently collected.

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

DeviceInterface objects are not accessible by CMDISE services. DeviceInterface objects contain at least one Mib-Element object (i.e., the DeviceInterfaceMibElement object, which represents device interface properties), which can be accessed by a special method defined by the CommunicationController object.

5.8.6 MibElement class

An instance of the MibElement class contains statistics and performance data for one DeviceInterface object.

The MibElement class is abstract and cannot be instantiated

Management information for a communication link is dependent on the lower layers of the communication stack (i.e., lower layers profile). Various MibElement subclasses are defined to group management information in defined packages, which can be generic or dependent on specific transport profiles. This standard defines two concrete subclasses of MibElement. Additional MibElement specializations may be defined in the future by other standards.

MibElement objects are not directly accessible. Their attributes can be accessed only through a CommunicationController object. MibElement objects are not part of the device's MDIB.

5.8.7 DeviceInterfaceMibElement class

An instance of the DeviceInterfaceMibElement class describes the properties of the device interface. If a Medical Device System (MDS) implements a CommunicationController then a DeviceInterfaceMibElement must be implemented as well.

5.8.8 SpecializedMibElement class

The SpecializedMibElement class is the abstract base class for all classes defined for managing information beyond that which is handled by the DeviceInterfaceMibElement. Only one subclass of SpecializedMibElement is defined in this standard. More specializations may be defined in the future by other standards.

5.8.9 GeneralCommunicationStatisticsMibElement class

The GeneralCommunicationStatisticsMibElement class is a subclass of SpecializedMibElement. The GeneralCommunicationStatistics class models typical communication statistics that are generally applicable.

~~#### 5.8.7 Specialized MibElement classes~~

~~Management information for a communication link is dependent on the lower layers of the communication stack (i.e., lower layers profile).~~

~~This standard, however, defines only two MibElement subclasses:~~

- ~~The mandatory DeviceInterfaceMibElement class, which describes properties of the device interface~~
- ~~An optional GeneralCommunicationStatisticsMibElement class, which models typical communication statistics that are generally applicable~~

5.9 Model for the Archival Package

5.9.1 General

The Archival Package deals with storage and representation of biosignals, status, and context information in an on-line or an off-line archive.

Figure 11 shows the UML class diagram for the Archival Package.

The Archival Package model contains the classes described in 5.9.2 through 5.9.8.

Formatted: IEEEStds Paragraph

Formatted: IEEEStds Paragraph

Formatted: IEEEStds Paragraph

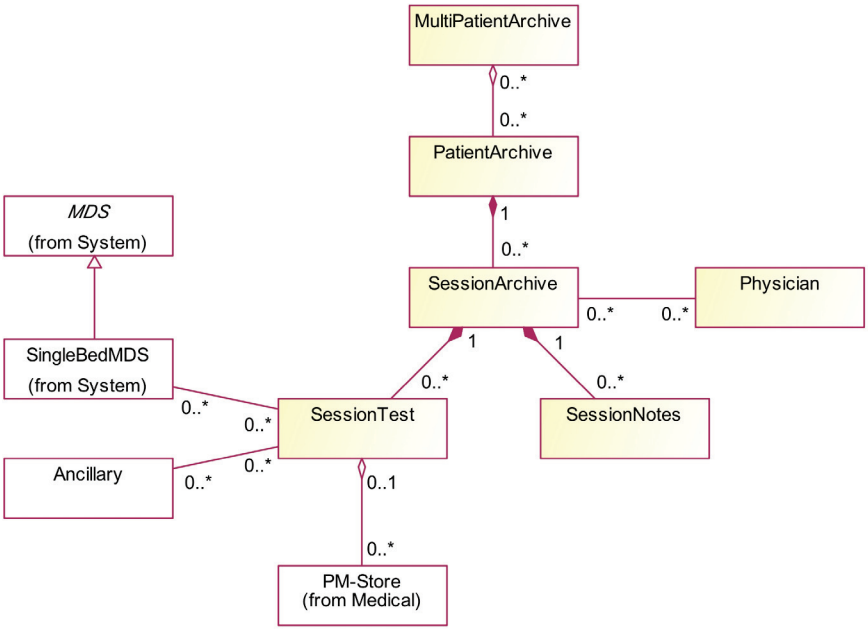

Figure 11—**Archival Package model**

5.9.2 MultipatientArchive class

Instances of the MultipatientArchive class group together multiple PatientArchive objects referring to different patients.

Example: A drug study may be documented in the form of a MultipatientArchive object containing multiple PatientArchive objects that show how the drug affected the monitored vital signs.

5.9.3 PatientArchive class

Instances of the PatientArchive class group patient-related information (e.g., vital signs data, treatment data, and patient demographics) together in a single archive object. This object relates to static (i.e., invariant) data in a PatientDemographics object only.

Example: A hospital may store data about multiple visits of a single patient in a PatientArchive object that contains a number of SessionArchive objects, each documenting vital signs information recorded during a specific visit in a hospital department.

5.9.4 SessionArchive class

The SessionArchive class models a patient visit or a continuous stay in ~~the~~ a hospital or hospital department. Diagnostic treatments performed during this time period are represented by SessionTest objects contained in a SessionArchive object. A SessionArchive object refers to dynamic (i.e., variant) data in a PatientDemographics object.

5.9.5 PhysicianArchive class

The Physician class models the physician responsible for the set of diagnostic and therapeutic activities during the time period represented by a SessionArchive object.

5.9.6 SessionTest class

The SessionTest class models vital signs information of a single patient that is recorded during a single examination or diagnostic treatment. Instances of the SessionTest class contain vital signs metrics in form of PM-Store objects. It also may contain information about equipment that was used for recording (in the form of relations to MDS and Ancillary objects).

Example: Vital signs information recorded during a ECG stress test examination is organized in a SessionTest object.

5.9.7 SessionNotes class

The SessionNotes class models a container for diagnostic data, patient care details, and treatment-related information in the form of textual data.

5.9.8 Ancillary class

The Ancillary class is not further defined in this standard. This class is present in the model to indicate that information from sources other than devices within the scope of this standard are permitted to be included in (or referenced by) the SessionTest object.

Example: Image data that complies with the DICOM (ISO 12052) standard are permitted to be included in the SessionTest object as ancillary data.

5.10 Model for the Patient Package

5.10.1 General

The Patient Package deals with all patient-related information that is relevant in the scope of this standard, but is not vital signs information modeled in the Medical Package.

Figure 12 shows the UML class diagram for the Patient Package:

The Patient Package model contains one class (see 5.10.2).

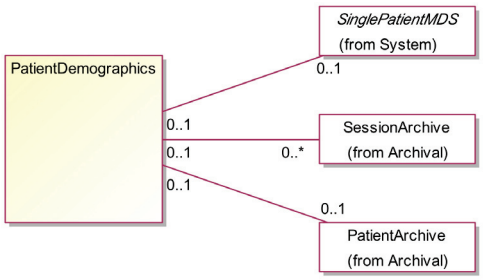

Figure 12—**Patient Package model**

5.10.2 PatientDemographics class

Instances of the PatientDemographics class store patient census data.

This standard provides minimal patient information as typically required by medical devices. A complete patient record is outside the scope of this standard.

5.11 DIM—dynamic model

5.11.1 General

Subclause 5.11 defines global dynamic system behavior.

Note that dynamic object behavior resulting from invocation of object management services is part of the class definitions in Clause 6.

5.11.2 MDS communication finite state machine (FSM)

Figure 13 shows the MDS FSM for a communicating medical device that complies with the definitions in this standard. The FSM is used to synchronize the operational behavior of manager (i.e., client) systems and agent (i.e., server) systems.

After power-up, the device performs all necessary local initializations (i.e, boot phase) and ends up in the disconnected state, where it waits for connection events.

When a connection event is detected, the device tries to establish a logical connection (i.e, an association) with the other device. A manager (i.e., client) system is the association requester, and an agent (i.e, server) system is the association responder. Basic compatibility checks are performed in the associating state.

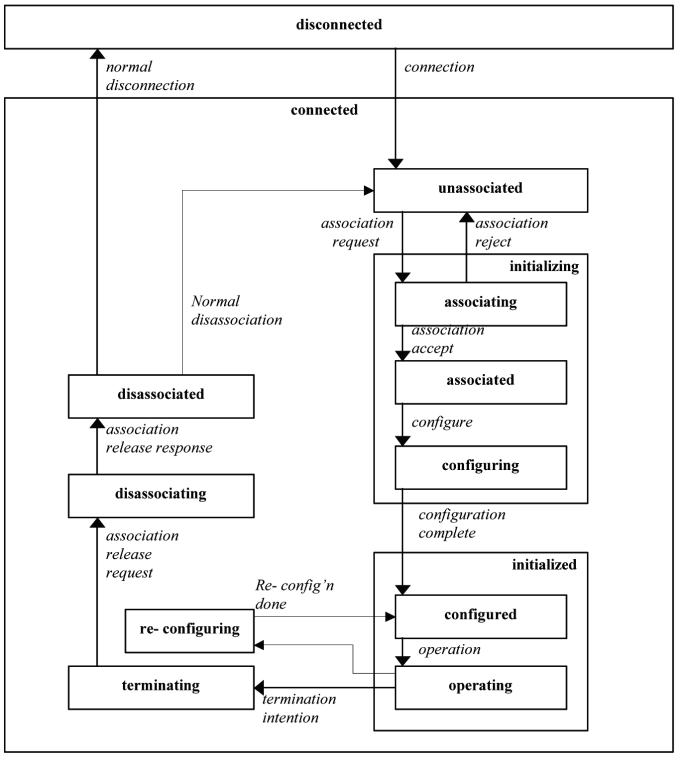

Figure 13—**MDS FSM**

After successful association, configuration data (i.e., the MDIB structure) is exchanged by the use of services and extended services (in particular, the ContextScanner object) as defined in this standard. Additional information (e.g., MDS attributes) is supplied that allows further compatibility and state checks.

After configuration, medical data are exchanged by using services and extended services as defined in this standard. Dynamic reconfiguration is allowed in the operating state. If the device or the type of reconfiguration does not allow dynamic handling in the operating state, a special “reconfiguring” state is provided.

If an event indicates an intention to disconnect, the disassociating state is entered.

The diagram does not show error events. Fatal error events take the state machine out of the operating state.

NOTE—This state machine describes the behavior of the MDS communication system only. Usually the device must perform its medical function independent of the communication system.

The FSM is considered a part of the MDS class. The MDS-Status attribute reflects the state of the machine. The MDS may announce state changes in the form of attribute change event reports.

Specific application profiles shall use this state machine as a general guideline, but they may define specific deviations to fulfill specific profile-dependent requirements or assumptions.

5.11.3 Communicating systems—startup object interaction diagram

Figure 14 presents the UML sequence diagram that visualizes the startup phase after connecting two devices.

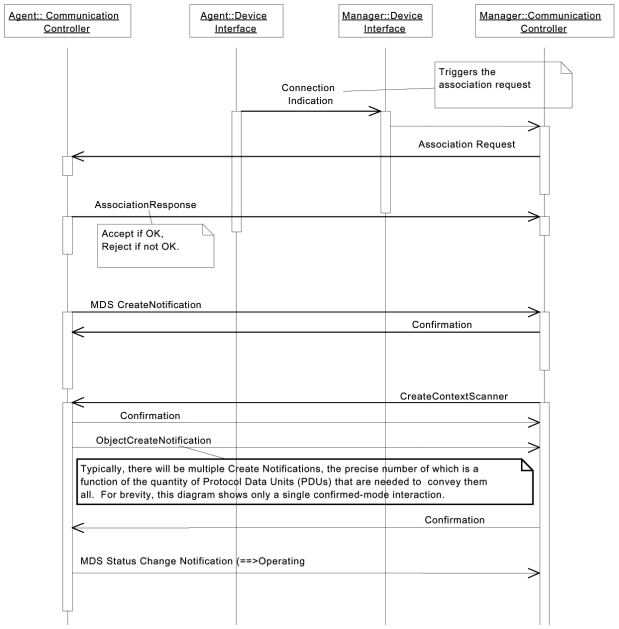

Figure 14—**Startup after connection**

It is assumed here that, conceptually, messages are exchanged between the CommunicationController 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; therefore, it is not further defined in this standard.

Specific application profiles shall use this interaction diagram as a general guideline, but they may define specific deviations to fulfill specific profile-dependent requirements or assumptions.

5.11.4 Communication Package—MibElement data access

Figure 15 presents the UML sequence diagram that shows how a manager system accesses the MibElement data using the objects defined in the Communication Package (see 5.8).

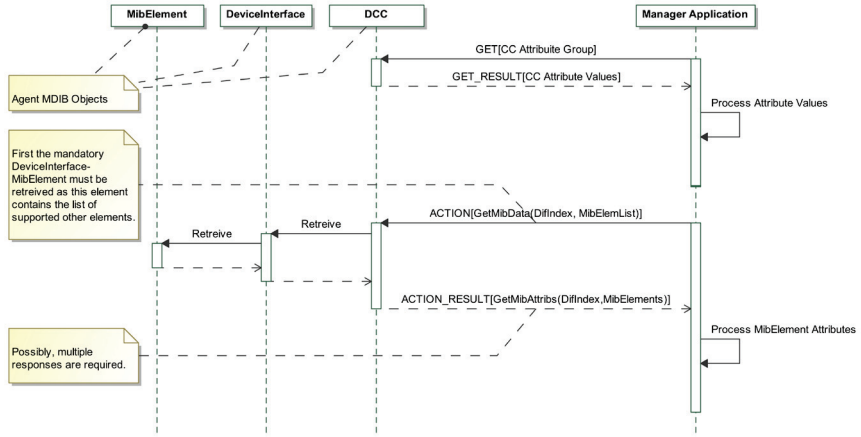

Figure 15—**MibElement data access**

The diagram assumes the following:

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

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

The manager uses the ACTION service with the CommunicationController's Get-Mib-Data method to retrieve the attributes of the mandatory DeviceInterfaceMibElement object. The MibElement attribute variables specify if any additional MibElement objects are available for the interface. If so, the manager can use the same ACTION command to retrieve the additional management information represented in the MibElement objects.

5.11.5 Dynamic object relations

5.11.5.1 General

This subclause deals with relations between managed medical objects (i.e., instances of classes that are defined as managed objects in this standard).

Generally, the relationships between objects that are defined in the package models are dynamic.

> **Example**: A modular patient monitor is modeled as an MDS. Measurement modules are modeled as VMDs. If a new module is connected to the monitor, there is also a new relationship between the MDS and the new VMD instance.

Communicating agent systems (i.e., agents) use services defined in this standard to announce configuration change events to other connected systems. These manager systems (i.e., managers) modify their view of the agent MDIB.

Not only does a vital signs information archive have to update its configuration, but it also has to permanently store these connection and disconnection events.

> **Example**: An instance of the SessionArchive class represents the stay of a patient in the intensive care unit (ICU). During that period, new devices are connected to the patient to increase the number of recorded vital signs. They are removed again as soon as the patient's condition stabilizes. The SessionArchive object shall not delete recorded data when the recording device is disconnected.

Thus, in certain applications (e.g., archival applications), object relationships have associated information that must be captured.

When required, the relationships themselves can be considered to be special managed objects as shown in Figure 16.

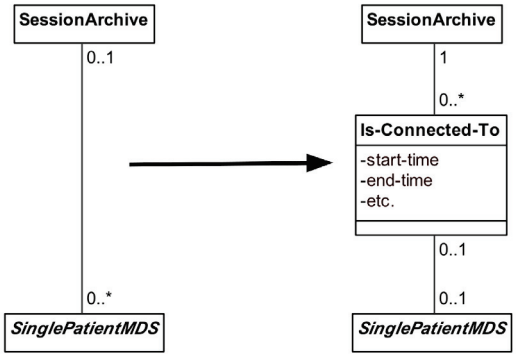

Figure 16—**Example of a relationship represented by an object**

The example in Figure 16 shows a relation between a SessionArchive object and an MDS object. The relation is represented as an object. This object has attributes that provide information, for example, about time of connection and disconnection.

Modeling the relations as objects has the advantage that information can be defined in the form of attributes. It is not necessary to assign the attributes to one or both objects that are related.

How dynamic object relations are handled by systems that comply with the definitions in this standard is defined in 5.11.5.2 and 5.11.5.3.

5.11.5.2 Dynamic object relations in communicating systems

Relations between objects that are defined in the package models are considered configuration information. The ContextScanner class provides configuration information in the form of object create notifications and object delete notifications. No means for persistent storage of past (i.e., old) configuration information is defined for communicating systems.

Other relations between objects (e.g., to reference a source signal in derived data) are specified in the form of attributes of the corresponding objects (e.g., the Vmo-Source-List attribute of the Metric object). Dynamic changes of these attributes are announced by attribute change notification events.

5.11.5.3 Dynamic object relations in archival systems

An archival system may need to provide persistent storage for configuration information. In this case, the corresponding relations are considered to be represented by objects, as shown in 5.11.5.

An archival system that uses a data format in compliance with the definitions in this standard has to provide means to store (i.e., archive) the attributes of dynamic relationship objects.

6. DIM class definitions

6.1 Overview

6.1.1 General

Clause 6 contains the definitions for all classes in the DIM. The packages defined in the model are used to categorize classes. Attributes, behavior, and notifications are defined for each class.

6.1.2 Notation

Each class is defined in a separate subclause (see 6.2 through 6.10). Further subclauses define attributes, behavior, and notifications for the classes.

The class is defined in a subclause as follows:

Class:	Defines the name of the class.
Description:	"Gives a short, informative textual description of the class."
Superclass:	Defines the class from which this class inherits attributes, attribute groups, behavior, and notifications.
Subclasses:	Defines classes that inherit attributes, attribute groups, behavior, and notifications from this class.
Name Binding:	Defines the attribute that uniquely identifies an object instance in a given context. For manageable objects, this definition is the Handle attribute and the context is the device system (i.e., single MDS context). See also 6.1.3.5.
Registered As:	Defines a term that is defined in the nomenclature to allow unique identification [e.g., object identifier (OID), code] of the class.

The attributes of each class are defined in an attribute subclause. Tables define attribute names, unique attribute IDs, attribute data types, and certain qualifiers. The qualifiers have the following meaning:

M	attribute is mandatory
O	attribute is optional
C	attribute is conditional; availability of attribute depends on a predefined condition

Unless otherwise noted, the attribute definition tables do not show inherited attributes again. In other words, the attribute lists of all superclasses have to be checked for a complete list of attributes.

Attributes are assigned to, or grouped together in, attribute groups so attributes can be classified according to their use (e.g., static context information, dynamic context information, value observations). The grouping also makes it possible to effectively deal with optional attributes: A GET service facilitates the retrieval of all members of the group so an application is able to determine which attributes are actually present in a particular object instance.

Attribute groups may be extensible. In other words, a derived class is able to add additional members to an inherited attribute group.

Attribute groups are also defined in tables that specify group identification and the list of group members. Inherited attribute groups are not shown in the attribute group tables again unless these groups are extensible.

Special instance methods or functions that may be called on an object are defined in a behavior subclause. These methods can be invoked by the CMDISE ACTION service.

Events generated by an object (other than a generic attribute change notification) are defined in a notifications subclause. An object reports these events by using the CMDISE EVENT REPORT service.

6.1.3 Common data types

This subclause defines a set of ASN.1 data types that are used in the class definitions.

The external nomenclature reference data type is a special data type that is defined for this function as follows:

6.1.3.1 Integer and bit string data types

For representing integer numbers, the class definitions use fixed-size data types only. The bit string data type represents a bit field where each single bit has a defined meaning (i.e., flag fields). The following integer data types and bit string data types are used:

```
--
-- 8-bit unsigned integer
--
INT-U8 ::= INTEGER (0..255)

--
-- 8-bit signed integer
--
INT-I8 ::= INTEGER (-128..127)

--
-- 16-bit unsigned integer
--
INT-U16 ::= INTEGER (0..65535)

--
-- 16-bit signed integer
--
INT-I16 ::= INTEGER (-32768..32767)

--
-- 32-bit unsigned integer
--
INT-U32 ::= INTEGER (0..4294967295)

--
-- 32-bit signed integer
--
INT-I32 ::= INTEGER (-2147483648..2147483647)

--
-- 8-bit bit string
--
BITS-8 ::= BIT STRING (SIZE(8))

--
-- 16-bit bit string
--
BITS-16 ::= BIT STRING (SIZE(16))
```

```
--
-- 32-bit bit string
--
BITS-32 ::= BIT STRING (SIZE(32))

--
-- 8-bit octet string
--
OCTET STRING-8 ::= OCTET STRING (SIZE(8))
```

NOTE 1—When interpreting integer numbers, the representation (e.g., little endian versus big endian) has to be considered. Communicating systems negotiate this representation at association (i.e., transfer syntax). Archival data formats have to provide a mechanism to uniquely identify integer representation (e.g., a field in a specification header).

NOTE 2—In the class definitions, data types with named constants or named bits also use the above notation for simplicity. The above notation is illegal ASN.1 syntax, but it can be easily transformed to the correct syntax.

6.1.3.2 Identification data type

All elements (e.g., classes, objects, measurement types) that need unique identification are assigned an OID. The set of valid OIDs for this standard is defined in ISO/IEEE 11073-10101. The nomenclature is split into a set of partitions, and each partition has its own range of 16-bit codes. In other words, the 16-bit code is context-sensitive.

The 16-bit identification data type is defined as follows:

```
--
-- OID type as defined in nomenclature (do not confuse with ASN.1 OID)
-- 16-bit integer type
--
OID-Type ::= INT-U16
```

For IDs that are not part of the standard nomenclature (i.e., private or manufacturer-specific codes), a special type is defined as follows:

```
--
-- Private OID
--
PrivateOid ::= INT-U16
```

6.1.3.3 Handle data type

The handle data type is used for efficient, locally unique identification of all managed medical object instances. (*Locally unique* means unique within one MDS context.) This data type is defined as follows:

```
--
-- handle
--
HANDLE ::= INT-U16
```

6.1.3.4 Instance number data type

The instance number data type is used to distinguish object instances of the same type or object instances that are not directly manageable (i.e., used, e.g., as the Name Binding attribute for Operation objects). This data type is defined as follows:

```
--
-- Instance Number
--
InstNumber ::= INT-U16
```

6.1.3.5 Global object identification

Handle and instance number data types must be unique inside one specific naming context (e.g., handles are unique within at least one MDS context). This uniqueness allows the identification of an object instance within its naming context by one single, small identifier.

To address larger scale systems, a context ID field at the MDS level within an MDIB is added to the handle data type so that multiple device systems can be distinguished. This global handle data type is defined as follows:

```
--
-- MDS Context ID
--
MdsContext ::= INT-U16

--
-- Global handle allows identification of an object in a larger scale system
--
GLB-HANDLE ::= SEQUENCE {
      context-id   MdsContext,
      handle       HANDLE
}

--
-- Managed OID as a type for complete global object identification
--
ManagedObjectId ::= SEQUENCE {
      m-obj-class OID-Type,
      m-obj-inst   GLB-HANDLE
}
```

Example: A medical device may interface with further medical devices (i.e., sub-devices). In the MDIB, this device may model these sub-devices as individual MDS objects with their own naming context. In this way, name space collisions (e.g., duplicate handle values, duplicate nomenclature codes) can be avoided without reassigning handle values. A manager system needs to interpret the MDS context IDs together with handle values to uniquely identify object instances within this composite MDIB. The context IDs are assigned when the MDIB is created by ContextScanner object create notifications.

Assumptions and possible restrictions about different naming contexts within an MDIB are profile dependent.

6.1.3.6 Type ID data type

The type ID data type is used in the VMOs and VMS objects to provide specific static information about the type of an object instance (e.g., blood pressure could be the type of a Numeric object). Codes defined in the nomenclature are used. The nomenclature contains a number of partitions, and code values are unique only within one partition. As the type

ID data type should be context-free, the partition of the nomenclature code is also provided. This data type is defined as follows:

```
--
-- Type ID
--
TYPE ::= SEQUENCE {
      partition     NomPartition,
      code          OID-Type
}

--
-- The following nomenclature partitions exist
--
NomPartition ::= INT-U16 {
      nom-part-unspec(0),
      nom-part-obj(1),              -- object-oriented partition
      nom-part-metric(2),           -- metric [supervisory control and data acquisition (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-infrastruct(8),      -- infrastructure elements partition
      nom-part-fef(9),              -- file exchange format partition
      nom-part-ecg-extn(10),        -- ECG extensions partition
      nom-part-icdo(11),            -- IDCO partiion
      nom-part-phddm(128),          -- phd disease management partition
      nom-part-hf(129),             -- health and fitness partition
      nom-part-ageind(130),         -- aging independently partition
      nom-part-returncodes(255),    -- return codes patition
      nom-part-ext-nom(256),        -- IDs of other nomenclatures and dictionaries
      nom-part-settings(258),       -- settings partition
      nom-part-priv(1024)           -- private partition
}
```

6.1.3.7 Attribute value assertion data type

A number of services defined in the service model in Clause 7 provide access to theattributes of an object (e.g., GET, SET). Typically, the attribute has to be identified by means of an attribute ID. The attribute data type itself is dependent on this ID. The attribute value assertion data type represents this ID-value pair and is defined as follows:

```
AVA-Type ::= SEQUENCE {
      attribute-id                  OID-Type,
      attribute-value               ANY DEFINED BY  attribute-id
}
```

6.1.3.8 Attribute list data type

Frequently, a list of attribute ID–attribute value pairs is needed. The attribute list data type is a special data type that is provided for this situation and is defined as follows:

```
AttributeList ::= SEQUENCE OF AVA-Type
```

6.1.3.9 Attribute ID list data type

Frequently, a list of attribute IDs is used. The attribute ID list data type is a special type that is provided for convenience and is defined as follows:

```
AttributeIdList ::= SEQUENCE OF OID-Type
```

6.1.3.10 Floating point type data type

For performance and efficiency, the class definitions use the floating point type data type, which is a special data type for representing floating point numbers. It is assumed that this data type is 32 bits. This data type is defined as follows:

```
--
-- 32-bit float type; the integer type is a placeholder only
--
FLOAT-Type ::= INT-U32
```

The ~~concrete~~ specific floating point number format is either explicitly negotiated at association or implicitly defined by the association application context.

6.1.3.11 Relative time data type

The relative time data type is a high-resolution time definition relative to some event (e.g., a synchronization event at startup). This data type is used to position events relative to each other. It is defined as follows:

```
--
-- Relative time has a resolution of 125 µs [least significant bit (LSB)], which is sufficient for sampling rates up to
-- 8 kHz and spans time periods up to 6.2 days
--
RelativeTime ::= INT-U32
```

Note that the time accuracy is defined by the system itself.

6.1.3.12 High-resolution relative time data type

If either the resolution or the time span of the previously defined relative time data type is not sufficient, a high-resolution relative time data type is defined. The data type is 64 bits long. However, as there is no 64- bit integer data type defined, an opaque (i.e., string) data structure is used. The type is defined as follows:

```
--
-- 64-bit (8 byte) high-resolution time, the LSB represents 1 µs
--
HighResRelativeTime ::= OCTET STRING-8
```

Note that the time accuracy is defined by the system itself.

6.1.3.13 Absolute time data type

Absolute time data type specifies the time of day with at least a resolution of 1 s. For efficiency, the values in the structure are BCD-encoded (i.e., 4-bit nibbles). The year 1996,

for example, is represented by the hexa- decimal value 0x19 in the century field and the hexadecimal value 0x96 in the year field. This format can easily be converted to character-based or integer-based representations. The absolute time data type is defined as follows:

```
AbsoluteTime ::= SEQUENCE {
    century                 INT-U8,
    year                    INT-U8,
    month                   INT-U8,
    day                     INT-U8,
    hour                    INT-U8,
    minute                  INT-U8,
    second                  INT-U8,
    sec-fractions           INT-U8          -- hundredths of second if available
}
```

6.1.3.14 Date data type

The date data type is used to specify a certain calendar date. For ease of transformation, the data type has the same encoding (i.e., BCD) as the absolute time data type. The date data type is defined as follows:

```
Date ::= SEQUENCE {
    century     INT-U8,
    year        INT-U8,
    month       INT-U8,
    day         INT-U8
}
```

6.1.3.15 Data Type: OperationalState

The operational state data type defines if a certain object or other property is enabled or disabled. The definitions are derived from ISO/IEC 10164-2 and are as follows:

```
OperationalState ::= INT-U16 {
    disabled(0),
    enabled(1),
    notAvailable(2)
}
```

6.1.3.16 Administrative state data type

The administrative state data type defines if a certain object is locked or unlocked. The definitions are derived from ISO/IEC 10164-2 and are as follows:

```
AdministrativeState ::= INT-U16 {
    locked(0),
    unlocked(1),
    shuttingDown(2)
}
```

6.1.3.17 Color data type

The color data type represents the basic RGB colors and is defined as follows:

```
--
-- 3 bits representing RGB, respectively
```

```
--
SimpleColour ::= INT-U16 {
    col-black(0),                   -- 000
    col-red(1),                     -- 100
    col-green(2),                   -- 010
    col-yellow(3),                  -- 110
    col-blue(4),                    -- 001
    col-magenta(5),                 -- 101
    col-cyan(6),                    -- 011
    col-white(7)                    -- 111
}
```

6.1.3.18 Locale data type

The locale data type shall be used to specify language and encoding information for data types that represent human-readable text strings. This data type is defined as follows:

```
Locale ::= SEQUENCE {
    language    INT-U32,        -- from ISO 639-1 or ISO 629-2, see below for encoding
    country     INT-U32,        -- from ISO 3166-1, ISO 3166-2, or ISO 3166-3, see below for encoding
    charset     CharSet,        -- format of character encoding
    str-spec    StringSpec
}

--
-- Charset names correspond to Internet Assigned Numbers Authority (IANA), the numeral constants are the
-- IANA MIBenum values for registered charsets
--
CharSet ::= INT-U16 {
    charset-unspec(0),
    charset-iso-10646-ucs-2(1000),  -- ISO 10646 two-octet character encoding scheme, big endian
    charset-iso-10646-ucs-4(1001),  -- ISO 10646 four-octet character encoding scheme, big endian
    charset-iso-8859-1(4),          -- encoding according to ISO/IEC 8859 Part 1
    charset-iso-8859-2(5),          -- encoding according to ISO/IEC 8859 Part 2
    charset-iso-8859-3(6),          -- encoding according to ISO/IEC 8859 Part 3
    charset-iso-8859-4(7),          -- encoding according to ISO/IEC 8859 Part 4
    charset-iso-8859-5(8),          -- encoding according to ISO/IEC 8859 Part 5
    charset-iso-8859-6(9),          -- encoding according to ISO/IEC 8859 Part 6
    charset-iso-8859-7(10),         -- encoding according to ISO/IEC 8859 Part 7
    charset-iso-8859-8(11),         -- encoding according to ISO/IEC 8859 Part 8
    charset-iso-8859-9(12),         -- encoding according to ISO/IEC 8859 Part 9
    charset-iso-8859-10(13),        -- encoding according to ISO/IEC 8859 Part 10
    charset-iso-8859-13(109),       -- encoding according to ISO/IEC 8859 Part 13
    charset-iso-8859-14(110),       -- encoding according to ISO/IEC 8859 Part 14
    charset-iso-8859-15(111),       -- encoding according to ISO/IEC 8859 Part 15
    charset-iso-2022-kr(37),        -- encoding according to RFC 1557 (Korean Character Encoding)
    charset-ks-c-5601(36),          -- encoding according to Korean Industrial Standard, KSC 5601-1987
    charset-iso-2022-jp(39),        -- encoding according to RFC 1468 (Japanese Character Encoding)
    charset-iso-2022-jp-2(40),      -- encoding according to RFC 1554 (Japanese Character Encoding)
    charset-jis-x0208(63),          -- encoding according to JIS X0208:1983,1990
    charset-iso-2022-cn(104),       -- encoding according to RFC 1922 (Chinese Character Encoding)
    charset-gb-2312(2025)           -- encoding according to Chinese Graphic Character Set, GB 2312:1980
}

StringSpec ::= SEQUENCE {
    str-max-len INT-U16,        -- maximum string length
    str-flags   StringFlags     -- specific flags for string representation and coding
}

StringFlags ::= BITS-16 {
```

```
        str-flag-nt(0)              -- strings are null terminated
    }
```

The field Locale::language shall represent the lowercase ISO/IEC 646 representation of a two-character language ID code from ISO 639-1 or ISO 639-2. For processing convenience, the language ID is stored in a 32- bit integer field. The first octet of the code is stored in the most significant byte of this field. Unused octets in the field are filled with NULL bytes.

Example:

Language:	“English”
Language identifier:	“en”
Encoding:	65 6E 00 00h

The field Locale::country shall represent the uppercase ISO/IEC 646 representation of a two-character coun- try ID code from ISO 3166-1, ISO 3166-2, or ISO 3166-3. For processing convenience, the country ID is stored in a 32-bit integer field. The first octet of the code is stored in the most significant byte of this field. Unused octets of the field are filled with NULL bytes.

The country code can be used to distinguish between certain aspects of the same language used in different countries, e.g., English in the United States versus English in the United Kingdom.

If no specific country is defined, this field shall be set to 0.

Example:

Country:	“United States”
Country identifier:	“US”
Encoding:	55 53 00 00h

The field Locale::charset denotes the encoding scheme of the characters used in string data types represent- ing readable text.

For interoperability, the character encoding scheme iso-10646-ucs-2 is recommended. This encoding scheme corresponds to ISO/IEC 10646 with a 2-octet (i.e., 16-bit per character) big-endian encoding, repre- senting the basic multilingual plane (BMP). The character codes within ISO/IEC 10646 do not correspond directly with glyphs, i.e., the graphical representation of a character. Also the ISO/IEC 10646 is language independent. Other Locale::charset values may be more language dependent because they also specify a cer- tain character repertoire.

6.1.3.19 External nomenclature reference data type

In certain cases, it is required to refer to standard coding systems (i.e., nomenclatures) that are outside the scope of this standard.

Example: The nomenclature defined in this standard does not define diagnostic codes or procedure codes. However, it is possible to reference a different coding system and provide the information in the form of an external code.

The external nomenclature reference data type is a special data type that is defined for this function as follows:

```
ExtNomenRef ::= SEQUENCE {
    nomenclature-id          OID-Type,          -- external nomenclature ID from external
                                                -- nomenclature partition
    nomenclature-code        ANY DEFINED BY  nomenclature-id
}
```

6.1.3.20 External object relation list data type

In certain cases, managed medical objects defined in the DIM may have relations to other objects that are not defined in this standard (i.e., they are external to the definitions).

The external object relation list data type can be used to provide information about these objects and the particular relation. This data type is defined as follows:

```
--
-- ExtObjRelationList
--
ExtObjRelationList ::= SEQUENCE OF ExtObjRelationEntry

ExtObjRelationEntry ::= SEQUENCE {
    relation-type            OID-Type,
    related-object           OID-Type,
    relation-attributes      AttributeList
}
```

Example 1: In certain situations, it is necessary to record specific production information (e.g., serial number) of a transducer that is used to derive a measurement. The transducer in this standard is not defined as a managed medical object. Therefore, the VMD object instances use a relation entry to supply the information, e.g., {relation-type = is-connected; related-object = Transducer; relation-attributes = {model, "A-Model," serial-number = "12345"}}.

Example 2: A certain numerical measurement value is manually validated by a nurse. A charting system keeps information about manual validations. The nurse is not modeled as an object in this standard. Therefore, the charting system uses a relation entry as an additional attribute of the Numeric object, e.g., {relation-type = validated-by; related-object = Nurse; relation-attributes = {name, "C. Smith," date, "041295"}}

The external object relation list data type is a very powerful concept to extend the information model without really defining additional classes of objects.

6.2 Top class

Class: Top
Description: "The Top class is the common inheritance base for most of the classes in the DIM."

Superclass: --
Subclasses: PM-Segment, VMO, Battery, Clock, Log, VMS, Operation, Scanner, CommunicationController, MultiPatientArchive, PatientArchive, Physician, SessionArchive, SessionNotes, SessionTest, PatientDemographics
Name Binding: -
Registered As: MDC_MOC_TOP

6.2.1 Attributes

The Top class defines the attributes in Table 1.

Table 1—Top class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Class	MDC_ATTR_CLASS	OID-Type	Defines the ID of the class; IDs come from the object-oriented nomenclature partition.	M
Name-Binding	MDC_ATTR_NAME_BINDING	OID-Type	Defines ID of Name Binding attribute, e.g., HANDLE; IDs come from the object-oriented nomenclature partition.	M

The Top class does not define any attribute groups.

6.2.2 Behavior

The Top class does not define any special methods.

6.2.3 Notifications

The Top class defines the events in Table 2.

Table 2—Top events

Event	Mode	Event ID	Event parameter	Event result
Attribute-Update	Confirmed/Unconfirmed	MDC_NOTI_ATTR_UPDATE	AttributeList	—

The attribute update notification allows all objects to communicate their attribute values with a generic event report. However, the use of this notification for systems with multiple object instances is not recommended. Instead, Scanner objects should be used.

6.3 Medical package

6.3.1 VMO class

Class: VMO
Description: "The VMO is the base class for all medical-related classes in the model. It provides consistent naming and identification across the Medical Package model. As a base class, the VMO cannot be instantiated."
Superclass: Top
Subclasses: Alert, AlertStatus, AlertMonitor, Channel, Metric, PM-Store, VMD, SCO

Name Binding: Handle (the value of the Handle attribute is sufficient for unique identification of an instance of a VMO-derived class in a device system)
Registered As: MDC_MOC_VMO

6.3.1.1 Attributes

The VMO class defines the attributes in Table 3.

Table 3—VMO class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Type	MDC_ATTR_ID_ TYPE	TYPE	Defines a specific static type of this object, as defined in the object-oriented or metric nomenclature partition.	M
Handle	MDC_ATTR_ID_ HANDLE	HANDLE	Locally unique short-hand identification.	M
Label-String	MDC_ATTR_ID_ LABEL_STRING	OCTET STRING	Textual representation of type ID.	O
Ext-Obj-Relations	MDC_ATTR_EXT_ OBJ_RELATION	ExtObjRelationList	Relations to objects that are not defined in the DIM.	O

The VMO class defines in Table 4 the attribute groups or extensions to inherited attribute groups.

Table 4—VMO class attribute groups

Attribute group	Attribute group ID	Group elements
VMO Static Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_STATIC	from VMO: Type, Handle
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_DYN	from VMO: Label-String
Relationship Attribute Group	MDC_ATTR_GRP_RELATION	from VMO: Ext-Obj-Relations

6.3.1.2 Behavior

The VMO class does not define any special methods.

6.3.1.3 Notifications

The VMO class does not generate any special notifications.

6.3.2 VMD class

Class: VMD
Description: "The VMD class is an abstraction of a medical-related subsystem (e.g., hardware or even pure software) of a medical device."
Superclass: VMO
Subclasses: --
Name Binding: Handle (VMO inherited)
Registered As: MDC_MOC_VMO_VMD

6.3.2.1 Attributes

The VMD class defines the attributes in Table 5.

Table 5—VMD class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Vmd-Status	MDC_ATTR_VMD_STAT	VMDStatus	Example: on.	M
Vmd-Model	MDC_ATTR_ID_MODEL	SystemModel	Manufacturer and model number.	O
Instance-Number	MDC_ATTR_ID_INSTNO	InstNumber	If multiple instances of the same VMD exist, this attribute helps to order the sequence.	O
Production-Specification	MDC_ATTR_ID_PROD_SPECN	ProductionSpec	Serial numbers and revisions; only present if VMD represents an independent subsystem.	O
Udi	MDC_ATTR_ID_UDI	UdiSpec	This attribute defines the UDI(s) under which this product is listed with an authority such as the US FDA.	O
Compatibility-Id	MDC_ATTR_ID_COMPAT	INT-U32	Static for manufacturer use.	O
Parameter-Group	MDC_ATTR_ID_PARAM_GRP	OID-Type	Example: cardiovascular.	O
Position	MDC_ATTR_ID_POSN	INT-U16	Example: slot number 0xffff marks an invalid or unknown position.	O
Operating-Hours	MDC_ATTR_TIME_PD_OP_HRS	INT-U32		O
Operation-Cycles	MDC_ATTR_CYC_OP	INT-U32	Example: number of measurements taken.	O
Measurement-Principle	MDC_ATTR_MSMT_PRINCIPLE	MsmtPrinciple	Describes the physical principle of the measurement.	O
Locale	MDC_ATTR_LOCALE	Locale	Defines charset and language of printable string attributes in this VMD and contained objects.	O

Identification and revision attributes are not needed if the VMD does not represent a hardware component.

The VMD class defines in Table 6 the attribute groups or extensions to inherited attribute groups.

Table 6—VMD class attribute groups

Attribute group	Attribute group ID	Group elements
VMO Static Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_STATIC	from VMO: Type, Handle from VMD: Instance-Number, Compatibility-Id, Parameter-Group, Measurement-Principle, Locale

Attribute group	Attribute group ID	Group elements
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_DYN	from VMO: Label-String from VMD: Vmd-Status
VMD Production Attribute Group	MDC_ATTR_GRP_VMD_PROD	from VMD: Vmd-Model, Production-Specification, Udi
VMD Application Attribute Group	MDC_ATTR_GRP_VMD_APPL	from VMD: Position, Operating-Hours, Operation-Cycles

NOTE—A separate attribute group is defined for VMD static attributes that are needed only in special applications.

The following ASN.1 data type definitions apply:

```
--
-- VMD status indication bits; all bits 0 indicate that VMD is operational
--
VMDStatus ::= BITS-16 {
      vmd-off(0),
      vmd-not-ready(1),          -- e.g. for an infusion pump that is not ready
      vmd-standby(2),            -- e.g. for device powered but not active
      vmd-transduc-discon(8),    -- transducer disconnected
      vmd-hw-discon(9)           -- measurement hardware disconnected
}

--
-- Physical principle of the measurement (multiple bits may be set)
--
MsmtPrinciple ::= BITS-16 {
      msp-other(0),
      msp-chemical(1),
      msp-electrical(2),
      msp-impedance(3),
      msp-nuclear(4),
      msp-optical(5),
      msp-thermal(6),
      msp-biological(7),
      msp-mechanical(8),
      msp-acoustical(9),
      msp-manual(15)
}
```

6.3.2.2 Behavior

The VMD class does not define any special methods.

6.3.2.3 Notifications

The VMD class does not generate any special notifications.

6.3.3 Channel class

Class: Channel
Description: "Channel objects are used for grouping Metric objects that have a contextual relationship with each other."

Superclass: VMO
Subclasses: --
Name Binding: Handle (VMO inherited)
Registered As: MDC_MOC_VMO_CHAN

6.3.3.1 Attributes

The Channel class defines the attributes in Table 7.

Table 7—Channel class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Channel-Id	MDC_ATTR_CHAN_ID	OID-Type	Dynamic identification.	O
Channel-Status	MDC_ATTR_CHAN_STAT	ChannelStatus	Example: Transducer Disconnected.	O
Physical-Channel-No	MDC_ATTR_CHAN_NUM_PHYS	INT-U16	Provides a reference to a particular hardware channel, e.g., A/D.	O
Logical-Channel-No	MDC_ATTR_CHAN_NUM_LOGICAL	INT-U16	Dynamic channel numbering.	O
Parameter-Group	MDC_ATTR_ID_PARAM_GRP	OID-Type	Static group of metrics, e.g., cardiovascular.	O
Measurement-Principle	MDC_ATTR_MSMT_PRINCIPLE	MsmtPrinciple	Describes the physical principle of the measurement.	O
Color	MDC_ATTR_COLOR	SimpleColour	Useful to assign a common color to objects in one channel.	O

The Channel class defines in Table 8 the attribute groups or extensions to inherited attribute groups.

Table 8—Channel class attribute groups

Attribute group	Attribute group ID	Group elements
VMO Static Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_STATIC	from VMO: Type, Handle from Channel: Physical-Channel-No, Parameter-Group, Measurement-Principle
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_DYN	from VMO: Label-String from Channel: Channel-Id, Channel-Status, Logical-Channel-No, Color

The following ASN.1 data type definitions apply:

```
--
-- Channel Status indication bits
--
ChannelStatus ::= BITS-16 {
      chan-off(0),
      chan-not-ready(1),
      chan-standby(2),
      chan-transduc-discon(8),
```

chan-hw-discon(9)
}

6.3.3.2 Behavior

The Channel class does not define any special methods.

6.3.3.3 Notifications

The Channel class does not generate any special notifications.

6.3.4 Metric class

Class: Metric
Description: "The Metric class is the base class for all classes representing direct and derived, quantitative and qualitative biosignal measurement, status, and context data. It is a base class that is used for inheritance only."
Superclass: VMO
Subclasses: ComplexMetric, Enumeration, Numeric, SampleArray
Name Binding: Handle (VMO inherited)
Registered As: MDC_MOC_VMO_METRIC

6.3.4.1 Attributes

The Metric class defines the attributes in Table 9.

Table 9—Metric class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Metric-Specification	MDC_ATTR_METRIC_SPECN	MetricSpec	Static; mandatory basic properties.	M
Max-Delay-Time	MDC_ATTR_DELAY_TIME_MAX	RelativeTime	Static; maximum delay to real time.	O
Metric-Status	MDC_ATTR_METRIC_STAT	MetricStatus		O
Measurement-Status	MDC_ATTR_MSMT_STAT	MeasurementStatus	Usually part of an observed value.	O
Metric-Id-Partition	MDC_ATTR_METRIC_ID_PART	NomPartition	Identifies the nomenclature partition associated with the MetricId if it is different from the partition specified in the object's VMO::Type attribute.	O
Metric-Id	MDC_ATTR_ID_PHYSIO	OID-Type	Contains dynamic identification (e.g., a specific blood pressure label) compared to the static, generic ID in the MetricSpecification. OID is from VMO::Type or Metric-Id-Partition partition. Usually this attribute is part of an observed value, not an individual attribute.	O

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Metric-Id-Ext	MDC_ATTR_ID_ MSMT_EXT	ExtNomenRef	Dynamic identification of the metric in a different nomenclature or dictionary. Use of this attribute severely limits interoperability of applications.	O
Unit-Code	MDC_ATTR_UNIT_ CODE	OID-Type	Example: mmHG;, usually part of observed value.	O
Unit-Labelstring	MDC_ATTR_UNIT_ LABEL_STRING	OCTET STRING	Textual representation of dimension.	O
Vmo-Source-List	MDC_ATTR_VMO_ LIST_SRC	VmoSourceList	Indicates sources of this metric in the form of references to other metrics.	O
Metric-Source-List	MDC_ATTR_ METRIC_LIST_SRC	MetricSourceList	Indicates sources of this metric in the form of a list of metric IDs.	O
Msmt-Site-List	MDC_ATTR_SITE_ LIST_MSMT	SiteList	Measurement sites, specified in internal nomenclature.	O
Msmt-Site-List-Ext	MDC_ATTR_SITE_ LIST_MSMT_EXT	SiteListExt	Measurement sites, specified in external nomenclature.	O
Body-Site-List	MDC_ATTR_SITE_ LIST_BODY	SiteList	Body sites, specified in internal nomenclature.	O
Body-Site-List-Ext	MDC_ATTR_SITE_ LIST_BODY_EXT	SiteListExt	Body sites, specified in external nomenclature.	O
Metric-Calibration	MDC_ATTR_ METRIC_CALIB	MetricCalibration	Indicates type and last time of calibration.	O
Color	MDC_ATTR_ COLOR	SimpleColour	Color for representation.	O
Measure-Mode	MDC_ATTR_ MODE_MSMT	PrivateOid	Manufacturer-specific measurement information.	O
Measure-Period	MDC_ATTR_TIME_ PD_MSMT	MetricMeasure	Measurement repetition time; not necessarily the same as update period.	O
Averaging-Period	MDC_ATTR_TIME_ PD_AVG	MetricMeasure	Time period used to average values, e.g., a metric for the average flow of last hour.	O
Start-Time	MDC_ATTR_TIME_ START	AbsoluteTime	Time when measurement activity was started, e.g., when infusion was started.	O
Stop-Time	MDC_ATTR_TIME_ STOP	AbsoluteTime	Time when measurement activity was stopped.	O
Metric-Info-Labelstring	MDC_ATTR_ METRIC_INFO_ LABEL_STRING	OCTET STRING	Textual attribute, e.g., to specify electrode displacements or other specific information about the measurement.	O
Substance	MDC_ATTR_ID_ SUBSTANCE	ExtNomenRef	Substance to which a metric pertains; expressed in nomenclature that is defined outside of this standard.	O
Substance-Labelstring	MDC_ATTR_ID_ SUBSTANCE_ LABEL_STRING	OCTET STRING	Textual attribute that identifies the substance.	O

The Metric class defines in Table 10 the attribute groups or extensions to inherited attribute groups.

Table 10—Metric class attribute groups

Attribute group	Attribute group ID	Group elements
VMO Static Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_STATIC	from VMO: Type, Handle from Metric: Metric-Specification, Max-Delay-Time
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_DYN	from VMO: Label-String from Metric: Metric-Status, Measurement-Status, Metric-Id, Metric-Id-Ext, Unit-Code, Unit- Labelstring, Vmo-Source-List, Metric-Source-List, Msmt-Site-List, Body-Site- List, Metric-Calibration, Color, Measure- Mode, Measure-Period, Averaging-Period, Start-Time, Stop-Time, Metric-Info-Labelstring, Substance, Substance-Labelstring, Body-Site-List-Ext, Msmt-Site- List-Ext
Metric Observed Value Group (extensible attribute group)	MDC_ATTR_GRP_METRIC_VAL_OBS	from Metric: Metric-Id-Partition

The following ASN.1 data type definitions apply:

```
--
-- Metric-Status attribute
--
MetricStatus ::= BITS-16 {
      metric-off(0),
      metric-not-ready(1),
      metric-standby(2),
      metric-transduc-discon(8),
      metric-hw-discon(9)
}

--
-- The Metric-Specification attribute defines all mandatory static properties of a Metric object
--
MetricSpec ::= SEQUENCE {
      update-period              RelativeTime,      -- minimum time between changes of observed value
      category                   MetricCategory,
      access                     MetricAccess,
      structure                  MetricStructure,
      relevance                  MetricRelevance
}

--
-- Structure describes if the object represents a single value or multiple related values (e.g., an invasive blood
-- pressure could be compound when it represents a pulsatile pressure and derives systolic, diastolic, and mean
```

```
-- values)
--
MetricStructure ::= SEQUENCE {
      ms-struct                  MetricComplexity,
      ms-comp-no                 INT-U8          -- maximum number of components in
                                                 -- compound/complex
}

--
-- The MetricComplexity value facilitates differentiation between simple, compound, and complex metric structures
--
MetricComplexity ::= INT-U8 {
      simple(0),
      compound(1),               -- multiple observed values, same dynamic context
      complex(2)                 -- multiple observed values, multiple dynamic contexts)
}

--
-- The MetricAccess bit field provides information on how it is possible to access the metric value and when a new
-- value is available
--
-- NOTES
-- 1--The avail-intermittent flag shall be set if the observed value is not always available
-- 2--Exactly one update mode bit (upd-) shall be set
-- 3--At least one access mode bit (acc-) shall be set
-- 4--It is possible to set scan option bits (sc-) only if the acc-scan bit is set
-- 5--If the acc-scan bit is set, at least one sc-opt bit shall be set
--
MetricAccess ::= BITS-16 {
      avail-intermittent(0),     -- value is intermittently available
      upd-periodic(1),           -- value is periodically (fixed period) updated
      upd-episodic(2),           -- value is episodically updated
      msmt-noncontinuous(3),     -- measurement is not continuous (e.g. NBP)
      acc-evrep(4),              -- metric sends event report for observed value
      acc-get(5),                -- metric supports explicit GET service
      acc-scan(6),               -- metric observed value is able to be accessed via Scanner object
      gen-opt-sync(8),           -- observed value shall be processed synchronously
      sc-opt-normal(10),         -- scan option: value can be scanned with update period
      sc-opt-extensive(11),      -- scan option: in update period multiple values may occur
      sc-opt-long-pd-avail(12),  -- scan option: value may be scanned slow (superpositive-avg scan bit)
      sc-opt-confirm(13),        -- scan option: scanner should operate in confirmed mode
      sc-opt-refresh(14)         -- scan option: a scan refresh operation is allowed
}

--
-- The metric category makes it possible to distinguish between measurements, settings, and calculations
--
MetricCategory ::= INT-U16 {
      mcat-unspec(0),
      auto-measurement(1),
      manual-measurement(2),
      auto-setting(3),
      manual-setting(4),
      auto-calculation(5),
      manual-calculation(6)
}

--
-- Metric relevance defines in what way the metric should be used (i.e., a value of 0 means normal)
--
MetricRelevance ::= BITS-16 {
```

```
        rv-unspec(0),                   -- relevance not specified; should normally not be used
        rv-internal(1),                 -- an internally used value only
        rv-store-only(2),               -- only relevant for storage
        rv-no-recording(3),             -- not relevant for recording
        rv-phys-ev-ind(4),              -- metric represents a physiological trigger (not a value)
        rv-btb-metric(5),               -- metric is calculated for each beat or breath, not time-averaged
        rv-app-specific(8),             -- dedicated application required to interpret the metric
        rv-service(9)                   -- metric is intended for service or diagnostic purposes
}

--
-- The Metric-Calibration attribute defines calibration methods and times
-- NOTE--Multiple entries allowed
--
MetricCalibration ::= SEQUENCE OF MetricCalEntry

MetricCalEntry ::= SEQUENCE {
    cal-type      MetricCalType,
    cal-state     MetricCalState,
    cal-time      AbsoluteTime
}

MetricCalType ::= INT-U16 {
    cal-unspec(0),
    cal-offset(1),                  -- offset calibration
    cal-gain(2),                    -- gain calibration
    cal-two-point(3)                -- two-point calibration
}

MetricCalState ::= INT-U16 {
    not-calibrated(0),
    cal-required(1),
    calibrated(2)
}

--
-- Ordered list of measurement sites, e.g., EEG electrode positions
-- Entries are from body site nomenclature partition
--
SiteList ::= SEQUENCE OF OID-Type

--
-- Site list may also refer to external nomenclatures to specify measurement sites
--
SiteListExt ::= SEQUENCE OF ExtNomenRef

--
-- Metric-Source-List attribute is an ordered list of metric OIDs
-- OIDs from VMO::Type or Metric-Id-Partition partition
--
MetricSourceList ::= SEQUENCE OF OID-Type

--
-- Vmo-Source-List attribute defines references to other VMO-derived objects that are used as sources of this
-- metric (this is an ordered list)
--
VmoSourceList ::= SEQUENCE OF VmoSourceEntry

VmoSourceEntry ::= SEQUENCE {
    vmo-type    OID-Type,           -- from object-oriented nomenclature partition
    glb-handle  GLB-HANDLE
```

```
}

--
-- Measurement-Status attribute defines the state of the measurement; used by derived classes
--
MeasurementStatus ::= BITS-16 {
      invalid(0),
      questionable(1),
      not-available(2),
      calibration-ongoing(3),
      test-data(4),
      demo-data(5),
      validated-data(8),          -- relevant e.g. in an archive
      early-indication(9),        -- early estimate of value
      msmt-ongoing(10),           -- indicates that a new measurement is just being taken (episodic)
      msmt-state-in-alarm(14),    -- indicates that the metric has an active alert condition
      msmt-state-al-inhibited(15) -- metric supports alarming, and alarms are turned off (optional)
}

--
-- In a number of derived metrics, specification of ranges is necessary
-- A type for this is defined here in the base class
--
AbsoluteRange ::= SEQUENCE {
      lower-value FLOAT-Type,
      upper-value FLOAT-Type
}

--
-- Metric measure is used for attributes that have a value and a dimension
--
MetricMeasure ::= SEQUENCE {
      value      FLOAT-Type,
      unit-code  OID-Type          -- from dimensions nomenclature partition
}
```

6.3.4.2 Behavior

The Metric class does not define any special methods.

6.3.4.3 Notifications

The Metric class does not generate any special notifications.

6.3.5 Numeric class

Class:	Numeric
Description:	"The Numeric class represents numerical measurements and status information, e.g., amplitude measures, counters."
Superclass:	Metric
Subclasses:	--
Name Binding:	Handle (VMO inherited)
Registered As:	MDC_MOC_VMO_METRIC_NU

6.3.5.1 Attributes

The Numeric class defines the attributes in Table 11.

Table 11—Numeric class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Nu-Observed-Value	MDC_ATTR_NU_VAL_OBS	NuObsValue	Example: measurement value; should also contain validity information to be useful.	C[a]
Compound-Nu-Observed-Value	MDC_ATTR_NU_CMPD_VAL_OBS	NuObsValueCmp	Used when multiple values are represented in a single Numeric object. (Structure is compound.)	C
Absolute-Time-Stamp	MDC_ATTR_TIME_STAMP_ABS	AbsoluteTime	Time of observation (timestamp).	O
Relative-Time-Stamp	MDC_ATTR_TIME_STAMP_REL	RelativeTime		O
Hires-Time-Stamp	MDC_ATTR_TIME_STAMP_REL_HI_RES	HighResRelativeTime	High-resolution timestamp.	O
Nu-Measure-Range	MDC_ATTR_NU_RANGE_MSMT	AbsoluteRange	Potential measurement range.	O
Nu-Physiological-Range	MDC_ATTR_NU_RANGE_PHYSIO	AbsoluteRange	Physiological reasonable range (note that this is not an alarming range).	O
Nu-Measure-Resolution	MDC_ATTR_NU_MSMT_RES	FLOAT-Type	Resolution of measurement; minimum difference between two observed values.	O
Display-Resolution	MDC_ATTR_DISP_RES	DispResolution	Used when different resolution is needed when value is displayed.	O
Accuracy	MDC_ATTR_NU_ACCUR_MSMT	FLOAT-Type	Maximum deviation of actual value from reported observed value (if it can be specified).	O

[a]Exactly one observed value type shall be present as defined by the Metric-Specification attribute.

The Numeric class defines in Table 12 the attribute groups or extensions to inherited attribute groups.

Table 12—Numeric class attribute groups

Attribute group	Attribute group ID	Group elements
VMO Static Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_STATIC	from VMO: Type, Handle from Metric: Metric-Specification, Max-Delay-Time
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_DYN	from VMO: Label-String from Metric: Metric-Status, Measurement-Status, Metric-Id, Metric-Id-Ext, Unit-Code, Unit- Labelstring, Vmo-Source-List, Metric-

Attribute group	Attribute group ID	Group elements
		Source-List, Msmt-Site-List, Body-Site- List, Metric-Calibration, Color, Measure- Mode, Measure-Period, Averaging-Period, Start-Time, Stop-Time, Metric-Info-Labelstring, Substance, Substance-Labelstring, Body-Site-List-Ext, Msmt-Site- List-Ext from Numeric: Nu-Measure-Range, Nu-Physiological-Range, Nu-Measure-Resolution, Display- Resolution, Accuracy
Metric Observed Value Group (extensible attribute group)	MDC_ATTR_GRP_METRIC_VAL_OBS	from Metric: Metric-Id-Partition from Numeric: Nu-Observed-Value, Absolute-Time-Stamp, Relative-Time-Stamp, Hires-Time- Stamp, Compound-Nu-Observed-Value

The following ASN.1 data type definitions apply:

```
--
-- Nu-Observed-Value attribute always includes identification, state, and dimension to ensure consistency with
-- minimal overhead
--
NuObsValue ::= SEQUENCE {
      metric-id    OID-Type,                    -- from VMO::Type or Metric-Id-Partition partition
      state        MeasurementStatus,           -- defined in Metric base
      unit-code    OID-Type,                    -- from dimensions nomenclature partition
      value        FLOAT-Type
}

--
-- Observed value for compound numerics
--
NuObsValueCmp ::= SEQUENCE OF NuObsValue

--
-- Display-Resolution attribute is the value representation on a display (may be lower resolution)
--
DispResolution ::= SEQUENCE {
      pre-point    INT-U8,          -- digits before decimal point
      post-point   INT-U8           -- digits after decimal point
}
```

6.3.5.2 Behavior

The Numeric class does not define any special methods.

6.3.5.3 Notifications

The Numeric class does not generate any special notifications.

6.3.6 SampleArray class

Class: SampleArray
Description: "The SampleArray class is the base class for metrics that have a graphical, curve type presentation and, therefore, have their observation values reported as arrays of data points by communicating systems."
Superclass: Metric
Subclasses: DistributionSampleArray, RealTimeSampleArray, TimeSampleArray
Name Binding: Handle (VMO inherited)
Registered As: MDC_MOC_VMO_METRIC_SA

6.3.6.1 Attributes

The SampleArray class defines the attributes in Table 13.

Table 13—SampleArray class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Sa-Observed-Value	MDC_ATTR_SA_VAL_OBS	SaObsValue	Example: array of measurement values.	C[a]
Compound-Sa-Observed-Value	MDC_ATTR_SA_CMPD_VAL_OBS	SaObsValueCmp		C
Sa-Specification	MDC_ATTR_SA_SPECN	SaSpec	Static description of sample array and sample types.	M
Compression	MDC_ATTR_COMPRES	PrivateOid	Defines potential compression algorithm.	O
Scale-And-Range-Specification-8	MDC_ATTR_SCALE_SPECN_I8	ScaleRangeSpec8	Defines mapping between samples and actual values as well as measurement range; type depends on sample size. Exactly one of the following attributes shall be implemented for any SampleArray object: Scale-And-Range-Specification-8, Scale-And-Range-Specification-16, Scale-And-Range-Specification-32.	C
Scale-And-Range-Specification-16	MDC_ATTR_SCALE_SPECN_I16	ScaleRangeSpec16	Defines mapping between samples and actual values as well as measurement range; type depends on sample size. Exactly one of the following attributes shall be implemented for any SampleArray object: Scale-And-Range-Specification-8, Scale-And-Range-Specification-16, Scale-And-Range-Specification-32.	C

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Scale-And-Range-Specification-32	MDC_ATTR_SCALE_SPECN_I32	ScaleRangeSpec32	Defines mapping between samples and actual values as well as measurement range; type depends on sample size. Exactly one of the following attributes shall be implemented for any SampleArray object: Scale-And-Range-Specification-8, Scale-And-Range-Specification-16, Scale-And-Range-Specification-32.	C
Sa-Physiological-Range-8	MDC_ATTR_SA_RANGE_PHYS_I8	ScaledRange8	For optimum display scaling, the physiologically meaningful range is specified.	O
Sa-Physiological-Range-16	MDC_ATTR_SA_RANGE_PHYS_I16	ScaledRange16	For optimum display scaling, the physiologically meaningful range is specified.	O
Sa-Physiological-Range-32	MDC_ATTR_SA_RANGE_PHYS_I32	ScaledRange32	For optimum display scaling, the physiologically meaningful range is specified.	O
Visual-Grid-8	MDC_ATTR_GRID_VIS_I8	SaVisualGrid8	Defines gridline positions on displays and recorders; type depends on sample size.	O
Visual-Grid-16	MDC_ATTR_GRID_VIS_I16	SaVisualGrid16	Defines gridline positions on displays and recorders; type depends on sample size.	O
Visual-Grid-32	MDC_ATTR_GRID_VIS_I32	SaVisualGrid32	Defines gridline positions on displays and recorders; type depends on sample size.	O
Sa-Calibration-Data-8	MDC_ATTR_SA_CALIB_I8	SaCalData8	Defines positions of calibration markers on display and recorders; type depends on sample size.	O
Sa-Calibration-Data-16	MDC_ATTR_SA_CALIB_I16	SaCalData16	Defines positions of calibration markers on display and recorders; type depends on sample size.	O
Sa-Calibration-Data-32	MDC_ATTR_SA_CALIB_I32	SaCalData32	Defines positions of calibration markers on display and recorders; type depends on sample size.	O
Filter-Specification	MDC_ATTR_FILTER_SPECN	SaFilterSpec		O
Filter-Label-String	MDC_ATTR_FILTER_LABEL_STRING	OCTET STRING	Text label of an active filter, e.g., ‘Butterworth’ or ‘LinearPhase.’	O
Sa-Signal-Frequency	MDC_ATTR_SA_FREQ_SIG	SaSignalFrequency	Maximum signal frequency.	O
Sa-Measure-Resolution	MDC_ATTR_SA_MSMT_RES	FLOAT-Type		O
Sa-Marker-List-8	MDC_ATTR_SA_MARKER_LIST_I8	MarkerListSaVal8		O
Sa-Marker-List-16	MDC_ATTR_SA_MARKER_LIST_I16	MarkerListSaVal16		O

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Sa-Marker-List-32	MDC_ATTR_SA_MARKER_LIST_I32	MarkerListSaVal32		O

[a]Exactly one observed value type shall be present as defined by the Metric-Specification attribute.

The SampleArray class defines in Table 14 the attribute groups or extensions to inherited attribute groups.

Table 14—SampleArray class attribute groups

Attribute group	Attribute group ID	Group elements
VMO Static Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_STATIC	from VMO: Type, Handle from Metric: Metric-Specification, Max-Delay-Time from SampleArray: Sa-Specification, Compression, Sa-Marker- List-8, Sa-Marker-List-16, Sa-Marker-List-32
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_DYN	from VMO: Label-String from Metric: Metric-Status, Measurement-Status, Metric-Id, Metric-Id-Ext, Unit-Code, Unit- Labelstring, Vmo-Source-List, Metric-Source-List, Msmt-Site-List, Body-Site- List, Metric-Calibration, Color, Measure- Mode, Measure-Period, Averaging-Period, Start-Time, Stop-Time, Metric-Info-Labelstring, Substance, Substance-Labelstring, Body-Site-List-Ext, Msmt-Site- List-Ext from SampleArray: Scale-And-Range-Specification-8, Sa-Physiological-Range-8, Visual-Grid-8, Sa- Calibration-Data-8, Filter-Specification, Filter-Label-String, Sa-Signal-Frequency, Sa-Measure-Resolution, Scale-And-Range- Specification-16, Scale-And-Range-Specification-32, Sa-Physiological-Range-16, Sa-Physiological-Range-32, Visual-Grid-16, Visual-Grid-32, Sa-Calibration-Data-16, Sa-Calibration-Data-32
Metric Observed Value Group (extensible attribute group)	MDC_ATTR_GRP_METRIC_VAL_OBS	from Metric: Metric-Id-Partition from SampleArray:

Attribute group	Attribute group ID	Group elements
		Sa-Observed-Value, Compound-Sa-Observed-Value

The following ASN.1 data type definitions apply:

```
--
-- Sa-Observed-Value attribute
--
SaObsValue ::= SEQUENCE {
    metric-id    OID-Type,                  -- from VMO::Type or Metric-Id-Partition partition
    state        MeasurementStatus,         -- defined in Metric base class
    array        OCTET STRING
}

--
-- Compound-Sa-Observed-Value attribute is the compound observed value
--
SaObsValueCmp ::= SEQUENCE OF SaObsValue

--
-- Sa-Specification attribute
--
SaSpec ::= SEQUENCE {
    array-size                  INT-U16,          -- number of samples per metric update period
    sample-type                 SampleType,
    flags                       SaFlags
}

-- Sample type describes one sample in the observed value array
--
SampleType ::= SEQUENCE {
    sample-size                 INT-U8,           -- e.g. 8 for 8bit samples, 16 for 16bit samples, shall be
                                                  -- divisible by 8
    significant-bits            SignificantBits   -- defines significant bits in one sample; if value is 255,
                                                  -- the samples are signed; all bits are significant;
                                                  -- samples are interpreted in twos complement
}

SignificantBits ::= INT-U8 {
    signed-samples(255)
}

--
-- SaFlags data type defines additional wave form properties
--
SaFlags ::= BITS-16 {
    smooth-curve(0),            -- for optimum display, use a smoothing algorithm
    delayed-curve(1),           -- curve is delayed (not real time)
    static-scale(2),            -- ScaleRangeSpec does not change
    sa-ext-val-range(3)         -- the nonsignificant bits in a sample are not 0, e.g., when they are used for
                                -- annotations or markers; the receiver must apply a bit mask to extract the
                                -- significant bits from the sample
}

--
-- Specification of an applied signal filter
--
SaFilterSpec ::= SEQUENCE OF SaFilterEntry
```

```
SaFilterEntry ::= SEQUENCE {
    filter-type                 FilterType,
    filter-frequency            FLOAT-Type,
    filter-order                INT-I16         -- e.g. -1: 6 dB/oct
}

FilterType ::= INT-U16 {
    other(0),
    low-pass(1),
    high-pass(2),
    notch(3)
}

--
-- Scale-and-Range-Specification attribute describes a relation between scaled values and absolute values;
-- depending on the sample size, multiple attribute types exist
--
-- NOTE--If a wave does not represent absolute values, the absolute value fields should contain a special value; if
-- the Sa-Specification attribute indicates signed samples, the scaled values have to be interpreted as signed values
--
ScaleRangeSpec8 ::= SEQUENCE {
    lower-absolute-value        FLOAT-Type,
    upper-absolute-value        FLOAT-Type,
    lower-scaled-value          INT-U8,
    upper-scaled-value          INT-U8
}

ScaleRangeSpec16 ::= SEQUENCE {
    lower-absolute-value        FLOAT-Type,
    upper-absolute-value        FLOAT-Type,
    lower-scaled-value          INT-U16,
    upper-scaled-value          INT-U16
}

ScaleRangeSpec32 ::= SEQUENCE {
    lower-absolute-value        FLOAT-Type,
    upper-absolute-value        FLOAT-Type,
    lower-scaled-value          INT-U32,
    upper-scaled-value          INT-U32
}

--
-- Visual-Grid attribute defines grid lines at different levels of grid lines; if the Sa-Specification attribute indicates
-- signed samples, the scaled values have to be interpreted as signed values
--
SaVisualGrid8 ::= SEQUENCE OF SaGridEntry8

SaGridEntry8 ::= SEQUENCE {
    absolute-value              FLOAT-Type,
    scaled-value                INT-U8,
    level                       INT-U8
}

SaVisualGrid16 ::= SEQUENCE OF SaGridEntry16

SaGridEntry16 ::= SEQUENCE {
    absolute-value              FLOAT-Type,
    scaled-value                INT-U16,
    level                       INT-U16
}
```

```
SaVisualGrid32 ::= SEQUENCE OF SaGridEntry32

SaGridEntry32 ::= SEQUENCE {
    absolute-value          FLOAT-Type,
    scaled-value            INT-U32,
    level                   INT-U16
}

--
-- Sa-Calibration-Data attribute defines calibration markers on a display or on a recording strip; if the Sa-
-- Specification attribute indicates signed samples, the scaled values have to be interpreted as signed values
--
SaCalData8 ::= SEQUENCE {
    lower-absolute-value    FLOAT-Type,
    upper-absolute-value    FLOAT-Type,
    lower-scaled-value      INT-U8,
    upper-scaled-value      INT-U8,
    increment               INT-U16,        -- scaled value for each step of the stair
    cal-type                SaCalDataType
}

SaCalData16 ::= SEQUENCE {
    lower-absolute-value    FLOAT-Type,
    upper-absolute-value    FLOAT-Type,
    lower-scaled-value      INT-U16,
    upper-scaled-value      INT-U16,
    increment               INT-U16,        -- scaled value for each step of the stair
    cal-type                SaCalDataType
}

SaCalData32 ::= SEQUENCE {
    lower-absolute-value    FLOAT-Type,
    upper-absolute-value    FLOAT-Type,
    lower-scaled-value      INT-U32,
    upper-scaled-value      INT-U32,
    increment               INT-U32,        -- scaled value for each step of the stair
    cal-type                SaCalDataType
}

SaCalDataType ::= INT-U16 {
    bar(0),     -- display a calibration bar
    stair(1)    -- display a calibration stair
}

--
-- Sa-Signal-Frequency attribute specifies the signal frequency
--
SaSignalFrequency ::= SEQUENCE {
    low-edge-freq           FLOAT-Type,
    high-edge-freq          FLOAT-Type      -- both in Hz
}

--
-- Sa-Physiological-Range attribute data types
-- If the Sa-Specification attribute indicates signed samples, the scaled values have to be interpreted as signed
-- values
--

ScaledRange8 ::= SEQUENCE {
    lower-scaled-value      INT-U8,
    upper-scaled-value      INT-U8
```

```
}

ScaledRange16 ::= SEQUENCE {
    lower-scaled-value          INT-U16,
    upper-scaled-value          INT-U16
}

ScaledRange32 ::= SEQUENCE {
    lower-scaled-value          INT-U32,
    upper-scaled-value          INT-U32
}

--
-- Sa-Marker-List attribute allows the definition of special sample values to mark or annotate certain conditions
-- directly in the sample value; the special sample value may be a full value or a bit mask, depending on the marker
-- ID; in any case, the sample value may use bits outside the normal range (as defined by the SampleType::
-- significant-bits) only if the SaFlags::sa-ext-val-range flag is set
--
MarkerListSaVal8 ::= SEQUENCE OF MarkerEntrySaVal8

MarkerEntrySaVal8 ::= SEQUENCE {
    marker-id   OID-Type,       -- from VMO::Type or Metric-Id-Partition partition
    marker-val  INT-U8,         -- a value or bit mask depending on marker-id
    unused      INT-U8          -- for alignment
}

MarkerListSaVal16 ::= SEQUENCE OF MarkerEntrySaVal16

MarkerEntrySaVal16 ::= SEQUENCE {
    marker-id   OID-Type,       -- from VMO::Type or Metric-Id-Partition partition
    marker-val  INT-U16         -- a value or bit mask depending on marker-id
}

MarkerListSaVal32 ::= SEQUENCE OF MarkerEntrySaVal32

MarkerEntrySaVal32 ::= SEQUENCE {
    marker-id   OID-Type,       -- from VMO::Type or Metric-Id-Partition partition
    marker-val  INT-U32         -- a value or bit mask depending on marker-id
}
```

6.3.6.2 Behavior

The SampleArray class does not define any special methods.

6.3.6.3 Notifications

The SampleArray class does not generate any special notifications.

6.3.7 RealTimeSampleArray class

Class:	RealTimeSampleArray
Description:	"The RealTimeSampleArray class describes a sample array that represents a real-time continuous waveform."
Superclass:	SampleArray
Subclasses:	--
Name Binding:	Handle (VMO inherited)
Registered As:	MDC_MOC_VMO_METRIC_SA_RT

6.3.7.1 Attributes

The RealTimeSampleArray class defines the attributes in Table 15.

Table 15—RealTimeSampleArray class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Sample-Period	MDC_ATTR_TIME_PD_SAMP	RelativeTime	Example: in (parts of) milliseconds.	M
Sweep-Speed	MDC_ATTR_SPD_SWEEP_DEFAULT	MetricMeasure	Example: millimeters per second.	O
Average-Reporting-Delay	MDC_ATTR_REPORTING_DELAY_AVG	RelativeTime	Indicates the average time between when the first element in an array update was sampled and when the FastPeriCfgScanner event report was generated (i.e., the event report timestamp).	O
Sample-Time-Sync	MDC_ATTR_SAMPLE_TIME_SYNC	RelativeTime	Indicates the precise sample time of the first element in an array update. Optional if the Average-Reporting-Delay attribute is present; out of the scope of this standard otherwise.	C
Hires-Sample-Time-Sync	MDC_ATTR_SAMPLE_TIME_SYNC_HIRES	HighResRelativeTime	Indicates the precise sample time of the first element in an array update. Optional if the Average-Reporting-Delay attribute is present; out of the scope of this standard otherwise.	C

NOTE 1—Together with the Average-Reporting-Delay attribute, the Sample-Time-Sync or HiRes-Sample-Time-Sync attribute can be used to accurately specify specific sample times. The Sample-Time-Sync and HiRes-Sample-Time-Sync attributes should be reported by an episodic scanner

- When reporting is first started and
- Periodically after that start at a frequency that ensures that time drift/clock skew will not compromise precise time correlation with a single waveform sample.

NOTE 2—See also 6.7.5 for the definition of the FastPeriCfgScanner class.

The RealTimeSampleArray class defines in Table 16 the attribute groups or extensions to inherited attribute groups.

Table 16—RealTimeSampleArray class attribute groups

Attribute group	Attribute group ID	Group elements
VMO Static Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_STATIC	from VMO: Type, Handle from Metric: Metric-Specification, Max-Delay-Time from SampleArray:

Attribute group	Attribute group ID	Group elements
		Sa-Specification, Compression, Sa-Marker- List-8, Sa-Marker-List-16, Sa-Marker-List-32 from RealTimeSampleArray: Sample-Period, Sweep-Speed, Average-Reporting-Delay
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_DYN	from VMO: Label-String from Metric: Metric-Status, Measurement-Status, Metric-Id, Metric-Id-Ext, Unit-Code, Unit- Labelstring, Vmo-Source-List, Metric-Source-List, Msmt-Site-List, Body-Site-List, Metric-Calibration, Color, Measure- Mode, Measure-Period, Averaging-Period, Start-Time, Stop-Time, Metric-Info-Labelstring, Substance, Substance-Labelstring, Body-Site-List-Ext, Msmt-Site- List-Ext from SampleArray: Scale-And-Range-Specification-8, Sa-Physiological-Range-8, Visual-Grid-8, Sa- Calibration-Data-8, Filter-Specification, Filter-Label-String, Sa-Signal-Frequency, Sa-Measure-Resolution, Scale-And-Range- Specification-16, Scale-And-Range-Specification-32, Sa-Physiological-Range-16, Sa-Physiological-Range-32, Visual-Grid-16, Visual-Grid-32, Sa-Calibration-Data-16, Sa-Calibration-Data-32 from RealTimeSampleArray: Sample-Time-Sync, Hires-Sample-Time- Sync
Metric Observed Value Group (extensible attribute group)	MDC_ATTR_GRP_METRIC_VAL_OBS	from Metric: Metric-Id-Partition from SampleArray: Sa-Observed-Value, Compound-Sa-Observed-Value

6.3.7.2 Behavior

The RealTimeSampleArray class does not define any special methods.

6.3.7.3 Notifications

The RealTimeSampleArray class does not generate any special notifications.

6.3.8 TimeSampleArray class

Class: TimeSampleArray
Description: "The TimeSampleArray class describes a sample array that represents noncontinuous waveforms (i.e., a wave snippet)."
Superclass: SampleArray
Subclasses: --
Name Binding: Handle (VMO inherited)
Registered As: MDC_MOC_VMO_METRIC_SA_T

6.3.8.1 Attributes

The TimeSampleArray class defines the attributes in Table 17.

Table 17—TimeSampleArray class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Absolute-Time-Stamp	MDC_ATTR_TIME_STAMP_ABS	AbsoluteTime	Time of observation (timestamp).	O
Relative-Time-Stamp	MDC_ATTR_TIME_STAMP_REL	RelativeTime		O
Hires-Time-Stamp	MDC_ATTR_TIME_STAMP_REL_HI_RES	HighResRelativeTime	High-resolution timestamp.	O
Sample-Period	MDC_ATTR_TIME_PD_SAMP	RelativeTime	Example: in (parts of) milliseconds.	M
Sweep-Speed	MDC_ATTR_SPD_SWEEP_DEFAULT	MetricMeasure	Example: millimeters per second.	O
Tsa-Marker-List	MDC_ATTR_TSA_MARKER_LIST	MarkerListRelTim	Marks positions in wave snippets.	O

The TimeSampleArray class defines in Table 18 the attribute groups or extensions to inherited attribute groups.

Table 18—TimeSampleArray class attribute groups

Attribute group	Attribute group ID	Group elements
VMO Static Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_STATIC	from VMO: Type, Handle from Metric: Metric-Specification, Max-Delay-Time from SampleArray: Sa-Specification, Compression, Sa-Marker-List-8, Sa-Marker-List-16, Sa-Marker-List-32 from TimeSampleArray: Sample-Period, Sweep-Speed
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_DYN	from VMO: Label-String from Metric: Metric-Status, Measurement-Status, Metric-Id, Metric-Id-Ext, Unit-Code, Unit-Labelstring, Vmo-Source-List, Metric-

Attribute group	Attribute group ID	Group elements
		Source-List, Msmt-Site-List, Body-Site-List, Metric-Calibration, Color, Measure-Mode, Measure-Period, Averaging-Period, Start-Time, Stop-Time, Metric-Info-Labelstring, Substance, Substance-Labelstring, Body-Site-List-Ext, Msmt-Site-List-Ext from SampleArray: Scale-And-Range-Specification-8, Sa-Physiological-Range-8, Visual-Grid-8, Sa-Calibration-Data-8, Filter-Specification, Filter-Label-String, Sa-Signal-Frequency, Sa-Measure-Resolution, Scale-And-Range-Specification-16, Scale-And-Range-Specification-32, Sa-Physiological-Range-16, Sa-Physiological-Range-32, Visual-Grid-16, Visual-Grid-32, Sa-Calibration-Data-16, Sa-Calibration-Data-32
Metric Observed Value Group (extensible attribute group)	MDC_ATTR_GRP_METRIC_VAL_OBS	from Metric: Metric-Id-Partition from SampleArray: Sa-Observed-Value, Compound-Sa-Observed-Value from TimeSampleArray: Absolute-Time-Stamp, Relative-Time-Stamp, Hires-Time-Stamp, Tsa-Marker-List

The following ASN.1 data type definitions apply:

```
--
-- Tsa-Marker-List attribute can be used to mark certain time points in the wave snippet; the first sample is at
-- relative time 0
--
MarkerListRelTim ::= SEQUENCE OF MarkerEntryRelTim

MarkerEntryRelTim ::= SEQUENCE {
    marker-id                   OID-Type,          -- from VMO::Type or Metric-Id-Partition partition
    marker-time                 RelativeTime
}
```

6.3.8.2 Behavior

The TimeSampleArray class does not define any special methods.

6.3.8.3 Notifications

The TimeSampleArray class does not generate any special notifications.

6.3.9 DistributionSampleArray class

Class: DistributionSampleArray
Description: "The DistributionSampleArray class describes a sample array that represents linear value distributions in the form of arrays containing scaled sample values. Each value within an observation array represents a measured value from a different point in the parameter space~~The index of a value within an observation array denotes a spatial value, not a time point~~. Thus, the observed value array can be considered an x-y coordinate system where the y axis is specified by the attributes inherited from the Metric class and the x axis is specified by attributes defined in the DistributionSampleArray class."
Superclass: SampleArray
Subclasses: --
Name Binding: Handle (VMO inherited)
Registered As: MDC_MOC_VMO_METRIC_SA_D

6.3.9.1 Attributes

The DistributionSampleArray class defines the attributes in Table 19.

Table 19—DistributionSampleArray class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Absolute-Time-Stamp	MDC_ATTR_TIME_STAMP_ABS	AbsoluteTime	Time of observation (timestamp).	O
Relative-Time-Stamp	MDC_ATTR_TIME_STAMP_REL	RelativeTime		O
Hires-Time-Stamp	MDC_ATTR_TIME_STAMP_REL_HI_RES	HighResRelativeTime	High-resolution timestamp.	O
Distribution-Range-Specification	MDC_ATTR_RANGE_DISTRIB	DsaRangeSpec	Maps array index to absolute value.	M
x-Unit-Code	MDC_ATTR_UNIT_CODE_X	OID-Type	Applies to x axis.	O
x-Unit-Label-String	MDC_ATTR_UNIT_LABEL_STRING_X	OCTET STRING	Applies to x axis.	O
Dsa-Marker-List	MDC_ATTR_DSA_MARKER_LIST	MarkerListIndex	User to mark positions in DistributionSampleArray object samples	O

The DistributionSampleArray class defines in Table 20 the attribute groups or extensions to inherited attribute groups.

Table 20—DistributionSampleArray class attribute groups

Attribute group	Attribute group ID	Group elements
VMO Static Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_STATIC	from VMO: Type, Handle from Metric: Metric-Specification, Max-Delay-Time from SampleArray: Sa-Specification, Compression, Sa-Marker-

Attribute group	Attribute group ID	Group elements
		List-8, Sa-Marker-List-16, Sa-Marker-List-32
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_DYN	from VMO: Label-String from Metric: Metric-Status, Measurement-Status, Metric-Id, Metric-Id-Ext, Unit-Code, Unit-Labelstring, Vmo-Source-List, Metric-Source-List, Msmt-Site-List, Body-Site-List, Metric-Calibration, Color, Measure-Mode, Measure-Period, Averaging-Period, Start-Time, Stop-Time, Metric-Info-Labelstring, Substance, Substance-Labelstring, Body-Site-List-Ext, Msmt-Site-List-Ext from SampleArray: Scale-And-Range-Specification-8, Sa-Physiological-Range-8, Visual-Grid-8, Sa-Calibration-Data-8, Filter-Specification, Filter-Label-String, Sa-Signal-Frequency, Sa-Measure-Resolution, Scale-And-Range-Specification-16, Scale-And-Range-Specification-32, Sa-Physiological-Range-16, Sa-Physiological-Range-32, Visual-Grid-16, Visual-Grid-32, Sa-Calibration-Data-16, Sa-Calibration-Data-32 from DistributionSampleArray: Distribution-Range-Specification, x-Unit-Code, x-Unit-Label-String
Metric Observed Value Group (extensible attribute group)	MDC_ATTR_GRP_METRIC_VAL_OBS	from Metric: Metric-Id-Partition from SampleArray: Sa-Observed-Value, Compound-Sa-Observed-Value from DistributionSampleArray: Absolute-Time-Stamp, Relative-Time-Stamp, Hires-Time-Stamp, Dsa-Marker-List

The following ASN.1 data type definitions apply:

```
--
-- Distribution-Range-Specification attribute defines the absolute value for the first and last array element; a linear
-- scale is assumed here unless a specific compression scheme is defined (last-value -first-value)/no.of.array
-- elements == step width
--
DsaRangeSpec ::= SEQUENCE {
      first-element-value              FLOAT-Type,
      last-element-value               FLOAT-Type
}
```

```
--
-- DSA-Marker-List attribute allows the annotation of samples by referencing the sample with an index
--
MarkerListIndex ::= SEQUENCE OF MarkerEntryIndex

MarkerEntryIndex ::= SEQUENCE {
    marker-id                   OID-Type,       -- from VMO::Type or Metric-Id-Partition partition
    marker-index                INT-U16
}
```

6.3.9.2 Behavior

The DistributionSampleArray class does not define any special methods.

6.3.9.3 Notifications

The DistributionSampleArray class does not generate any special notifications.

6.3.10 Enumeration class

Class: Enumeration
Description: "The Enumeration class represents status information and/or annotation information. Observation values may be presented in the form of normative codes (that are included in the nomenclature defined in this standard or in some other external nomenclature), bit string, or in the form of free text."
Superclass: Metric
Subclasses: --
Name Binding: Handle (VMO inherited)
Registered As: MDC_MOC_VMO_METRIC_ENUM

6.3.10.1 Attributes

The Enumeration class defines the attributes in Table 21.

Table 21—Enumeration class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Enum-Observed-Value	MDC_ATTR_VAL_ENUM_OBS	EnumObsValue	Either Enum-Observed-Value or Compound-EnumObserved-Value shall be supported in one object instance.	C
Compound-Enum-Observed-Value	MDC_ATTR_VAL_ENUM_OBS_CMPD	EnumObsValueCmp	Either Enum-Observed-Value or Compound-Enum-Observed-Value shall be supported in one object instance.	C
Absolute-Time-Stamp	MDC_ATTR_TIME_STAMP_ABS	AbsoluteTime		O
Relative-Time-Stamp	MDC_ATTR_TIME_STAMP_REL	RelativeTime		O
Hires-Time-Stamp	MDC_ATTR_TIME_STAMP_REL_HI_RES	HighResRelativeTime	High-resolution timestamp.	O

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Enum-Measure-Range	MDC_ATTR_ ENUM_RANGE_ MSMT	EnumMsmtRange	List of supported observed value OIDs. Optional if the OID type (EnumVal::enumobjid) is used in the observed value; out of the scope of this standard otherwise.	C
Enum-Measure-Range-Bit-String	MDC_ATTR_ ENUM_RANGE_ MSMT_BIT_ STRING	BITS-32	List of supported observed value bits in the bit string data type. Optional if the bit string type (EnumVal::enum-bit-str) is used in the observed value; out of the scope of this standard otherwise.	C
Enum-Measure-Range-Labels	MDC_ATTR_ ENUM_RANGE_ MSMT_LABELS	EnumMsmtRangeLabels	Associates text strings with specific enumeration values.	O

The Enumeration class defines in Table 22 the attribute groups or extensions to inherited attribute groups.

Table 22—Enumeration class attribute groups

Attribute group	Attribute group ID	Group elements
VMO Static Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_STATIC	from VMO: Type, Handle from Metric: Metric-Specification, Max-Delay-Time from Enumeration: Enum-Measure-Range-Labels
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_DYN	from VMO: Label-String from Metric: Metric-Status, Measurement-Status, Metric-Id, Metric-Id-Ext, Unit-Code, Unit- Labelstring, Vmo-Source-List, Metric-Source-List, Msmt-Site-List, Body-Site- List, Metric-Calibration, Color, Measure- Mode, Measure-Period, Averaging-Period, Start-Time, Stop-Time, Metric-Info-Labelstring, Substance, Substance-Labelstring, Body-Site-List-Ext, Msmt-Site- List-Ext from Enumeration: Enum-Measure-Range, Enum-Measure- Range-Bit-String
Metric Observed Value Group (extensible attribute group)	MDC_ATTR_GRP_METRIC_VAL_OBS	from Metric: Metric-Id-Partition from Enumeration:

Attribute group	Attribute group ID	Group elements
		Enum-Observed-Value, Absolute-Time-Stamp, Relative-Time-Stamp, Hires-Time-Stamp

The following ASN.1 data type definitions apply:

```
--
-- Enum-Observed-Value attribute
--
EnumObsValue ::= SEQUENCE {
      metric-id     OID-Type,                    -- from VMO::Type or Metric-Id-Partition partition
      state         MeasurementStatus,
      value         EnumVal                      -- supports different value data types
}

--
-- The enumeration value data type is used to provide different specific observation data types as follows
-- (Note that the type of measurement is coded in the top level structure EnumObsVal::metric-id)
--
--      enum-obj-id:                    used to communicate a metric OID, e.g., as an annotation
--                                      or other event defined in the VMO::Type or Metric-Id-
--                                      Partition partition
--      enum-text-string:               used to communicate a free text string (e.g., a status
--                                      message)
--      enum-external-code:             used to provide the code of an external nomenclature (e.g.,
--                                      could be used for procedure codes not covered in the
--                                      standard nomenclature)
--      enum-bit-str:                   for coding bit string values; the bit string data type must be
--                                      defined separately, e.g., in the nomenclature or in a
--                                      device-specific standard
--      enum-record-metric/oo:          allows the identification of additional data types by a
--                                      nomenclature code from the VMO::Type or
--                                      Metric-Id-Partition partition; the appended data type must
--                                      be defined separately, e.g., in a device-specific standard
--      enum-numeral:                   used to provide numeral enumerated values that must be
--                                      defined separately, e.g., in a device-specific standard;
--                                      this type is not to be used for numeric measurements
--
EnumVal ::= CHOICE {
      enum-obj-id             [1] OID-Type,           -- from VMO::Type or Metric-Id-
                                                      -- Partition partition
      enum-text-string        [2] OCTET STRING,       -- free text
      enum-external-code      [8] ExtNomenRef,        -- code defined in other coding
                                                      -- system
      enum-bit-str            [16] BITS-32,           -- bit string
      enum-record-metric      [33] EnumRecordMetric,  -- record type defined by ID from
                                                      -- VMO::Type or Metric-Id-
                                                      -- Partition partition
      enum-record-oo          [34] EnumRecordOo,      -- record type defined by ID from
                                                      -- object-oriented nomenclature
                                                      -- partition
      enum-numeral            [64] INT-U32            -- enumerated integer value
}

--
-- Record data type with structure and contents defined by a nomenclature ID from the VMO::Type or Metric-Id-
-- Partition partition
```

```
--
EnumRecordMetric ::= SEQUENCE {
    record-type-code        OID-Type,           -- from VMO::Type or Metric-Id-Partition partition
    record-data             ANY DEFINED BY  record-type-code
}

--
-- Record data type with structure and contents defined by a nomenclature ID from the object-oriented
-- nomenclature partition
--
EnumRecordOo ::= SEQUENCE {
    record-type-code        OID-Type,           -- must be from object-oriented nomenclature partition
    record-data             ANY DEFINED BY  record-type-code
}

--
-- Compound-Enum-Observed-Value attribute is the compound observed value
--
EnumObsValueCmp ::= SEQUENCE OF EnumObsValue

--
-- Enum-Measure-Range attribute defines the set of potential (i.e., legal) values of the Enum-Observed-Value
-- attribute (only allowed when EnumVal::enum-obj-id type is used)
-- OID-Types from VMO::Type or Metric-Id-Partition partition
--
EnumMsmtRange ::= SEQUENCE OF OID-Type

--
-- Enum-Measure-Range-Labels attribute defines both the set of potential (i.e., legal) values of the Enum-
-- Observed-Value attribute as well as a text label that can be associated with each enumeration value
--
EnumMsmtRangeLabels ::= SEQUENCE OF EnumMsmtRangeLabel

EnumMsmtRangeLabel ::= SEQUENCE {
    value       EnumVal,                    -- specific enumeration setting
    label       OCTET STRING                -- textual label associated with value
}
```

6.3.10.2 Behavior

The Enumeration class does not define any special methods.

6.3.10.3 Notifications

The Enumeration class does not generate any special notifications.

6.3.11 ComplexMetric class

Class: ComplexMetric
Description: "ComplexMetric objects act as a containers objects for other Metric objects, enabling reporting of the collection as a single semantic entity."
Superclass: Metric
Subclasses: --
Name Binding: Handle (VMO inherited)
Registered As: MDC_MOC_VMO_METRIC_CMPLX

6.3.11.1 Attributes

The ComplexMetric class defines the attributes in Table 23.

Table 23—ComplexMetric class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Cmplx-Metric-Info	MDC_ATTR_CMPLX_INFO	CmplxMetricInfo	Static attribute defining the object types used in the container.	M
Cmplx-Observed-Value	MDC_ATTR_CMPLX_VAL_OBS	CmplxObsValue		M
Cmplx-Dyn-Attr	MDC_ATTR_CMPLX_DYN_ATTR	CmplxDynAttr	Dynamic attributes of the individual objects within the ComplexMetric object.	O
Cmplx-Static-Attr	MDC_ATTR_CMPLX_STATIC_ATTR	CmplxStaticAttr	Static attributes of the individual objects within the ComplexMetric object.	O
Cmplx-Recursion-Depth	MDC_ATTR_CMPLX_RECURSION_DEPTH	INT-U16	Mandatory if the ComplexMetric object contains further ComplexMetric objects (e.g., recursion). If so, the attribute defines the maximum recursion depth.	C

ComplexMetric objects shall set the Metric::MetricSpec::structure::ms-struct::complex flag.

The ComplexMetric class defines in Table 24 the attribute groups or extensions to inherited attribute groups.

Table 24—ComplexMetric class attribute groups

Attribute group	Attribute group ID	Group elements
VMO Static Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_STATIC	from VMO: Type, Handle from Metric: Metric-Specification, Max-Delay-Time from ComplexMetric: Cmplx-Metric-Info, Cmplx-Static-Attr, Cmplx-Recursion-Depth
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_DYN	from VMO: Label-String from Metric: Metric-Status, Measurement-Status, Metric-Id, Metric-Id-Ext, Unit-Code, Unit- Labelstring, Vmo-Source-List, Metric-Source-List, Msmt-Site-List, Body-Site- List, Metric-Calibration, Color, Measure- Mode, Measure-Period, Averaging-Period, Start-Time, Stop-Time, Metric-Info-Labelstring, Substance, Substance-Labelstring, Body-Site-List-Ext, Msmt-Site- List-Ext from ComplexMetric: Cmplx-Dyn-Attr

Attribute group	Attribute group ID	Group elements
Metric Observed Value Group (extensible attribute group)	MDC_ATTR_GRP_METRIC_VAL_OBS	from Metric: Metric-Id-Partition from ComplexMetric: Cmplx-Observed-Value

The following ASN.1 data type definitions apply:

```
--
-- Definitions for Cmplx-Metric-Info attribute
--
CmplxMetricInfo ::= SEQUENCE {
    max-mplex-obs           INT-U8,             -- maximum number of messages
                                                -- until all multiplexed elements are
                                                -- transmitted in the Metric
                                                -- Observed Value Group
    max-mplex-dyn           INT-U8,             -- maximum number of messages
                                                -- until all multiplexed elements are
                                                -- transmitted in the VMO
                                                -- Dynamic Context Group
    cm-elem-info-list       CmplxElemInfoList
}

CmplxElemInfoList ::= SEQUENCE OF CmplxElemInfo

CmplxElemInfo ::= SEQUENCE {
    class-id                OID-Type,
    max-inst                INT-U8,         -- number of objects from type class-id
    max-inst-comp           INT-U8,         -- number of compound objects from type class-id
    max-comp-no             INT-U8,         -- maximum number of elements within a compound
                                            -- object
    max-inst-mplex          INT-U8,         -- number of multiplexed objects within max-inst +
                                            -- max-inst-comp
    max-str-size            INT-U16         -- maximum string size
}

--
-- Cmplx-Observed-Value attribute, representing the hierarchy of contained Metric objects
--
CmplxObsValue ::= SEQUENCE {
    cm-obs-cnt              INT-U8,             -- sequence counter begins with 0
                                                -- when a multiplex period begins
    cm-obs-flags            CmplxFlags,
    cm-obs-elem-list        CmplxObsElemList
}

CmplxFlags ::= BITS-8 {
    cmplx-flag-reserved(0)  -- for future extensions
}

CmplxObsElemList ::= SEQUENCE OF CmplxObsElem

CmplxObsElem ::= SEQUENCE {
    cm-elem-idx             INT-U8,
    cm-obs-elem-flgs        CmplxObsElemFlags,
    attributes              AttributeList
}

CmplxObsElemFlags ::= BITS-8 {
```

```
        cm-obs-elem-flg-mplex(0),                   -- the element will be multiplexed
        cm-obs-elem-flg-is-setting(2),
        cm-obs-elem-flg-updt-episodic(4),
        cm-obs-elem-flg-msmt-noncontinuous(5)
    }

    --
    -- Cmplx-Dyn-Attr with the dynamic context data of the hierarchy of contained Metric objects
    --
    CmplxDynAttr ::= SEQUENCE {
        cm-dyn-cnt              INT-U8,             -- sequence counter begins with 0
                                                    -- when a multiplex period begins
        unused                  INT-U8,
        cm-dyn-elem-list        CmplxDynAttrElemList
    }

    CmplxDynAttrElemList ::= SEQUENCE OF CmplxDynAttrElem

    CmplxDynAttrElem ::= SEQUENCE {
        cm-elem-idx-1           INT-U8,             -- allows the definition, with cm-elem-idx-2, of a range
                                                    -- of elements where the dynamic attributes apply
        cm-elem-idx-2           INT-U8,
        attributes              AttributeList
    }

    --
    -- Cmplx-Static-Attr with the static context data of the hierarchy of contained Metric objects
    --
    CmplxStaticAttr ::= SEQUENCE {
        cm-static-elem-list     CmplxStaticAttrElemList
    }

    CmplxStaticAttrElemList ::= SEQUENCE OF CmplxStaticAttrElem

    CmplxStaticAttrElem ::= SEQUENCE {
        cm-elem-idx-1           INT-U8,             -- allows the 94definition, with cm-elem-idx-2, of a range
                                                    -- of elements where the static attributes apply
        cm-elem-idx-2           INT-U8,
        attributes              AttributeList       -- only static attributes as defined for metric
                                                    -- specialization are allowed here (i.e., no VMO or
                                                    -- Metric attributes)
    }
```

6.3.11.2 Behavior

The ComplexMetric class does not define any special methods.

6.3.11.3 Notifications

The ComplexMetric class does not generate any special notifications.

6.3.12 PM-Store class

Class: PM-Store
Description: "PM-Store objects provide long-term storage capabilities for metric data. A PM-Store contains a variable number of PM-Segment objects that can be accessed only through the PM-Store object."
Superclass: VMO
Subclasses: --

Name Binding: Handle
Registered As: MDC_MOC_VMO_PMSTORE

6.3.12.1 Attributes

The PM-Store class defines the attributes in Table 25.

Table 25—PM-Store class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Metric-Class	MDC_ATTR_ METRIC_CLASS	OID-Type	Class of stored metric(s).	M
Store-Sample-Algorithm	MDC_ATTR_ METRIC_STORE_ SAMPLE_ALG	StoSampleAlg	Method used to derive stored values from metric observed values.	O
Storage-Format	MDC_ATTR_ METRIC_STORE_ FORMAT	StorageFormat	Layout of stored data in PM-Segment objects.	M
Store-Capacity-Count	MDC_ATTR_ METRIC_STORE_ CAPAC_CNT	INT-U32	Maximum number of stored values.	O
Store-Usage-Count	MDC_ATTR_OP_ STAT	OperationalState	Actual number of stored values.	O
Sample-Period	MDC_ATTR_TIME_ PD_SAMP	RelativeTime	Used if values are sampled periodically.	C
Number-Of-Segments	MDC_ATTR_NUM_ SEG	INT-U16	Currently instantiated PM-Segment objects contained in the PM-Store object.	M

The PM-Store class defines in Table 26 the attribute groups or extensions to inherited attribute groups.

Table 26—PM-Store class attribute groups

Attribute group	Attribute group ID	Group elements
VMO Static Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_STATIC	from VMO: Type, Handle
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_DYN	from VMO: Label-String
PM-Store Attribute Group	MDC_ATTR_GRP_PMSTORE	from PM-Store: Metric-Class, Store-Sample-Algorithm, Storage-Format, Store-Capacity-Count, Store-Usage-Count, Sample-Period, Number-Of-Segments

The following ASN.1 data type definitions apply:

```
--
-- The storage format defines the structure of the Segment-Data attribute in all contained PM-Segment objects
-- 1..255: range for normative formats
-- 32768..65535: range for private formats
-- other: reserved
--
```

```
StorageFormat ::= INT-U16 {
      sto-t-nos(0),
      sto-t-gen(1),                 -- implies general format (i.e., PM-Segment, see 6.3.13)
      sto-t-nu-opt(2),              -- implies optimized Numeric object format
      sto-t-rtsa-opt(3)             -- implies optimized RealTimeSampleArray object format
}

--
-- Store-Sample-Algorithm attribute describes how samples are derived
--
StoSampleAlg ::= INT-U16 {
      st-alg-nos(0),
      st-alg-moving-average(1),
      st-alg-recursive(2),
      st-alg-min-pick(3),
      st-alg-max-pick(4),
      st-alg-median(5)
}
```

6.3.12.2 Behavior

The PM-Store class defines the methods in Table 27.

Table 27—PM-Store instance methods

Action	Mode	Action ID	Action parameter	Action result
Clear-Segments	Confirmed	MDC_ACT_SEG_CLEAR	SegmSelection	—
Get-Segments	Confirmed	MDC_ACT_SEG_GET	SegmSelection	SegmentAttrList
Get-Segment-Info	Confirmed	MDC_ACT_SEG_GET_INFO	SegmSelection	SegmentInfoList

The following ASN.1 data type definitions apply:

```
--
-- SegmSelection selects the PM-Segment objects that are subject to the method
--
SegmSelection ::= CHOICE {
      all-segments                  [1] INT-U16,      -- if this type is chosen to select all segments, the actual
                                                      -- contents of the field are “do not care” and should be 0
      segm-id-list                  [2] SegmIdList,
      abs-time-range                [3] AbsTimeRange
}

--
-- SegmIdList selects PM-Segment objects by ID
--
SegmIdList ::= SEQUENCE OF InstNumber

--
-- The time range allows selection of PM-Segment objects by time period
--
AbsTimeRange ::= SEQUENCE {
      from-time   AbsoluteTime,
      to-time     AbsoluteTime
}

--
```

```
-- Get-Segments method returns a list of PM-Segment attribute lists that include the Segment-Data attribute; the
-- instance number is used to identify each segment
--
SegmentAttrList ::= SEQUENCE OF SegmentAttr

SegmentAttr ::= SEQUENCE {
      seg-inst-no  InstNumber,
      seg-attr     AttributeList
}

--
-- Segment contents information as a result to the Get-Segment-Info returns all attributes of the PM-Segment
-- objects except the Segment-Data attribute; this is useful to get just information about the contents
--
SegmentInfoList ::= SEQUENCE OF SegmentInfo

SegmentInfo ::= SEQUENCE {
      seg-inst-no  InstNumber,
      seg-info     AttributeList
}
```

6.3.12.3 Notifications

The PM-Store class does not generate any special notifications.

6.3.13 PM-Segment class

Class: PM-Segment
Description: "A PM-Segment object represents a continuous time period in which a metric is stored without any changes of relevant metric context attributes (e.g., scales, labels). PM-Segment objects are contained in a PM-Store object and are not directly accessible by management services."
Superclass: Top
Subclasses: --
Name Binding: Instance Number (object not directly manageable; instance number unique within a single PM-Store instance)
Registered As: MDC_MOC_PM_SEGMENT

6.3.13.1 Attributes

The PM-Segment class defines the attributes in Table 28.

Table 28—PM-Segment class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Instance-Number	MDC_ATTR_ID_INSTNO	InstNumber		M
Metric-Id	MDC_ATTR_ID_PHYSIO	OID-Type	ID of stored metric (from VMO::Type or Metric-IdPartition partition).	M
Metric-Id-Ext	MDC_ATTR_ID_MSMT_EXT	ExtNomenRef	Dynamic identification of the metric in a different nomenclature or dictionary. Use of this attribute severely limits interoperability of applications.	O
Vmo-Global-Reference	MDC_ATTR_VMO_REF_GLB	GLB-HANDLE	Reference to stored Metric object.	O

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Segment-Start-Abs-Time	MDC_ATTR_TIME_START_SEG	AbsoluteTime	Start time of segment.	O
Segment-End-Abs-Time	MDC_ATTR_TIME_END_SEG	AbsoluteTime	End time of segment.	O
Segment-Usage-Count	MDC_ATTR_SEG_USAGE_CNT	INT-U32	Actual (i.e., current) number of stored values.	O
Segment-Data-Gen	MDC_ATTR_SEG_DATA_GEN	SegDataGen	Segment data stored in a format as specified in the PM-Store object. Exactly one of the following attributes shall be implemented for any PM-Segment object: Segment-Data-Gen, Segment-Data-Nu-Opt, Segment-Data-Rtsa-Opt.	C
Segment-Data-Nu-Opt	MDC_ATTR_SEG_DATA_NU_OPT	SegDataNuOpt	Segment data stored in a format as specified in the PM-Store object. Exactly one of the following attributes shall be implemented for any PM-Segment object: Segment-Data-Gen, Segment-Data-Nu-Opt, Segment-Data-Rtsa-Opt.	C
Segment-Data-Rtsa-Opt	MDC_ATTR_SEG_DATA_RTSA_OPT	SegDataRtsaOpt	Segment data stored in a format as specified in the PM-Store object. Exactly one of the following attributes shall be implemented for any PM-Segment object: Segment-Data-Gen, Segment-Data-Nu-Opt, Segment-Data-Rtsa-Opt.	C
Context Attributes	As defined for Metric derived objects	Any attribute from Metric-derived objects that is member of either the VMO Static Context Group or the VMO Dynamic Context Group	Metric context attributes are allowed in this object without container. Attributes are identified by their OID. This reference to attributes is an editorial convenience. There is no need to copy all attributes from the various objects to the PM-Segment object. Copying attributes is not a hidden form of inheritance.	

The PM-Segment class defines no attribute groups.

The following ASN.1 data type definitions apply:

```
--
-- General segment data format; each stored value is one attribute list
-- NOTE--This format may be very storage-intensive
--
SegDataGen ::= SEQUENCE OF AttributeList

--
-- Optimized Numeric object format for periodically acquired numerics; only the actual value is stored
--
SegDataNuOpt ::= SEQUENCE OF FLOAT-Type

--
-- Optimized RealTimeSampleArray object format; a consecutive array of samples
--
```

SegDataRtsaOpt ::= OCTET STRING

6.3.13.2 Behavior

The PM-Segment class does not define any special methods.

6.3.13.3 Notifications

The PM-Segment class does not generate any special notifications.

6.4 Alert package

6.4.1 Alert class

Class: Alert
Description: "An Alert object stands for the status of a simple alert condition check. As such, it represents a single alert condition only. The alert condition can be either a physiological alarm or a technical alarm condition of a related object (e.g., MDS, VMD, Metric)."
Superclass: VMO
Subclasses: --
Name Binding: Handle
Registered As: MDC_MOC_VMO_AL

6.4.1.1 Attributes

The Alert class defines the attributes in Table 29.

Table 29—Alert class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Alert-Condition	MDC_ATTR_AL_COND	AlertCondition		M
Limit-Specification	MDC_ATTR_AL_LIMIT	LimitSpecEntry	Relevant for limit alarms only.	O
Vmo-Reference	MDC_ATTR_VMO_REF	HANDLE		O

NOTE—The VMO inherited type field defines if the Alert represents a technical or physiological alarm.

The Alert class defines in Table 30 the attribute groups or extensions to inherited attribute groups.

Table 30—Alert class attribute groups

Attribute group	Attribute group ID	Group elements
VMO Static Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_STATIC	from VMO: Type, Handle from Alert: Vmo-Reference
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_DYN	from VMO: Label-String from Alert: Limit-Specification

Attribute group	Attribute group ID	Group elements
Alert Group	MDC_ATTR_GRP_AL	from Alert: Alert-Condition

The following ASN.1 data type definitions apply:

```
--
-- Alert-Condition attribute is the status output of the process that is detecting the alert
--
AlertCondition ::= SEQUENCE {
    obj-reference           HANDLE,
    controls                AlertControls,
    alert-flags             AlertFlags,         -- supporting flags
    alert-source            OID-Type,           -- from metric or object-oriented nomenclature
                                                -- partition
    alert-code              OID-Type,           -- from events nomenclature partition
    alert-type              AlertType,          -- defines type and severity of condition
    alert-info-id           PrivateOid,         -- specific information can be appended; 0 if not used
    alert-info              ANY DEFINED BY  alert-info-id
}
```

NOTE—The alert-code value is a code that comes from the events nomenclature partition. Entries (i.e., codes) in this partition are even. The last bit of the code is used to determine whether the alert-source comes from the metric nomenclature partition or the object-oriented nomenclature partition. If the last bit of the alert-code value is 0, the alert-source comes from the metric nomenclature partition. If the last bit is 1 (1 is added to the base code in the events nomenclature), the alert-source comes from the object-oriented nomenclature partition.

```
--
-- Alert controls define flags to communicate status information relevant for alert processor; this structure is
-- reused in the AlertStatus class
--
AlertControls ::= BITS-16 {
    ac-obj-off(0),              -- the object supervised by the alert is off
    ac-chan-off(1),             -- channel is off
    ac-all-obj-al-off(3),       -- all alerts supervising the referenced objects are off
    ac-alert-off(4),            -- this alert supervisor process is off
    ac-alert-muted(5)           -- this alert is temporarily muted by the user (e.g. on ventilators to allow
                                -- physiotherapy or suction)
}

--
-- Alert flags give additional support information on how to process the condition; this structure is used by
-- AlertStatus objects as well
--
AlertFlags ::= BITS-16 {
    local-audible(1),           -- indicates that the condition is audible at the local system
    remote-audible(2),          -- condition can be audible at remote (i.e., not suppressed)
    visual-latching(3),         -- visible latching of the condition is allowed
    audible-latching(4),        -- audio latching of the condition is allowed
    derived(6),
    record-inhibit(8)           -- do not start alarm recording
}

--
-- Alert type is used to distinguish severity of technical and physiological alarms
--
AlertType ::= INT-U16 {
    no-alert(0),
    low-pri-t-al(1),            -- low-priority technical alarm
    med-pri-t-al(2),            -- medium-priority technical alarm
```

```
        hi-pri-t-al(4),             -- high-priority technical alarm
        low-pri-p-al(256),          -- awareness condition
        med-pri-p-al(512),          -- prompt response required (i.e., abnormal condition)
        hi-pri-p-al(1024)           -- immediate response required (i.e., emergency condition)
    }

    --
    -- Limit-Specification attribute specifies the supervised limit range
    --
    LimitSpecEntry ::= SEQUENCE {
        object-handle               HANDLE,
        al-source-id                OID-Type,       -- typically the metric ID of the measurement
        unit-code                   OID-Type,       -- from DIM partition
        lim-al-stat                 CurLimAlStat,   -- see 6.6.8.1 for definition
        lim-al-val                  CurLimAlVal     -- see 6.6.8.1 for definition
                                                    -- for definition
    }
```

6.4.1.2 Behavior

The Alert class does not define any special methods.

6.4.1.3 Notifications

The Alert class does not generate any special notifications.

6.4.2 AlertStatus class

Class: AlertStatus
Description: "An AlertStatus object represents the output of an alarm process that considers all alert conditions in a scope that spans one or more objects. In contrast to an Alert object, an AlertStatus object collects all alert conditions related to a VMD object hierarchy or multiple alert conditions related to an MDS object and provides this information in list-structured attributes."
Superclass: VMO
Subclasses: --
Name Binding: Handle
Registered As: MDC_MOC_VMO_AL_STAT

6.4.2.1 Attributes

The AlertStatus class defines the attributes in Table 31.

Table 31—AlertStatus class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Alert-Capab-List	MDC_ATTR_AL_STAT_AL_C_LIST	AlertCapabList	Capabilities of the AlertStatus object.	M
Tech-Alert-List	MDC_ATTR_ALSTAT_AL_T_LIST	AlertList	List of technical alert information.	O
Physio-Alert-List	MDC_ATTR_ALSTAT_AL_P_LIST	AlertList	List of physiological alert information.	O
Limit-Spec-List	MDC_ATTR_AL_LIMIT_SPEC_LIST	LimitSpecList	List of limit alarm ranges.	O

The AlertStatus class defines in Table 32 the attribute groups or extensions to inherited attribute groups.

Table 32—AlertStatus class attribute groups

Attribute group	Attribute group ID	Group elements
VMO Static Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_STATIC	from VMO: Type, Handle from AlertStatus: Alert-Capab-List
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_DYN	from VMO: Label-String from AlertStatus: Limit-Spec-List
AlertStatus Group	MDC_ATTR_GRP_AL_STAT	from AlertStatus: Tech-Alert-List, Physio-Alert-List

The following ASN.1 data type definitions apply:

```
--
-- The alert list is used to communicate alert conditions derived by the AlertStatus object
--
AlertList ::= SEQUENCE OF AlertEntry

AlertEntry ::= SEQUENCE {
    obj-reference           HANDLE,
    instance                InstNumber,     -- to support multiple alarms of one object
    controls                AlertControls,
    alert-source            OID-Type,       -- from metric or object-oriented nomenclature
                                            -- partition
    alert-code              OID-Type,       -- from alerts nomenclature partition
    alert-type              AlertType,
    alert-info-id           PrivateOid,
    alert-info              ANY DEFINED BY  alert-info-id
}
```

NOTE— The alert-code value is a code that comes from the events nomenclature partition. Entries (i.e., codes) in this partition are even. The last bit of the code is used to determine whether the alert-source comes from the metric nomenclature partition or the object-oriented nomenclature partition. If the last bit of the alert-code value is 0, the alert-source comes from the metric nomenclature partition. If the last bit is 1 (1 is added to the base code in the events nomenclature), the alert-source comes from the object-oriented nomenclature partition.

```
--
-- AlertStatus object provides a capability list with entries for each supervised object in its scope
--
AlertCapabList ::= SEQUENCE OF AlertCapabEntry

AlertCapabEntry ::= SEQUENCE {
    obj-reference           HANDLE,
    obj-class               OID-Type,
    alert-group             OID-Type,       -- allows grouping of Alert objects so that a processor
                                            -- can select to display only one from a given group
                                            -- (metric ID)
    al-rep-flags            AlertRepFlags,  -- defines how multiple alarms are communicated
    max-t-severity          AlertType,      -- most severe technical alarm
    max-t-obj-al            INT-U16,        -- maximum number of parallel technical alarms for
                                            -- this object
    max-p-severity          AlertType,      -- most severe physiological alarm
    max-p-obj-al            INT-U16         -- maximum number of parallel physiological alarms
```

```
                                                    -- for this object
    }

    AlertRepFlags ::= BITS-16 {
          dyn-inst-contents(1),
          rep-all-inst(2)
    }

    --
    -- Limit-Spec-List attributed specifies the supervised limit ranges
    --
    LimitSpecList ::= SEQUENCE OF LimitSpecEntry
```

6.4.2.2 Behavior

The AlertStatus class does not define any special methods.

6.4.2.3 Notifications

The AlertStatus class does not generate any special notifications.

6.4.3 AlertMonitor class

Class: AlertMonitor
Description: "An AlertMonitor object represents the output of a medical device system or system alert processor. As such, it represents the overall device or system alert condition and provides a list of all alert conditions of the system in its scope. This list includes global state information and individual alarm state information that allows the implementation of a safety standard-compliant alarm display on a remote system."
Superclass: VMO
Subclasses: --
Name Binding: Handle
Registered As: MDC_MOC_VMO_AL_MON

6.4.3.1 Attributes

The AlertMonitor class defines the attributes in Table 33.

Table 33—AlertMonitor class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Device-Alert-Condition	MDC_ATTR_DEV_AL_COND	DevAlertCondition	Global device alert status.	M
Device-P-Alarm-List	MDC_ATTR_AL_MON_P_AL_LIST	DevAlarmList	Active physiological alarm list.	M
Device-T-Alarm-List	MDC_ATTR_AL_MON_T_AL_LIST	DevAlarmList	Active technical alarm list.	M
Device-Sup-Alarm-List	MDC_ATTR_AL_MON_S_AL_LIST	DevAlarmList	Suppressed physiological alarm list.	O
Limit-Spec-List	MDC_ATTR_AL_LIMIT_SPEC_LIST	LimitSpecList	List of limit alarm ranges.	O
Suspension-Period	MDC_ATTR_TIME_PD_AL_SUSP	RelativeTime	Remaining alarm suspend time.	O

The AlertMonitor class defines in Table 34 the attribute groups or extensions to inherited attribute groups.

Table 34—AlertMonitor class attribute groups

Attribute group	Attribute group ID	Group elements
VMO Static Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_STATIC	from VMO: Type, Handle
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_DYN	from VMO: Label-String from AlertMonitor: Limit-Spec-List
AlertMonitor Group	MDC_ATTR_GRP_AL_MON	from AlertMonitor : Device-Alert-Condition, Device-P-Alarm-List, Device-T-Alarm-List, Device-Sup-Alarm-List, Suspension-Period

The following ASN.1 data type definitions apply:

```
--
-- Device-Alert-Condition attribute describes the global MDS alarm status
--
DevAlertCondition ::= SEQUENCE {
    device-alert-state      AlertState,
    al-stat-chg-cnt         AlStatChgCnt,    -- change counter marks state or active alerts change
    max-p-alarm             AlertType,
    max-t-alarm             AlertType,
    max-aud-alarm           AlertType        -- maximum severity of audible alarm
}

AlertState ::= BITS-16 {
    al-inhibited(0),        -- off
    al-suspended(1),        -- alert(ing) inactivated temporarily; alert condition acknowledged
    al-latched(2),          -- specific alert is latched (or AlMon latches alert conditions)
    al-silenced-reset(3),   -- (transition only); alert indication stopped, but alarming re-enabled
    al-dev-in-test-mode(5), -- device is in test mode; the alarms are not real patient alarms
    al-dev-in-standby-mode(6), -- device is in standby mode
    al-dev-in-demo-mode(7)  -- device is in demonstration mode, the alarms are not real patient alarms
}

AlStatChgCnt ::= SEQUENCE {
    al-new-chg-cnt          INT-U8,          -- Device-Alert-Condition attribute changed
    al-stack-chg-cnt        INT-U8           -- alert stack (active alarm list attributes) changed
}

--
-- Device alarm list
--
DevAlarmList ::= SEQUENCE OF DevAlarmEntry

DevAlarmEntry ::= SEQUENCE {
    al-source               OID-Type,        -- from metric or object-oriented nomenclature
                                             -- partition
    al-code                 OID-Type,        -- from events nomenclature partition
    al-type                 AlertType,
    al-state                AlertState,
    object                  ManagedObjectId,
    alert-info-id           PrivateOid,
```

alert-info ANY DEFINED BY alert-info-id
}

NOTE— The al-code value is a code that comes from the events nomenclature partition. Entries (i.e., codes) in this partition are even. The last bit of the code is used to determine whether the al-source comes from the metric nomenclature partition or the object-oriented nomenclature partition. If the last bit of the al-code value is 0, the al-source comes from the metric nomenclature partition. If the last bit is 1 (1 is added to the base code in the events nomenclature), the al-source comes from the object-oriented nomenclature partition.

6.4.3.2 Behavior

The AlertMonitor class does not define any special methods.

6.4.3.3 Notifications

The AlertMonitor class does not generate any special notifications.

6.5 System package

6.5.1 VMS class

Class: VMS
Description: "The VMS class is the abstract base class for all System Package classes in this model. It provides consistent naming and identification of system-related objects."
Superclass: Top
Subclasses: MDS
Name Binding: Handle
Registered As: MDC_MOC_VMS

6.5.1.1 Attributes

The VMS class defines the attributes in Table 35.

Table 35—VMS class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Handle	MDC_ATTR_ID_HANDLE	HANDLE	Name binding attribute.	M
System-Type	MDC_ATTR_SYS_TYPE	TYPE	Examples: ventilator, monitor as defined in nomenclature	M
System-Model	MDC_ATTR_ID_MODEL	SystemModel	Model describes manufacturer and model number.	C
System-Id	MDC_ATTR_SYS_ID	OCTET STRING	Unique system ID, e.g., serial number.	C
Compatibility-Id	MDC_ATTR_ID_COMPAT	INT-U32	For manufacturer use.	O
Nomenclature-Version	MDC_ATTR_NOM_VERS	NomenclatureVersion	Version ISO/IEEE 11073 part 10101 used by the system.	C
System-Capability	MDC_ATTR_SYS_CAPAB	SystemCapability	Set of supported features; system specific.	O
System-Specification	MDC_ATTR_SYS_SPECN	SystemSpec	Defines functional components.	O
Production-Specification	MDC_ATTR_ID_PROD_SPECN	ProductionSpec	Component revisions, serial numbers, etc.	O

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Udi	MDC_ATTR_ID_UDI	UdiSpec	This attribute defines the UDI(s) under which this product is listed with an authority such as the US FDA.	O
Ext-Obj-Relations	MDC_ATTR_EXT_OBJ_REL ATION	ExtObjRelationList	Relations to objects that are not defined in the DIM.	O

The conditional (C) VMS attributes are mandatory for the top-level VMS object instance (i.e., root object instance of the containment tree); they are optional otherwise.

The VMS class defines in Table 36 the attribute groups or extensions to inherited attribute groups.

Table 36—VMS class attribute groups

Attribute group	Attribute group ID	Group elements
System Identification Attribute Group (extensible attribute group)	MDC_ATTR_GRP_SYS_ID	from VMS: System-Type, System-Model, System-Id, Compatibility-Id, Nomenclature-Version
System Application Attribute Group (extensible attribute group)	MDC_ATTR_GRP_SYS_APPL	from VMS: System-Capability, System-Specification
System Production Attribute Group (extensible attribute group)	MDC_ATTR_GRP_SYS_PROD	from VMS: Production-Specification, Udi

NOTE—The Relationship Attribute Group is not shown again in the definitions of derived classes.

The following ASN.1 data type definitions apply:

```
--
-- System-Model attribute is specified by manufacturer and manufacturer-specific model number
--
SystemModel ::= SEQUENCE {
    manufacturer                OCTET STRING,
    model-number                OCTET STRING
}

--
-- System-Capability attribute is a top-level specification of implemented functions; (the following is an example
-- only)
--
SystemCapability ::= BITS-32 {
    sc-multiple-context(0),         -- indicates that system uses multiple naming contexts
    sc-dyn-configuration(1),        -- containment tree changes dynamically
    sc-dyn-scanner-create(2),       -- system allows host to create Scanner objects dynamically
    sc-auto-init-scan-list(3),      -- CfgScanner object supports automatic scan list initialization
    sc-auto-updt-scan-list(4)       -- CfgScanner object supports automatic scan list update
}

--
-- System-Specification attribute allows specific entries for system functional components
--
SystemSpec ::= SEQUENCE OF SystemSpecEntry
```

```
SystemSpecEntry ::= SEQUENCE {
    component-capab-id      PrivateOid,
    component-spec          ANY DEFINED BY  component-capab-id
}

--
-- Production-Specification attribute deals with serial numbers, part numbers, revisions, etc.; note that a device
-- may have multiple components so the Production-Specification attribute should be a printable string defining
-- the component and the "number"
--
ProductionSpec ::= SEQUENCE OF ProdSpecEntry

ProdSpecEntry ::= SEQUENCE {
    spec-type               ProdSpecType,
    component-id            PrivateOid,
    prod-spec               OCTET STRING
}

ProdSpecType ::= INT-U16 {
    unspecified(0),
    serial-number(1),
    part-number(2),
    hw-revision(3),
    sw-revision(4),
    fw-revision(5),
    protocol-revision(6),
    prod-spec-gmdn(7),      -- Global Medical Device Nomenclature[7]
    device-identifier(8),   -- component of UDi.
    manufacture-date(9),    -- component of UDI. ISO 8601 formatted date with optional hour.
    expiry-date(10),        -- component of UDI. ISO 8601 formatted date with optional hour.
    lot-number(11)          -- component of UDI.
}

--
-- UdiSpec supports the encoding of a unique device identitifier such as the US FDA UDI.
--
UdiSpec ::= SEQUENCE OF UdiSpecification

UdiSpecification ::= SEQUENCE {
    udi-authority           UdiAuthority,
    udi-issuer              UdiIssuer,
    udi-label               UdiLabel
}

--

-- Identifies the regulatory agency having jurisdiction over the creation of the UDI, such as the US FDA.
-- The ITU OID for US FDA is 2.16.840.1.113883.3.24
UdiAuthority ::= ITU-OID;


-- Organization that is charged with issuing UDIs for devices.
-- For example, the US FDA issuers include:
-- GS1(2.51), HIBCC(1.0.15961.13.10) and ICCBBA(2.16.840.1.113883.6.18)
-- An ITU OID length of 0 should be used if the issuer is unknown or not recorded in the
-- device.
UdiIssuer ::= ITU-OID;
```

[7] The Global Medical Device Nomenclature (GMDN) is based on ISO 15225 and was developed under the auspices of CEN TC257 SC1

```
--
-- An ITU OID can be used as an unique identifier for an organization
-- ITU OID trees are maintained by registration authorities with ITU-T and ISO at the top;
-- HL7 manages an OID tree;
-- see http://oid-info.com/index.htm
--
ITU-OID ::= OCTET STRING

-- Full text label on the human readable string
-- Matches the human readable UDI string that is part of the labeling of the device
UdiLabel ::= OCTET STRING

--
-- Nomenclature-Version attribute contains a part of the major version field (i.e., basic compatibility) and the
-- minor version (used to identified the latest used update); the major version part is coded as a bit field so that
-- systems supporting multiple versions can negotiate the version used within an association
--
NomenclatureVersion ::= SEQUENCE {
    nom-major-version       NomenclatureMajorVersion,    -- major version identifier
    nom-minor-version       INT-U16                      -- counter to identify minor updates
}

NomenclatureMajorVersion ::= BITS-16 {
    majorVersion1(0),
    majorVersion2(1),
    majorVersion3(2),
    majorVersion4(3)
}
```

6.5.1.2 Behavior

The VMS class does not define any special methods.

6.5.1.3 Notifications

The VMS class does not generate any special notifications.

6.5.2 MDS class

Class:	MDS
Description:	"The MDS class is an abstraction of a device that provides medical information in the form of instances of classes that are defined in the Medical Package of the DIM. Further specializations of this class are used to represent differences in complexity and scope. As a base class, the MDS object cannot be instantiated."
Superclass:	VMS
Subclasses:	CompositeMultipleBedMDS, SinglePatientMDS
Name Binding:	Handle
Registered As:	MDC_MOC_VMS_MDS

6.5.2.1 Attributes

The MDS class defines the attributes in Table 37.

Table 37—MDS class attributes

Attribute name	Attribute ID[a]	Attribute type	Remark	Qualifier[b]
Mds-Status	MDC_ATTR_VMS_MDS_STAT	MDSStatus	Device state according to MDS FSM.	C
Bed-Label	MDC_ATTR_ID_BED_LABEL	OCTET STRING	Printable string identifying system location.	O
Soft-Id	MDC_ATTR_ID_SOFT	OCTET STRING	Settable, e.g., hospital inventory number.	O
Operating-Mode	MDC_ATTR_MODE_OP	PrivateOid		O
Application-Area	MDC_ATTR_AREA_APPL	ApplicationArea		O
Patient-Type	MDC_ATTR_PT_TYPE	PatientType	May control algorithms, see 7.10.1.1 for definition of type.	O[xc]
Date-And-Time	MDC_ATTR_TIME_ABS	AbsoluteTime	MDS maintains device time.	O
Relative-Time	MDC_ATTR_TIME_REL	RelativeTime		O
Hires-Relative-Time	MDC_ATTR_TIME_REL_HI_RES	HighResRelativeTime		O
Power-Status	MDC_ATTR_POWER_STAT	PowerStatus	Either onBattery or onMains.	O[bd]
Altitude	MDC_ATTR_ALTITUDE	INT-I16	Meters above or below sea level.	O
Battery-Level	MDC_ATTR_VALBAT_CHARGE	INT-U16	In % of capacity; undefined if value > 100.	O
Remaining-Battery-Time	MDC_ATTR_TIME_BATT_REMAIN	BatMeasure	See 7.5.9.1 for the definition of type; minutes are the recommended measurement unit.	O
Line-Frequency	MDC_ATTR_LINE_FREQ	LineFrequency	Frequency of mains; implicitly in hertz (typically either 50 Hz or 60 Hz)	O
Association-Invoke-Id	MDC_ATTR_ID_ASSOC_NO	INT-U16	Counter for number of associations on a given communications port; Incremented with each association control service element (ACSE) association	O
Locale	MDC_ATTR_LOCALE	Locale	Defines charset and language of printable string attributes in this MDS and contained objects. Contained MDS or VMD objects may define different Locale attributes for their scope. The top-level MDS shall support this attribute.	C

[a] Some of the VMS and MDS attributes need to be exchanged during association as user information fields in the ACSE protocol. The ACSE user information fields should contain only VMS or MDS attributes.

[b] The conditional (C) MDS attributes are mandatory for the top-level MDS object instance (i.e., root object instance of the containment tree); they are optional otherwise.
[c]If MDS supports the PatientDemographics object, the MDS object should not contain this attribute to avoid conflicts
[d]If more information for battery-powered devices about the battery is needed (especially if the battery is manageable), a special Battery object should be used.

The MDS class defines in Table 38 the attribute groups or extensions to inherited attribute groups.

Table 38—MDS class attribute groups

Attribute group	Attribute group ID	Group elements
System Identification Attribute Group (extensible attribute group)	MDC_ATTR_GRP_SYS_ID	from VMS: System-Type, System-Model, System-Id, Compatibility-Id, Nomenclature-Version from MDS: Soft-Id, Association-Invoke-Id, Locale
System Application Attribute Group (extensible attribute group)	MDC_ATTR_GRP_SYS_APPL	from VMS: System-Capability, System-Specification from MDS: Mds-Status, Bed-Label, Operating-Mode, Application-Area, Patient-Type, Date-And-Time, Relative-Time, Hires-Relative-Time, Power-Status, Altitude, Battery-Level, Remaining-Battery-Time, Line-Frequency
System Production Attribute Group (extensible attribute group)	MDC_ATTR_GRP_SYS_PROD	from VMS: Production-Specification

The following ASN.1 data type definitions apply:

```
--
-- MDS state of one association/connection according to FSM
--
MDSStatus ::= INT-U16 {
      disconnected(0),
      unassociated(1),
      associating(2),
      associated(3),
      configuring(4),
      configured(5),
      operating(6),
      re-initializing(7),
      terminating(8),
      disassociating(9),
      disassociated(10),
      re-configuring(11)
}

--
-- Application-Area attribute
--
ApplicationArea ::= INT-U16 {
      area-unspec(0),
      area-operating-room(1),
```

```
        area-intensive-care(2)
    }

    --
    -- Power-Status attribute defines whether the device is on battery or on mains; upper bits define the charging state
    --
    PowerStatus ::= BITS-16 {
        onMains(0),                 -- Power source is mains, feed instance agnostic.
        onBattery(1),               -- Power source is battery, battery instance agnostic.
        deviceOff(4),               -- Device is off, condition detected by an external device interface
                                    -- or other means.
        standby(5),                 -- Device is in standby mode, typically at a lower power consumption
                                    -- level and typically after the power on self-test. Device is
                                    -- immediately available for use.
        chargingFull(8),            -- Indicates fully charged, battery instance agnostic.
        chargingTrickle(9),         -- Indicates at least one charging in trickle state, battery instance
                                    -- agnostic.
        chargingOff(10)             -- Indicates all charging is off, battery instance agnostic.
    }

    --
    -- Line-Frequency attribute
    --
    LineFrequency ::= INT-U16 {
        line-f-unspec(0),
        line-f-50hz(1),
        line-f-60hz(2)
    }
```

6.5.2.2 Behavior

The MDS class defines the methods in Table 39.

Table 39—MDS instance methods

Action	Mode	Action ID	Action parameter	Action result
Mds-Set-Status	Confirmed	MDC_ACT_SET_MDS_STATE	MdsSetStateInvoke	MdsSetStateResult

The following ASN.1 data type definitions apply:

```
    --
    -- MDS-Set-State method permits modification of the state of the MDS state machine e.g., to trigger a reset (if
    -- supported by a device)
    -- NOTE--Usage of the authorization type is implementation-specific, especially given the security and operational
    -- coordination issues involved
    --
    MdsSetStateInvoke ::= SEQUENCE {
        new-state                   MDSStatus,
        authorization               INT-U32
    }

    MdsSetStateResult ::= MDSStatus
```

6.5.2.3 Notifications

The MDS class defines the events in Table 40.

Table 40—MDS events

Event	Mode	Event ID	Event parameter	Event result
System-Error	Unconfirmed	MDC_NOTI_SYS_ERR	MdsErrorInfo	—
Mds-Create-Notification	Confirmed	MDC_NOTI_MDS_CREAT	MdsCreateInfo	—
Mds-Attribute-Update	Confirmed	MDC_NOTI_MDS_ATTR_UPDT	MdsAttributeChangeInfo	—

The following ASN.1 data type definitions apply:

```
--
-- System-Error notification in case of system errors
--
MdsErrorInfo ::= SEQUENCE {
      error-type    PrivateOid,
      error-info    ANY DEFINED BY  error-type
}

--
-- Mds-Create-Notification event is sent after association is established
--
MdsCreateInfo ::= SEQUENCE {
      class-id                  ManagedObjectId,
      attribute-list            AttributeList     -- attributes from the System Identification Attribute
                                                  -- Group and System Application Attribute Group
}

--
-- MDS may report changes of attribute values
--
MdsAttributeChangeInfo ::= AttributeList
```

6.5.3 CompositeMultipleBedMDS class

Class: CompositeMultipleBedMDS
Description: "The CompositeMultipleBedMDS class represents a device that contains multiple MDS objects at multiple locations (i.e., multiple beds)."
Superclass: MDS
Subclasses: --
Name Binding: Handle
Registered As: MDC_MOC_VMS_MDS_COMPOS_MULTI_BED

The CompositeMultipleBedMDS class does not define any attributes, methods, or notifications.

6.5.4 SinglePatientMDS class

Class: SinglePatientMDS
Description: "The SinglePatientMDS is an abstract class that models a medical device that is associated with a single patient."
Superclass: MDS
Subclasses: CompositeSingleBedMDS, SingleSystemMDS
Name Binding: Handle
Registered As: N/A

The SinglePatientMDS class does not define any attributes, methods, or notifications.

6.5.5 CompositeSingleBedMDS class

Class:	CompositeSingleBedMDS
Description:	"The CompositeSingleBedMDS class models a device that contains one or more SimpleMDS or HydraMDS objects at one location (i.e., a bed)."
Superclass:	SinglePatientMDS
Subclasses:	--
Name Binding:	Handle
Registered As:	MDC_MOC_VMS_MDS_COMPOS_SINGLE_BED

The CompositeSingleBedMDS class does not define any attributes, methods, or notifications.

6.5.6 SingleSystemMDS class

Class:	SingleSystemMDS
Description:	"The SingleSystemMDS class models a medical device that contains VMD objects and that may be contained by a CompositeMDS object."
Superclass:	SinglePatientMDS
Subclasses:	SimpleMDS, HydraMDS
Name Binding:	Handle
Registered As:	N/A

The SingleSystemMDS class does not define any attributes, methods, or notifications.

6.5.7 SimpleMDS class

Class:	SimpleMDS
Description:	"The SimpleMDS class models a medical device that contains a single VMD instance only (i.e., a single-purpose device)."
Superclass:	SingleSystemMDS
Subclasses:	--
Name Binding:	Handle
Registered As:	MDC_MOC_VMS_MDS_SIMP

The SimpleMDS class does not define any attributes, methods, or notifications.

6.5.8 HydraMDS class

Class:	HydraMDS
Description:	"The HydraMDS class models a device that contains multiple VMD instances (i.e., a multipurpose device)."
Superclass:	SingleSystemMDS
Subclasses:	--
Name Binding:	Handle
Registered As:	MDC_MOC_VMS_MDS_HYD

The HydraMDS class does not define any attributes, methods, or notifications.

6.5.9 Log class

Class:	Log
Description:	"A Log object is a storage container for important local system notifications and events. Further specializations define specific event types that are stored in the Log object. The Log class is an abstract base class and cannot be instantiated."

Superclass: Top
Subclasses: EventLog
Name Binding: Handle
Registered As: MDC_MOC_LOG

6.5.9.1 Attributes

The Log class defines the attributes in Table 41.

Table 41—Log class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Handle	MDC_ATTR_ID_HANDLE	HANDLE	Name binding attribute.	M
Max-Log-Entries	MDC_ATTR_LOG_ENTRIES_MAX	INT-U32	Maximum capacity of the Log object; GET service used to retrieve this attribute.	M
Current-Log-Entries	MDC_ATTR_LOG_ENTRIES_CURR	INT-U32	Current used capacity of the Log object; GET service used to retrieve this attribute.	M
Log-Change-Count	MDC_ATTR_LOG_CHANGE_COUNT	INT-U16	Incremented when log contents change.	O

NOTE—It is assumed that Log object entries are indexed from 0 to the Current-Log-Entries attribute value.

The Log class does not define any attribute groups.

6.5.9.2 Behavior

The Log class defines the methods in Table 42.

Table 42—Log instance methods

Action	Mode	Action ID	Action parameter	Action result
Clear-Log	Confirmed	MDC_ACT_CLEAR_LOG	ClearLogRangeInvoke	ClearLogRangeResult (optional)

The following ASN.1 data type definitions apply:

```
--
-- Range of log entries to be deleted; if the parameter is not appended to the Clear-Log method, the complete log
-- shall be cleared unconditionally
--
ClearLogRangeInvoke ::= SEQUENCE {
      clear-log-option          ClearLogOptions,
      log-change-count          INT-U16,                    -- 0 if unconditional clear
      from-log-entry-index      INT-U32,
      to-log-entry-index        INT-U32
}

ClearLogRangeResult ::= SEQUENCE {
      clear-log-result          ClearLogResult,
      log-change-count          INT-U16,        -- current change count after clear
      from-log-entry-index      INT-U32,        -- do not care if not successful
      to-log-entry-index        INT-U32,        -- do not care if not successful
      current-log-entries       INT-U32         -- updated number of entries in the log
```

```
}

--
-- Options that control the clear command
--
ClearLogOptions ::= BITS-16 {
    log-clear-if-unchanged(1)       -- only perform this action if the log has not been changed; in other words,
                                    -- the evlog-change-count in the request is still current
}

--
-- Result of the clear log function
--
ClearLogResult ::= INT-U16 {
    log-range-cleared(0),                   -- successful operation
    log-changed-clear-error(1),             -- the change count was wrong (i.e., log has been
                                            -- modified)
    log-change-counter-not-supported(2)     -- log does not support a change counter
}
```

6.5.9.3 Notifications

The Log class does not generate any special notifications.

6.5.10 EventLog class

Class: EventLog
Description: "An EventLog object is a general Log object that stores system events in a free-text or in a binary representation."
Superclass: Log
Subclasses: --
Name Binding: Handle
Registered As: MDC_MOC_LOG_EVENT

6.5.10.1 Attributes

The EventLog class defines the attributes in Table 43.

Table 43—EventLog class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Type	MDC_ATTR_ID_TYPE	TYPE	Further specification of log entry format.	O
Event-Log-Entry-List	MDC_ATTR_EVENT_LOG_ENTRY_LIST	EventLogEntryList	Event entries; can be retrieved with GET service.	M
Event-Log-Info	MDC_ATTR_EVENT_LOG_INFO	EventLogInfo	Static and dynamic specifications.	O

The EventLog class does not define any attribute groups.

The following ASN.1 data type definitions apply:

```
--
-- Event-Log-Entry-List attribute
--
EventLogEntryList ::= SEQUENCE OF EventLogEntry
```

```
EventLogEntry ::= SEQUENCE {
    entry-number            INT-U32,                -- entry counter independent of the
                                                    -- index number that is used for
                                                    -- access
    abs-time                AbsoluteTime,           -- event time
    event-entry             OCTET STRING            -- free text or binary event
                                                    -- information; structure defined by
                                                    -- the Type attribute
}

--
-- Event-Log-Info attribute
-- Bits 0 to 15 are reserved for static information; bits 16 to 31 are dynamically updated to reflect log status
-- changes
-- If this attribute is not present, all bits are implicitly assumed 0
--
EventLogInfo ::= BITS-32 {
    ev-log-clear-range-sup(0),  -- supports to clear specified ranges (not just the entire log)
    ev-log-get-act-sup(1),      -- supports retrieving individual entries using the Get-Event-Log method
                                -- (not just a simple GET service)
    ev-log-binary-entries(8),   -- log entries are binary, not free text
    ev-log-full(16),            -- log is full; cleared as soon as the log contains at least 1 free entry as a
                                -- result of a clear action
    ev-log-wrap-detect(17)      -- set when the log is full and the first old entry is overwritten; cleared as
                                -- soon as the log contains at least 1 free entry as a result of a clear action
}
```

6.5.10.2 Behavior

The EventLog class defines the methods in Table 44.

Table 44—EventLog instance methods

Action	Mode	Action ID	Action parameter	Action result
Get-Event-Log-Entries	Confirmed	MDC_ACT_GET_EVENT_LOG_ENTRIES	GetEventLogEntryInvoke	GetEventLogEntryResult

The following ASN.1 data type definitions apply:

```
--
-- Range of log entries to be retrieved
--
GetEventLogEntryInvoke ::= SEQUENCE {
    from-log-entry-index    INT-U32,
    to-log-entry-index      INT-U32
}

--
-- Reply containing the requested entries; depending on agent restrictions, the reply may contain only a part of the
-- requested entries; this situation must be checked by the manager
--
GetEventLogEntryResult ::= SEQUENCE {
    log-change-count        INT-U16,                -- current log change counter (0 if
                                                    -- not supported)
    from-log-entry-index    INT-U32,
    to-log-entry-index      INT-U32,
    entry-list              EventLogEntryList
}
```

6.5.10.3 Notifications

The EventLog class does not generate any special notifications.

6.5.11 Battery class

Class: Battery
Description: "For battery-powered devices, some battery information is contained in the MDS object in the form of attributes. If the battery subsystem is either capable of providing more information (i.e., smart battery) or manageable, then a special Battery object is provided."
Superclass: Top
Subclasses: --
Name Binding: Handle
Registered As: MDC_MOC_BATT

6.5.11.1 Attributes

The Battery class defines the attributes in Table 45.

Table 45—Battery class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Handle	MDC_ATTR_ID_ HANDLE	HANDLE	Name binding attribute.	M
Battery-Status	MDC_ATTR_BATT_ STAT	BatteryStatus		M
Production-Specification	MDC_ATTR_ID_ PROD_SPECN	ProductionSpec	A smart battery system may have a serial number or version.	O
Capacity-Remaining	MDC_ATTR_ CAPAC_BATT_ REMAIN	BatMeasure	Remaining capacity at current load (e.g., in milliAmperehours).	O
Capacity-Full-Charge	MDC_ATTR_ CAPAC_BATT_ FULL	BatMeasure	Battery capacity after a full charge.	O
Capacity-Specified	MDC_ATTR_ CAPAC_BATT_ SPECN	BatMeasure	Specified capacity of new battery.	O
Remaining-Battery-Time	MDC_ATTR_TIME_ BATT_REMAIN	BatMeasure		O
Voltage	MDC_ATTR_BATT_ VOLTAGE	BatMeasure	Present battery voltage	O
Voltage-Specified	MDC_ATTR_BATT_ VOLTAGE_SPECN	BatMeasure	Specified battery voltage.	O
Current	MDC_ATTR_BATT_ CURR	BatMeasure	Present current delivered by/to battery; negative if battery is charge.	O
Battery-Temperature	MDC_ATTR_TEMP_ BATT	BatMeasure		O
Charge-Cycles	MDC_ATTR_BATT_ CHARGE_CYCLES	INT-U32	Number of charge/discharge cycles.	O

The Battery class defines in Table 46 the attribute groups or extensions to inherited attribute groups.

Table 46—Battery class attribute groups

Attribute group	Attribute group ID	Group elements
Battery Attribute Group	MDC_ATTR_GRP_BATT	from Battery: Handle, Battery-Status, Production-Specification, Capacity-Remaining, Capacity-Full-Charge, Capacity-Specified, Remaining-Battery-Time, Voltage, Voltage-Specified, Current, Battery-Temperature, Charge-Cycles

The following ASN.1 data type definitions apply:

```
--
-- Battery Status bit field
--
BatteryStatus ::= BITS-16 {
      batt-discharged(0),
      batt-full(1),                    -- > 95% of capacity
      batt-discharging(2),
      batt-chargingFull(8),
      batt-chargingTrickle(9),
      batt-malfunction(12),
      batt-needs-conditioning(13)      -- battery needs conditioning
}

--
-- All measures about the battery are values with their dimensions
--
BatMeasure ::= SEQUENCE {
      value       FLOAT-Type,
      unit        OID-Type             -- from dimensions nomenclature partition
}
```

6.5.11.2 Behavior

The Battery class does not define any special methods.

6.5.11.3 Notifications

The Battery class does not generate any special notifications.

6.5.12 Clock class

Class: Clock
Description: "The Clock class provides additional date/time capability and status information beyond the information provided by the ~~basic MDS class's~~ time-related attributes. The Clock class does not imply any specific hardware or software support."
Superclass: Top
Subclasses: --
Name Binding: Handle
Registered As: MDC_MOC_CLOCK

6.5.12.1 Attributes

The Clock class defines the attributes in Table 47.

Table 47—Clock class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Handle	MDC_ATTR_ID_HANDLE	HANDLE	Name binding attribute.	M
Time-Support	MDC_ATTR_TIME_SUPPORT	TimeSupport	Indicates the time services provided by the device.	M
Date-Time-Status	MDC_ATTR_DATE_TIME_STATUS	DateTimeStatus	General information about the functioning of time-support services. Mandatory if remote sync services are supported by the device [e.g., Simple Network Time Protocol (SNTP)]; optional otherwise.	C
Date-And-Time	MDC_ATTR_TIME_ABS	AbsoluteTime	Current date/time setting.	O
ISO-Date-And-Time	MDC_ATTR_TIME_ABS_ISO	AbsoluteTimeISO	Date and time string formatted in accordance with ISO 8601; provides for international coordinated universal time (UTC) coordination. Attribute is in wide use by computing systems; however, it is ASCIIbased and thus less efficient than absolute time.	O
Relative-Time	MDC_ATTR_TIME_REL	RelativeTime	Relative time (in 8 kHz ticks).	O
Hires-Relative-Time	MDC_ATTR_TIME_REL_HI_RES	HighResRelativeTime	High-resolution relative time (in 1 MHz ticks).	O
Ext-Time-Stamp-List	MDC_ATTR_TIME_STAMP_LIST_EXT	ExtTimeStampList	Extended timestamp (which may be used individually elsewhere in data structures).	O
Absolute-Relative-Sync	MDC_ATTR_TIME_ABS_REL_SYNC	AbsoluteRelativeTimeSync	Provides a means of correlating between absolute time and relative time values.[a]	O
Time-Zone	MDC_ATTR_TIME_ZONE	UTCTimeZone	Identifies the UTC local time zone offset [from Greenwich mean time (GMT)] and label.	O
Daylight-Savings-Transition	MDC_ATTR_TIME_DAYLIGHT_SAVINGS_TRANS	DaylightSavingsTransition	Provides the settings for the next daylight savings time transition.	O
Cumulative-Leap-Seconds	MDC_ATTR_CUM_LEAP_SECONDS	INT-U32	Cumulative leap-seconds relative to January 1, 1900, 00:00:00.00. Format is +nn. For the entire year 2001, this value is +32.[b]	O

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Next-Leap-Seconds	MDC_ATTR_ NEXT_LEAP_ SECOND	LeapSecondsTransition	Specifies the settings for when the next leap-seconds transition shall occur and the next value.	O

[a]This attribute is periodically updated internally (e.g., once per minute) and thus does not reflect the actual time when read (e.g., using a GET service). The error between relative and absolute time should be as small as possible given system limitations (e.g., an atomic operation should be used if possible). The attribute should be updated frequently enough to minimize the error between the reported mapping and should be updated at a minimum of every 6 days, namely when the relative time would roll over to 0.
[b]When subtracted from SNTP seconds yields UTC seconds.

The Clock class defines in Table 48 the attribute groups or extensions to inherited attribute groups.

Table 48—Clock class attribute groups

Attribute group	Attribute group ID	Group elements
Clock Attribute Group	MDC_ATTR_GRP_CLOCK	from Clock: Handle, Time-Support, Date-Time-Status, Date-And-Time, ISO-Date-And-Time, Relative-Time, Hires-Relative-Time, Ext-Time-Stamp-List, Absolute-Relative-Sync, Time-Zone, Daylight-Savings-Transition, Cumulative-Leap-Seconds, Next-Leap-Seconds

The following ASN.1 data type definitions apply:

```
--
-- Time-Support attribute provides general information about time-related services that are provided by the device
-- Some of this information could be determined by examining the presence/absence of various attributes in a
-- containment tree; however, its presence here simplifies time management for device managers
--
-- NOTES
-- 1-- If remote date/time synchronization is supported (e.g., SNTP), then either the Date-And-Time or ISO-Date-
--       And-Time attribute must also be supported
-- 2-- If the device is also a server of time information (e.g., an SNTP server), this fact should be indicated in the
--       time protocol IDs
--
TimeSupport ::= SEQUENCE {
      time-capability              TimeCapability,              -- Flags indicating general time
                                                                -- support
      relative-resolution          INT-U32,                     -- Time between actual ticks in
                                                                -- microseconds; set to
                                                                -- 0xFFFFFFFF if not defined or
                                                                -- specified
      time-protocols               TimeProtocolIdList           -- List of external time protocols
                                                                -- supported (e.g., SNTP)
}

TimeProtocolIdList ::= SEQUENCE OF TimeProtocolId
```

NOTE—The relative-resolution type provides a means of correlating the 8 kHz frequency reported by the relative time value to the device's time sources from which it is being derived. For example, if the device's timer updates at 100 Hz or 18.2 Hz [as is the case in older personal computers (PCs)], then the resolution and accuracy of the relative time would reflect this time source resolution and accuracy.

```
--
-- Time capability
--
TimeCapability ::= BITS-32 {
      time-capab-real-time-clock(0),          -- the device includes hardware support for time
                                              -- (including battery power)
      time-capab-ebww(1),                     -- time can be set locally/manually ("eyeball and
                                              -- wristwatch" or "EBWW")
      time-capab-leap-second-aware(2),        -- supports adjustment of time for leap-seconds (SNTP-
                                              -- related)
      time-capab-time-zone-aware(3),          -- supports time zone-related attributes
      time-capab-internal-only(4),            -- date/time is used only internally to the device; not
                                              -- displayed to operator
      time-capab-time-displayed(5),           -- date/time can be displayed continually on the device
                                              -- versus in a menu
      time-capab-patient-care(6),             -- date/time is used in critical patient care
                                              -- algorithms/protocols
      time-capab-rtsa-time-sync-annotations(7),  -- timestamp annotations supported for real-time
                                              -- waveform data (RealTimeSampleArray objects)
      time-capab-rtsa-time-sync-high-precision(8), -- RealTimeSampleArray objects support attributes
                                              -- for high precision sample timestamps
      time-capab-set-time-action-sup(16),     -- Clock object supports the set time action
      time-capab-set-time-zone-action-sup(17), -- Clock object supports the set time zone action
      time-capab-set-leap-sec-action-sup(18), -- Clock object supports the set leap-seconds action
      time-capab-set-time-iso-sup(19)         -- Clock object supports the set time ISO action
}

--
-- Time protocol ID indicates the time protocols that are supported/used by the device
-- OID-Types from the infrastructure nomenclature partition
--
TimeProtocolId ::= OID-Type

--
-- Timestamp ID (e.g., for SNTP timestamps)
-- OID-Types from the infrastructure nomenclature partition
--
TimeStampId ::= OID-Type

--
-- Extended timestamp (e.g., SNTP timestamp value)
--
ExtTimeStamp ::= SEQUENCE {
      time-stamp-id           TimeStampId,
      time-stamp              ANY DEFINED BY  time-stamp-id
}

ExtTimeStampList ::= SEQUENCE OF ExtTimeStamp

--
-- Date-Time-Status attribute defines the current/active usage status for date and time in the device
--
DateTimeStatus ::= SEQUENCE {
      usage-status            DateTimeUsage,  -- flags indicating dynamic time usage
      clock-last-set          AbsoluteTime,   -- time the absolute time was last set
      clock-accuracy          FLOAT-Type,     -- decimal number indicating the accuracy or
                                              -- maximum error of the absolute time relative to a
```

```
                                                    -- primary reference clock source (in seconds)
        active-sync-protocol        TimeProtocolId  -- protocol that is actively being used for time
                                                    -- synchronization
}
```

NOTE 1—If a time synchronization protocol is used that changes the time and date at a high frequency, the clock-last-set type value should be updated at a lower periodicity (e.g., once every 10 min or once an hour), so that communications bandwidth is not consumed unnecessarily.

NOTE 2—In systems where time synchronization is not used (i.e., EBWW is source), the clock-accuracy type should be initialized to 2 or 3 min when the clock time is set and should be incremented periodically to reflect drift from an absolute external reference source. If NTP is used, clock-accuracy type initialization is equivalent to Root Dispersion + ½ Root Delay.

```
--
-- Date/time usage flags indicate dynamic usage status for date and time in the device; no bits set indicates
-- unknown/indeterminate status
--
DateTimeUsage ::= BITS-16 {
    dt-use-remote-sync(0),      -- date/time is synchronized to an external source
    dt-use-operator-set(1),     -- date/time set by operator (i.e., EBWW)
    dt-use-rtc-synced(2),       -- date/time in the RTC has been synchronized to a remote time source
    dt-use-critical-use(3),     -- date/time is actively being used in care delivery algorithms/protocols
    dt-use-displayed(4)         -- date/time is actively being displayed to the operator
}

--
-- ISO-Date-and-Time attribute is an ASCII string that can provide additional information beyond the basic
-- date/time setting (e.g., UTC offset or device-local time zone indication); this attribute can be set using the SET
-- service (as can the Date-And-Time attribute)
-- Note that if both AbsoluteTime and AbsoluteTimeISO types are concurrently supported, they shall reflect the
-- same time (relative to their accuracy and resolution limitations)
-- Although not mandatory, it is highly recommended that all optional fields be included in the string
-- To simplify processing, the following constraints shall apply:
-- (a) Only complete representations shall be used
-- (b) Only extended formats shall be used
-- (c) "Week date" and “ordinal "day of the year" representations shall not be used; only calendar dates
-- (d) Decimal fractions shall be used only for partial seconds (e.g., not fractional hours)
-- (e) Per ISO 8601:2000(E), the representation of decimal fractions shall be in accordance with Section 5.3.1.3
-- (f) If known, UTC shall be communicated as the offset between local and
-- GMT/UTC time. If the local time zone offset is unknown, i.e. NTP is used, then UTC timeshall be specified
-- ~~using Zulu (or Z) format~~by adding a time offset of -0000. +0000 shall be used to indicate GMT.
-- (g) Specification of time intervals and recurring periods is beyond the use of this data type and shall require a
-- definition of a new data type if used (e.g., ISOTimeInterval ::= OCTET STRING); for example: November 24,
-- 2001, 3:45:32.65 P.M. in San Diego, California, USA, shall be represented by the following string: 2001-11-
-- 24T15:45:32,65-08:00
--
-- ASCII text string that adheres to ISO 8601 format
--
AbsoluteTimeISO ::= OCTET STRING

--
-- SNTPTimeStamp, a 64-bit timestamp value that is provided by an SNTP time synchronization service
--
SNTPTimeStamp ::= SEQUENCE {
    seconds     INT-U32,        -- Seconds since January 1, 1900 00:00
    fraction    INT-U32         -- Binary fraction of a second
}

--
-- Absolute-Relative-Sync attribute provides a means for correlating relative timestamps to the device's date/time
-- setting
```

```
-- NOTE--This attribute needs to be updated only periodically to account for drift between the various time
-- sources (e.g., once a minute)
--
AbsoluteRelativeTimeSync ::= SEQUENCE {
    absolute-time-mark          AbsoluteTime,           -- use of this data type limits
                                                        -- resolution to 1/100 second
    relative-time-mark          RelativeTime,           -- resolution limited by 125 µs tick
                                                        -- and resolution/accuracy settings
                                                        -- for relative time service
    relative-rollovers          INT-U16,                -- number of times the relative time
                                                        -- has "rolled over" from its
                                                        -- maximum value to 0
                                                        -- NOTE--The relative time will
                                                        -- roll over every 6.2 days
    hires-time-mark             HighResRelativeTime,    -- defaults to 0x00000000 if not
                                                        -- supported
    ext-time-marks              ExtTimeStampList        -- list is empty if no extended
                                                        -- timestamps are supported
}

--
-- Time-Zone attribute supports time zone information for UTC
--
UTCTimeZone ::= SEQUENCE {
    time-zone-offset-hours      INT-I8,                 -- device's local time zone (i.e., at
                                                        -- the point of care), relative to
                                                        -- UTC
                                                        -- format is +hh for time zones east
                                                        -- of GMT and -hh for locations
                                                        -- west of GMT
    time-zone-offset-minutes    INT-U8,                 -- minutes offset from GMT (if
                                                        -- specified); format conventions
                                                        -- are the same as the conventions
                                                        -- for hours, only they are not
                                                        -- signed (shall always be a positive
                                                        -- value); default is NULL
    time-zone-label             OCTET STRING            -- device's local time zone label,
                                                        -- e.g., PST or PDT; see device's
                                                        -- Locale attribute for string
                                                        -- encoding
}

--
-- Daylight-Savings-Transition attribute specifies the settings for the next transition to/from daylight savings time
--
DaylightSavingsTransition ::= SEQUENCE {
    transition-date             AbsoluteTime,   -- device's local date/time when the daylight savings
                                                -- transition will occur
    next-offset                 UTCTimeZone     -- new local time zone offset and label after transition
                                                -- date
                                                -- NOTE--May be same as previous value
}

--
-- Next-Leap-Seconds attribute specifies the settings for the next leap-seconds transition
--
LeapSecondsTransition ::= SEQUENCE {
    transition-date             Date,           -- device's local date when the transition will occur;
                                                -- adjustment occurs at the end (i.e., 23:59:59Z) of the
                                                -- specified date
    next-cum-leap-seconds       INT-U32         -- next cumulative leap-seconds value (see Cumulative-
                                                -- Leap-Seconds in Clock class attributes table)
```

```
                                        -- NOTE--May be same as previous value
}
```

6.5.12.2 Behavior

The Clock class defines the methods in Table 49.

Table 49—Clock instance methods

Action	Mode	Action ID	Action parameter	Action result
Set-Time	Confirmed	MDC_ACT_SET_TIME	SetTimeInvoke	—
Set-Time-Zone	Confirmed	MDC_ACT_SET_TIME_ZONE	SetTimeZoneInvoke	—
Set-Leap-Seconds	Confirmed	MDC_ACT_SET_LEAP_SECONDS	SetLeapSecondsInvoke	—
Set-Time-ISO	Confirmed	MDC_ACT_SET_TIME_ISO	AbsoluteTimeISO	—

When setting the time with either Set-Time or Set-Time-ISO methods, all supported absolute timestamp attributes (i.e., Date-and-Time, ISO-Date-and-Time and possibly Ext-Time-Stamp-List) shall be updated consistently.

The following ASN.1 data type definitions apply:

```
--
-- Date/time to be set
--
SetTimeInvoke ::= SEQUENCE {
      date-time    AbsoluteTime,
      accuracy     FLOAT-Type      -- accounts for manually set time (e.g., 2 min error); value is defined in
                                   -- seconds
}

--
-- Time zone information to be set
--
SetTimeZoneInvoke ::= SEQUENCE {
      time-zone                    UTCTimeZone,                -- current time zone to be used by
                                                               -- device
      next-time-zone               DaylightSavingsTransition   -- information for the next
                                                               -- transition to/from daylight
                                                               -- savings time
}

--
-- Cumulative leap-seconds information to be set
--
SetLeapSecondsInvoke ::= SEQUENCE {
      leap-seconds-cum             INT-I32,                    -- cumulative leap-seconds, which
                                                               -- when subtracted from S/NTP
                                                               -- seconds yields UTC seconds
      next-leap-seconds            LeapSecondsTransition       -- date of transition from previous
                                                               -- to new cumulative leap-second
                                                               -- value + new value
}
```

6.5.12.3 Notifications

The Clock class defines the events in Table 50.

Table 50—Clock events

Event	Mode	Event ID	Event parameter	Event result
Clock-Date-Time-Status- Changed	Unconfirmed	MDC_NOTI_DATE_TIME_CHANGED	ClockStatusUpdateInfo	—

The following ASN.1 data type definitions apply:

```
--
-- Clock status update information is sent, for example, when the relative time setting rolls over to 0 or when the
-- time is changed by the device operator
--
ClockStatusUpdateInfo ::= SEQUENCE {
    date-time-status        DateTimeStatus,             -- current clock/time usage status
    time-sync               AbsoluteRelativeTimeSync    -- current time synchronization
                                                        -- values
}
```

6.6 Control package

6.6.1 SCO class

Class: SCO
Description: "The SCO is responsible for managing all remote control capabilities that are supported by a medical device. The SCO is the primary access point for invoking remote control functions. It contains all Operation objects and provides a means for transaction processing. All Operation object invoke commands shall be done through the SCO."
Superclass: VMO
Subclasses: --
Name Binding: Handle (VMO inherited)
Registered As: MDC_MOC_CNTRL_SCO

6.6.1.1 Attributes

The SCO class defines the attributes in Table 51.

Table 51—SCO class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Sco-Capability	MDC_ATTR_SCO_CAPAB	ScoCapability	Static option flag field.	M
Sco-Help-Text-String	MDC_ATTR_SCO_HELP_TEXT_STRING	OCTET STRING	Help text.	O
Vmo-Reference	MDC_ATTR_VMO_REF	HANDLE	Reference to controlled item, if not the VMD.	O
Activity-Indicator	MDC_ATTR_INDIC_ACTIV	ScoActivityIndicator	Can be set by remote system to give feedback that system is under remote control.	O
Lock-State	MDC_ATTR_STAT_LOCK	AdministrativeState	If locked, no operation can be invoked.	M

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Invoke-Cookie	MDC_ATTR_ID_ INVOK_COOKIE	INT-U32	Transaction ID assigned by invoke command.	M

The SCO class defines in Table 52 the attribute groups or extensions to inherited attribute groups.

Table 52—SCO class attribute groups

Attribute group	Attribute group ID	Group elements
VMO Static Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_STATIC	from VMO: Type, Handle from SCO: Sco-Capability, Sco-Help-Text-String
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_DYN	from VMO: Label-String from SCO: Vmo-Reference, Activity-Indicator
SCO Transaction Group	MDC_ATTR_GRP_SCO_TRANSACTION	from SCO: Lock-State, Invoke-Cookie

The following ASN.1 data type definitions apply:

```
--
-- Activity-Indicator attribute can be set by a remote system to indicate that remote control is active
--
ScoActivityIndicator ::= INT-U16 {
      act-ind-off(0),
      act-ind-on(1),
      act-ind-blinking(2)
}

--
-- Sco-Capability bits
--
ScoCapability ::= BITS-16 {
      act-indicator(0),          -- supports activity indicator
      sco-locks(1),              -- at least one operation sets the SCO lock flag
      sco-ctxt-help(8)           -- SCO supports context-dependent dynamic help
}
```

6.6.1.2 Behavior

In addition to the SET service, which can be used to modify the Activity-Indicator attribute, the SCO defines the methods in Table 53.

Table 53—SCO instance methods

Action	Mode	Action ID	Action parameter	Action result
Operation-Invoke	Confirmed	MDC_ACT_SCO_OP_ INVOKE	OperationInvoke	OperationInvokeResult
Get-Ctxt-Help	Confirmed	MDC_ACT_GET_ CTXT_HELP	CtxtHelpRequest	CtxtHelpResult

The following ASN.1 data type definitions apply:

```
--
-- Operation-Invoke method has an additional security mechanism
--
OperationInvoke ::= SEQUENCE {
    checksum                INT-I16,        -- 16-bit twos complement
    invoke-cookie           INT-U32,        -- arbitrary ID mirrored back in resulting updates
    op-elem-list            OpInvokeList
}
```

NOTE—If check-summing is not used, the checksum field shall be 0. If calculated checksum is 0, the checksum field shall be –1. Checksum calculation is the 16-bit twos-complement sum of 16-bit words in the message starting at the address after the checksum field.

```
OpInvokeList ::= SEQUENCE OF OpInvokeElement

OpInvokeElement ::= SEQUENCE {
    op-class-id             OID-Type,       -- from object-oriented nomenclature partition
    op-instance-no          InstNumber,
    op-mod-type             OpModType,
    attributes              AttributeList
}

OpModType ::= INT-U16 {
    op-replace(0),              -- normally replace value of virtual attribute
    op-setToDefault(3),         -- set to default value if this is supported
    op-invokeAction(10),        -- needed for singular action type of operations
    op-invokeActionWithArgs(15) -- action with arguments
}

--
-- Result confirms reception (and execution) of operations
-- Updated attributes are communicated via normal update method (e.g., scanner) to avoid inconsistencies
--
OperationInvokeResult ::= SEQUENCE {
    invoke-cookie           INT-U32,
    result                  OpInvResult
}

OpInvResult ::= INT-U16 {
    op-successful(0),
    op-failure(1)
}

--
-- The following types allow the retrieval of dynamic help information that is SCO or Operation object context-
-- dependent (i.e., state-dependent)
--
CtxtHelpRequest ::= SEQUENCE {
    type                    OID-Type,       -- either Operation object class ID or SCO class ID
    op-instance-no          InstNumber      -- operation instance number (0 if SCO is addressed)
}

CtxtHelpResult ::= SEQUENCE {
    type                    OID-Type,       -- either Operation object class ID or SCO class ID
    op-instance-no          InstNumber,
    hold-time               RelativeTime,   -- how long to display help; 0 if not applicable
    help                    CtxtHelp
}
```

```
CtxtHelp ::= CHOICE {
    text-string   [1] OCTET STRING,
    oid           [8] OID-Type
}
```

6.6.1.3 Notifications

The SCO class defines the events in Table 54.

Table 54—SCO events

Event	Mode	Event ID	Event parameter	Event result
SCO-Operating-Request	Confirmed/Unconfirmed	MDC_NOTI_SCO_OP_ REQ	ScoOperReqSpec	—
SCO-Operation-Invoke- Error	Confirmed/Unconfirmed	MDC_NOTI_SCO_OP_ INVOK_ERR	ScoOperInvokeError	—

The following ASN.1 data type definitions apply:

```
--
-- An operating request may append additional information
--
ScoOperReqSpec ::= SEQUENCE {
    op-req-id     PrivateOid,          -- device-or manufacturer-specific
    op-req-info   ANY DEFINED BY  op-req-id
}

--
-- SCO-Operation-Invoke-Error notification
--
ScoOperInvokeError ::= SEQUENCE {
    invoke-cookie                 INT-U32,
    op-error                      OpErrorType,
    failed-operation-list         InstNumberList
}

OpErrorType ::= INT-U16 {
    op-err-unspec(0),
    checksum-error(1),
    sco-lock-violation(2),
    unknown-operation(3),
    invalid-value(4),
    invalid-mod-type(5)
}

InstNumberList ::= SEQUENCE OF InstNumber
```

6.6.2 Operation class

Class: Operation
Description: "The Operation class is the abstract base class for classes that represent remote controllable items."
Superclass: Top
Subclasses: ActivateOperation, LimitAlertOperation, SelectItemOperation, SetRangeOperation, SetStringOperation, SetValueOperation, ToggleFlagOperation
Name Binding: Instance-Number (not directly accessible by object management services; unique within a single SCO instance)
Registered As: MDC_MOC_CNTRL_OP

6.6.2.1 Attributes

The Operation class defines the attributes in Table 55.

Table 55—Operation class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Instance-Number	MDC_ATTR_ID_INSTNO	InstNumber	Unique within SCO for operation identification.	M
Operation-Spec	MDC_ATTR_OP_SPEC	OperSpec	Structure defining operation types and properties.	M
Operation-Text-Strings	MDC_ATTR_OP_TEXT_STRING	OperTextStrings	Static description of operation.	O
Operation-Text-Strings-Dyn	MDC_ATTR_OP_TEXT_STRING_DYN	OperTextStrings	Dynamic description of operation.	O
Vmo-Reference	MDC_ATTR_VMO_REF	HANDLE	Reference to an object.	O
Operational-State	MDC_ATTR_OP_STAT	OperationalState	Specifies whether operation is accessible.	O

The Operation class defines in Table 56 the attribute groups or extensions to inherited attribute groups.

Table 56—Operation class attribute groups

Attribute group	Attribute group ID	Group elements
Operation Static Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_STATIC_CTXT	from Operation: Operation-Spec, Operation-Text-Strings
Operation Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_DYN_CTXT	from Operation: Vmo-Reference, Operational-State

The following ASN.1 data type definitions apply:

```
--
-- Operation-Spec attribute indicates what this operation really does
--
OperSpec ::= SEQUENCE {
    vattr-id    OID-Type,       -- ID of the virtual attribute that is changed by operation
    op-target   OID-Type,       -- from metric or object-oriented nomenclature partition
    options     OpOptions,      -- special options
    level       OpLevel,        -- range of importance
    grouping    OpGrouping      -- to describe relations between operations
}
```

NOTE—The vattr-id code comes from the virtual attribute nomenclature partition. Entries (i.e., codes) in this partition are even. The last bit of the code is used to define from which nomenclature partition the op-target code comes. If the last bit is 0, the op-target code comes from the metric nomenclature partition. If the last bit is 1 (1 is added to the base code in the virtual attribute nomenclature), the op-target code comes from the object-oriented nomenclature partition.

```
--
-- Operation texts
--
OperTextStrings ::= SEQUENCE {
```

```
        label       OCTET STRING,                   -- the label string indicates the meaning of the
                                                    -- operation
        help        OCTET STRING,                   -- the help string may contain additional help for the
                                                    -- user
        confirm     OCTET STRING                    -- the confirm string is shown by manager to a user to
                                                    -- reconfirm the operation (e.g., "do you really want to
                                                    -- shut down?")
    }

    --
    -- Operation options
    --
    OpOptions ::= BITS-16 {
        needs-confirmation(0),
        supports-default(1),            -- a default value is supported for the virtual attributes
        sets-sco-lock(2),               -- needs transaction processing to avoid side effects
        is-setting(3),                  -- value preserved over system power fail
        op-dependency(6),               -- operation has dependencies to others (always set if sets-sco-lock bit is set)
        op-auto-repeat(7),              -- supports auto repeat
        op-ctxt-help(8)                 -- provides context-dependent help via SCO action
    }

    --
    -- Level
    --
    OpLevel ::= BITS-16 {
        op-level-basic(0),              -- a normal operation
        op-level-advanced(1),           -- an advanced operation
        op-level-professional(2),
        op-item-normal(8),              -- operation modifies a normal user item
        op-item-config(9),              -- operation modifies a configuration item
        op-item-service(10)             -- operation modifies a service item (not used by regular operator)
    }

    --
    -- Field for grouping operations (i.e., defines logical relations); can be used to organize operations in a useful
    -- sequence on an operator interface (i.e., display)
    --
    OpGrouping ::= SEQUENCE {
        group       INT-U8,
        priority    INT-U8
    }
```

6.6.2.2 Behavior

The Operation class does not define any special methods.

6.6.2.3 Notifications

The Operation class does not generate any special notifications.

6.6.3 SelectItemOperation class

Class:	SelectItemOperation
Description:	"The SelectItemOperation class allows selection of one item out of a given list. The list can have different types."
Superclass:	Operation
Subclasses:	--
Name Binding:	Instance-Number (not directly accessible by object management services)
Registered As:	MDC_MOC_CNTRL_OP_SEL_IT

6.6.3.1 Attributes

The SelectItemOperation class defines the attributes in Table 57.

Table 57—SelectItemOperation class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Selected-Item-Index	MDC_ATTR_INDEX_SEL	INT-U16	Index of current selection.	M
Nom-Partition	MDC_ATTR_ID_NOM_PARTITION	NomPartition	If entries in list are OIDs, specifies the nomenclature partition that is used.	C
Select-List	MDC_ATTR_LIST_SEL	SelectList	List of possible choices.	M

The SelectItemOperation class defines in Table 58 the attribute groups or extensions to inherited attribute groups.

Table 58—SelectItemOperation class attribute groups

Attribute group	Attribute group ID	Group elements
Operation Static Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_STATIC_CTXT	from Operation: Operation-Spec, Operation-Text-Strings from SelectItemOperation: Nom-Partition
Operation Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_DYN_CTXT	from Operation: Vmo-Reference, Operational-State from SelectItemOperation: Selected-Item-Index, Select-List

The following ASN.1 data type definitions apply:

```
--
-- Select-List attribute defines valid selections
--
SelectList ::= CHOICE {
      oid-list      [1] OID-TypeList,
      value-list    [3] FLOAT-TypeList,
      value-u-list  [4] SelectUValueEntryList,
      string-list   [5] OCTET-STRING-List
}

OID-TypeList ::= SEQUENCE OF OID-Type

FLOAT-TypeList ::= SEQUENCE OF FLOAT-Type

SelectUValueEntryList ::= SEQUENCE OF SelectUValueEntry

--
-- Value with a unit/dimension code
--
SelectUValueEntry ::= SEQUENCE {
      value         FLOAT-Type,
      m-units       OID-Type            -- from dimensions nomenclature partition
}
```

OCTET-STRING-List ::= SEQUENCE OF OCTET STRING

6.6.3.2 Behavior

The SelectItemOperation class does not define any special methods.

6.6.3.3 Notifications

The SelectItemOperation class does not generate any special notifications.

6.6.4 SetValueOperation class

Class: SetValueOperation
Description: "The SetValueOperation class allows the system to adjust a value within a given range with a given resolution."
Superclass: Operation
Subclasses: --
Name Binding: Instance-Number (not directly accessible by object management services)
Registered As: MDC_MOC_CNTRL_OP_SEL_VAL

6.6.4.1 Attributes

The SetValueOperation class defines the attributes in Table 59.

Table 59—SetValueOperation class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Current-Value	MDC_ATTR_VAL_CURR	FLOAT-Type	Current value.	M
Set-Value-Range	MDC_ATTR_VAL_RANGE	OpSetValueRange	Range of legal values.	M
Step-Width	MDC_ATTR_VAL_STEP_WIDTH	OpValStepWidth	Allowed step width.	O
Unit-Code	MDC_ATTR_UNIT_CODE	OID-Type	From dimensions nomenclature partition.	O

The SetValueOperation class defines in Table 60 the attribute groups or extensions to inherited attribute groups.

Table 60—SetValueOperation class attribute groups

Attribute group	Attribute group ID	Group elements
Operation Static Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_STATIC_CTXT	from Operation: Operation-Spec, Operation-Text-Strings
Operation Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_DYN_CTXT	from Operation: Vmo-Reference, Operational-State from SetValueOperation: Current-Value, Set-Value-Range, Step-Width, Unit-Code

The following ASN.1 data type definitions apply:

--

```
-- Set-Value-Range attribute defines range and minimum resolution
--
OpSetValueRange ::= SEQUENCE {
      minimum    FLOAT-Type,
      maximum    FLOAT-Type,
      resolution    FLOAT-Type
}

--
-- Step-Width attribute is an ordered (in ascending order) array of ranges and corresponding minimum step
-- widths; the lower edge is defined in the minimum value of the range specification
--
OpValStepWidth ::= SEQUENCE OF StepWidthEntry

StepWidthEntry ::= SEQUENCE {
      upper-edge  FLOAT-Type,
      step-width   FLOAT-Type
}
```

6.6.4.2 Behavior

The SetValueOperation class does not define any special methods.

6.6.4.3 Notifications

The SetValueOperation class does not generate any special notifications.

6.6.5 SetStringOperation class

Class: SetStringOperation
Description: "The SetStringOperation class is used to set the contents of a string type virtual attribute."
Superclass: Operation
Subclasses: --
Name Binding: Instance-Number (not directly accessible by object management services)
Registered As: MDC_MOC_CNTRL_OP_SET_STRING

6.6.5.1 Attributes

The SetStringOperation class defines the attributes in Table 61.

Table 61—SetStringOperation class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Current-String	MDC_ATTR_ STRING_CURR	OCTET STRING	Current value of the string type virtual attribute.	C[a]
Set-String-Spec	MDC_ATTR_SET_ STRING_SPEC	SetStringSpec	Properties of the string type virtual attribute.	M

[a]The Current-String attribute is out of the scope of this standard if the setstr-hidden-val flag is set in the specification attribute; it is mandatory otherwise.

The SetStringOperation class defines in Table 62 the attribute groups or extensions to inherited attribute groups.

Table 62—SetStringOperation class attribute groups

Attribute group	Attribute group ID	Group elements
Operation Static Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_STATIC_CTXT	from Operation: Operation-Spec, Operation-Text-Strings
Operation Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_DYN_CTXT	from Operation: Vmo-Reference, Operational-State from SetStringOperation: Current-String, Set-String-Spec

The following ASN.1 data type definitions apply:

```
--
-- Set-String-Spec attribute
--
SetStringSpec ::= SEQUENCE {
      max-str-len INT-U16,          -- maximum supported string length
      char-size   INT-U16,          -- character size in bits, e.g. 7, 8 or 16
      set-str-opt SetStrOpt         -- special option bits
}

--
-- Options for the string
--
SetStrOpt ::= BITS-16 {
      setstr-null-terminated(0),    -- string is terminated with NULL character
      setstr-displayable(1),        -- string is displayable
      setstr-var-length(2),         -- string has variable length (up to maximum)
      setstr-hidden-val(3)          -- actual contents is hidden, e.g., for password entry
}
```

6.6.5.2 Behavior

The SetStringOperation class does not define any special methods.

6.6.5.3 Notifications

The SetStringOperation class does not generate any special notifications.

6.6.6 ToggleFlagOperation class

Class: ToggleFlagOperation
Description: "The ToggleFlagOperation class allows a switch to be toggled (with two states, e.g., on/off)."
Superclass: Operation
Subclasses: --
Name Binding: Instance-Number (not directly accessible by object management services)
Registered As: MDC_MOC_CNTRL_OP_TOG

6.6.6.1 Attributes

The ToggleFlagOperation class defines the attributes in Table 63.

Table 63—ToggleFlagOperation class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Toggle-State	MDC_ATTR_STAT_OP_TOG	ToggleState	Current state of toggle.	M
Toggle-Label-Strings	MDC_ATTR_TOG_LABELS_STRING	ToggleLabelStrings		M

The ToggleFlagOperation class defines in Table 64 the attribute groups or extensions to inherited attribute groups.

Table 64—ToggleFlagOperation class attribute groups

Attribute group	Attribute group ID	Group elements
Operation Static Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_STATIC_CTXT	from Operation: Operation-Spec, Operation-Text-Strings
Operation Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_DYN_CTXT	from Operation: Vmo-Reference, Operational-State from ToggleFlagOperation: Toggle-State, Toggle-Label-Strings

The following ASN.1 data type definitions apply:

```
--
-- Toggle-State attribute
--
ToggleState ::= INT-U16 {
      tog-state0(0),
      tog-state1(1)
}

--
-- Each state has a label
--
ToggleLabelStrings ::= SEQUENCE {
      lbl-state0    OCTET STRING,
      lbl-state1    OCTET STRING
}
```

6.6.6.2 Behavior

The ToggleFlagOperation class does not define any special methods.

6.6.6.3 Notifications

The ToggleFlagOperation class does not generate any special notifications.

6.6.7 ActivateOperation class

Class: ActivateOperation
Description: "The ActivateOperation class allows a defined activity to be started (e.g., a zero pressure)."
Superclass: Operation
Subclasses: --
Name Binding: Instance-Number (not directly accessible by object management services)
Registered As: MDC_MOC_CNTRL_OP_ACTIV

6.6.7.1 Attributes

The ActivateOperation class does not define any additional attributes.

The ActivateOperation class defines in Table 65 the attribute groups or extensions to inherited attribute groups.

Table 65—ActivateOperation class attribute groups

Attribute group	Attribute group ID	Group elements
Operation Static Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_STATIC_CTXT	from Operation: Operation-Spec, Operation-Text-Strings
Operation Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_DYN_CTXT	from Operation: Vmo-Reference, Operational-State

6.6.7.2 Behavior

The ActivateOperation class does not define any special methods.

6.6.7.3 Notifications

The ActivateOperation class does not generate any special notifications.

6.6.8 LimitAlertOperation class

Class: LimitAlertOperation
Description: "The LimitAlertOperation class allows the limits of a limit alarm detector to be adjusted and the limit alarm to be switched on or off."
Superclass: Operation
Subclasses: --
Name Binding: Instance-Number (not directly accessible by object management services)
Registered As: MDC_MOC_CNTRL_OP_LIM

6.6.8.1 Attributes

The LimitAlertOperation class defines the attributes in Table 66.

Table 66—LimitAlertOperation class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Alert-Op-Capability	MDC_ATTR_AL_OP_CAPAB	AlOpCapab	Indicates what can be switched on or off.	M
Alert-Op-State	MDC_ATTR_AL_OP_STAT	CurLimAlStat	Current on/off state; can be set by Operation-Invoke method.	M
Current-Limits	MDC_ATTR_LIMIT_CURR	CurLimAlVal	Current alarm limits; can be set by Operation-Invoke method.	M
Alert-Op-Text-String	MDC_ATTR_AL_OP_TEXT_STRING	AlertOpTextString	Individual text for upper and lower limit.	O
Set-Value-Range	MDC_ATTR_VAL_RANGE	OpSetValueRange	Allowed range for limits.	M

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Unit-Code	MDC_ATTR_UNIT_CODE	OID-Type	Dimension of values.	M
Metric-Id	MDC_ATTR_ID_PHYSIO	OID-Type	Measurement (i.e., Numeric object) to which the limit applies, from metric nomenclature partition.	M

The LimitAlertOperation class defines in Table 67 the attribute groups or extensions to inherited attribute groups.

Table 67—LimitAlertOperation class attribute groups

Attribute group	Attribute group ID	Group elements
Operation Static Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_STATIC_CTXT	from Operation: Operation-Spec, Operation-Text-Strings from LimitAlertOperation: Alert-Op-Capability, Alert-Op-Text-String
Operation Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_DYN_CTXT	from Operation: Vmo-Reference, Operational-State from LimitAlertOperation: Alert-Op-State, Current-Limits, Set-Value-Range, Unit-Code, Metric-Id

The following ASN.1 data type definitions apply:

```
--
-- Alert operation static flags indicate which on/off flags are supported
--
AlOpCapab ::= BITS-16 {
      low-limit-sup(1),              -- supports low limit
      high-limit-sup(2),             -- supports high limit
      auto-limit-sup(5),             -- supports automatic limits
      low-lim-on-off-sup(8),         -- supports to switch on/off low limit
      high-lim-on-off-sup(9),        -- supports to switch on/off high limit
      lim-on-off-sup(10)             -- supports to switch on/off the complete alarm
}

--
-- Alert-Op-State attribute defines the current limit alert state
-- NOTE--The bits refer to the limit alarm only, not to the global alert state of the metric
--
CurLimAlStat ::= BITS-16 {
      lim-alert-off(0),              -- if this bit is set, all alerts (both high and low) are off
      lim-low-off(1),                -- low-limit violation detection is off
      lim-high-off(2)                -- high-limit violation detection is off
}

--
-- Current-Limits attribute
--
CurLimAlVal ::= SEQUENCE {
      lower       FLOAT-Type,
      upper       FLOAT-Type
}
```

```
--
-- Alert-Op-Text-String attribute assigns individual labels to upper and lower alarm limit
--
AlertOpTextString ::= SEQUENCE {
      lower-text   OCTET STRING,
      upper-text   OCTET STRING
}
```

6.6.8.2 Behavior

The LimitAlertOperation class does not define any special methods.

6.6.8.3 Notifications

The LimitAlertOperation class does not generate any special notifications.

6.6.9 SetRangeOperation class

Class: SetRangeOperation
Description: "The SetRangeOperation class allows the system to adjust low and high values (i.e., a value range) within defined boundaries."
Superclass: Operation
Subclasses: --
Name Binding: Instance-Number (not directly accessible by object management services)
Registered As: MDC_MOC_CNTRL_OP_SET_RANGE

6.6.9.1 Attributes

The SetRangeOperation class defines the attributes in Table 68.

Table 68—SetRangeOperation class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Current-Range	MDC_ATTR_ RANGE_CURR	CurrentRange	Current value.	M
Range-Op-Text	MDC_ATTR_ RANGE_OP_TEXT_ STRING	RangeOpText	Static attribute to define individual texts for upper and lower boundaries.	O
Set-Value-Range	MDC_ATTR_VAL_ RANGE	OpSetValueRange	Range of legal values.	M
Step-Width	MDC_ATTR_VAL_ STEP_WIDTH	OpValStepWidth	Allowed step width.	O
Unit-Code	MDC_ATTR_UNIT_ CODE	OID-Type	From dimensions nomenclature partition.	O

The SetRangeOperation class defines in Table 69 the attribute groups or extensions to inherited attribute groups.

Table 69—SetRangeOperation class attribute groups

Attribute group	Attribute group ID	Group elements
Operation Static Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_STATIC_CTXT	from Operation: Operation-Spec, Operation-Text-Strings from SetRangeOperation: Range-Op-Text

Attribute group	Attribute group ID	Group elements
Operation Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_DYN_CTXT	from Operation: Vmo-Reference, Operational-State from SetRangeOperation: Current-Range, Set-Value-Range, Step-Width, Unit-Code

The following ASN.1 data type definitions apply:

```
--
-- Current-Range attribute defines the current upper and lower range values
--
CurrentRange ::= SEQUENCE {
      lower       FLOAT-Type,
      upper       FLOAT-Type
}

--
-- Range-Op-Text attribute assigns labels to the upper and lower boundaries
--
RangeOpText ::= SEQUENCE {
      low-text    OCTET STRING,                 -- printable label text for low value
      high-text   OCTET STRING                  -- printable label text for high value
}
```

6.6.9.2 Behavior

The SetRangeOperation class does not define any special methods.

6.6.9.3 Notifications

The SetRangeOperation class does not generate any special notifications.

6.7 ExtendedServices package

6.7.1 Scanner class

Class: Scanner
Description: "A Scanner object is an observer and 'summarizer' of attribute values. It observes attributes of managed medical objects and generates summaries in the form of notification event reports. The Scanner object class is an abstract class, it cannot be instantiated."
Superclass: Top
Subclasses: CfgScanner, UcfgScanner
Name Binding: Handle
Registered As: MDC_MOC_SCAN

6.7.1.1 Attributes

The Scanner class defines the attributes in Table 70.

Table 70—Scanner class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Handle	MDC_ATTR_ID_ HANDLE	HANDLE	Scanners are identified by handles.	M

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Instance-Number	MDC_ATTR_ID_ INSTNO	InstNumber	Shall be used when dynamic creation of scanner instances is allowed.	C
Operational-State	MDC_ATTR_OP_ STAT	OperationalState	Defines if scanner is active; can be set.	M

The Scanner class defines in Table 71 the attribute groups or extensions to inherited attribute groups.

Table 71—Scanner class attribute groups

Attribute group	Attribute group ID	Group elements
Scanner Attribute Group (extensible attribute group)	MDC_ATTR_GRP_SCAN	from Scanner: Handle, Instance-Number, Operational-State

6.7.1.2 Behavior

The Scanner class does not define any special methods.

Scanner subclasses use the following ASN.1 data type definitions:

```
--
-- List of objects for which scanned attributes are refreshed
-- If list is empty, all objects in the scan list are refreshed
-- If scanned-attribute is 0 (NOS), all attributes of that object that are scanned are refreshed
-- If the object-glb-handle is 0 (in all components), the specified attribute ID is refreshed for all objects in the scan
-- list
--
RefreshObjList ::= SEQUENCE OF RefreshObjEntry

RefreshObjEntry ::= SEQUENCE {
    object-glb-handle       GLB-HANDLE,
    scanned-attribute       OID-Type        -- attribute ID from object-oriented nomenclature
                                            -- partition
}
```

6.7.1.3 Notifications

Events are defined in Scanner subclasses.

Most Scanner subclasses share a common event report data structure that is defined as follows:

```
--
-- A scanner may scan objects from multiple device contexts. For efficiency, scanned data that belongs to a single
-- device context are grouped together
--
ScanReportInfo ::= SEQUENCE {
    scan-report-no          INT-U16,                -- counter for detection of missing
                                                    -- events
    glb-scan-info           SingleCtxtScanList
}
```

```
SingleCtxtScanList ::= SEQUENCE OF SingleCtxtScan

SingleCtxtScan ::= SEQUENCE {
      context-id   MdsContext,
      scan-info    ObservationScanList
}

ObservationScanList ::= SEQUENCE OF ObservationScan

ObservationScan ::= SEQUENCE {
      obj-handle   HANDLE,
      attributes   AttributeList
}
```

6.7.2 CfgScanner class

Class: CfgScanner

Description: "The CfgScanner class defines a special attribute (i.e., the Scan-List attribute) that is used to configure which ~~of an object's~~ attributes of an object are scanned. A CfgScanner object has the following properties:

- It scans VMO-derived objects (mostly Metric, Channel, and VMD objects).
- It contains a list of scanned objects/attributes that can be modified. The CfgScanner object is an abstract class; it cannot be instantiated."

Superclass: Scanner

Subclasses: EpiCfgScanner, PeriCfgScanner

Name Binding: Handle

Registered As: MDC_MOC_SCAN_CFG

6.7.2.1 Attributes

The CfgScanner class defines the attributes in Table 72.

Table 72—CfgScanner class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Scan-List	MDC_ATTR_SCAN_LIST	ScanList	List of scanned objects and attributes; can be set.	M
Confirm-Mode	MDC_ATTR_CONFIRM_MODE	ConfirmMode	Determines whether confirmed event reports are used.	M
Confirm-Timeout	MDC_ATTR_CONFIRM_TIMEOUT	RelativeTime	Determines when a confirmed event report is resent in case of a missing response.	C
Transmit-Window	MDC_ATTR_TX_WIND	INT-U16	Maximum number of not-yet-acknowledged event reports at one time.	C
Scan-Config-Limit	MDC_ATTR_SCAN_CFG_LIMIT	ScanConfigLimit	Even a configurable scanner may restrict the way it can be configured.	O

The CfgScanner class defines in Table 73 the attribute groups or extensions to inherited attribute groups.

Table 73—CfgScanner class attribute groups

Attribute group	Attribute group ID	Group elements
Scanner Attribute Group	MDC_ATTR_GRP_SCAN	from Scanner:

Attribute group	Attribute group ID	Group elements
(extensible attribute group)		Handle, Instance-Number, Operational-State from CfgScanner: Scan-List, Confirm-Mode, Confirm-Timeout, Transmit-Window, Scan-Config-Limit

The following ASN.1 data type definitions apply:

```
--
-- Scan-List attribute determines which of an object's attributes of an object are observedscanned
-- NOTES
-- 1--If the scan list is empty, an episodic scanner has to send empty event reports
-- 2--The scan list will typically contain attribute group IDs for specific objects
--
ScanList ::= SEQUENCE OF ScanEntry

ScanEntry ::= SEQUENCE {
    object-glb-handle       GLB-HANDLE,   -- works for all objects with name binding handle
    scanned-attribute       OID-Type      -- could also be attribute group ID
}

--
-- Confirm-Mode attribute defines if confirmed event reports or unconfirmed event reports are used
--
ConfirmMode ::= INT-U16 {
    unconfirmed(0),
    confirmed(1)
}

--
-- Even a configurable scanner may restrict the way it can be configured
-- If Scan-Config-Limit attribute is absent, the scanner is fully configurable
--
ScanConfigLimit ::= BITS-16 {
    no-scan-delete(0),          -- scanner cannot be deleted
    no-scan-list-mod(1),        -- scan list cannot be dynamically modified
    auto-init-scan-list(3),     -- scan list is automatically initialized after scanner create
    auto-updt-scan-list(4)      -- scan list is automatically updated in case of configuration change
}
```

6.7.2.2 Behavior

The CfgScanner class does not define any special methods.

6.7.2.3 Notifications

The CfgScanner class does not generate any special notifications.

6.7.3 EpiCfgScanner class

Class: EpiCfgScanner
Description: "An EpiCfgScanner object is responsible for scanning attributes or attribute groups of objects and for reporting these attributes in episodic, unbuffered (i.e., on change only) event reports."
Superclass: CfgScanner
Subclasses: --

Name Binding: Handle
Registered As: MDC_MOC_SCAN_CFG_EPI

6.7.3.1 Attributes

The EpiCfgScanner class does not define attributes other than the attributes inherited from the CfgScanner class.

The EpiCfgScanner class uses the Scanner Attribute Group that is inherited from the CfgScanner class.

6.7.3.2 Behavior

The EpiCfgScanner class defines the methods in Table 74.

Table 74—EpiCfgScanner instance methods

Action	Mode	Action ID	Action parameter	Action result
Refresh-Episodic-Data	Confirmed	MDC_ACT_REFR_EPI_DATA	RefreshObjList	—

The Refresh-Episodic-Data method triggers a refresh of all scanned attributes.

6.7.3.3 Notifications

The EpiCfgScanner class defines the events in Table 75.

Table 75—EpiCfgScanner events

Event	Mode	Event ID	Event parameter	Event result
Unbuf-Scan-Report	Confirmed/Unconfirmed	MDC_NOTI_UNBUF_SCAN_RPT	ScanReportInfo	—

NOTE 1—If the EpiCfgScanner scans attribute groups of an object and one or more of the attribute values in the group change, then the scanner reports all values of attributes in the group, even those that did not change their value. This is important so that attributes that are dynamically deleted from an object instance can be detected without a special notification.

NOTE 2—If no attribute of an object changes its value, then no data of this object are included in the scan report (unless an explicit refresh phase was triggered).

NOTE 3—Because an episodic scanner does not buffer any changes and does not have an update period specification attribute (which is not needed because updates are sent on value changes), attribute change notifications should be sent at a rate that ensures no data loss. For example, in order to ensure that no metric value changes more than once between scans of dynamic attribute groups, the episodic scanner should check for changes at a rate at least as fast as the the shortest MetricSpec::update-period of the metric instances in the scanner's scan list.

NOTE 4—After instantiation of the scanner, all attribute values are considered changed so that the first scan report contains all attribute values of all objects.

6.7.4 PeriCfgScanner class

Class: PeriCfgScanner
Description: "A PeriCfgScanner object is responsible for scanning attributes and attribute groups of objects and for reporting these attributes in periodic event reports."

Superclass: CfgScanner
Subclasses: FastPeriCfgScanner
Name Binding: Handle
Registered As: MDC_MOC_SCAN_CFG_PERI

6.7.4.1 Attributes

The PeriCfgScanner class defines the attributes in Table 76.

Table 76—PeriCfgScanner class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Scan-Extensibility	MDC_ATTR_SCAN_EXTEND	ScanExtend	Default is extensive.	M
Reporting-Interval	MDC_ATTR_SCAN_REP_PD	RelativeTime	Period of reports.	M

The PeriCfgScanner class defines in Table 77 the attribute groups or extensions to inherited attribute groups.

Table 77—PeriCfgScanner class attribute groups

Attribute group	Attribute group ID	Group elements
Scanner Attribute Group (extensible attribute group)	MDC_ATTR_GRP_SCAN	from Scanner: Handle, Instance-Number, Operational-State from CfgScanner: Scan-List, Confirm-Mode, Confirm-Timeout, Transmit-Window, Scan-Config-Limit from PeriCfgScanner: Scan-Extensibility, Reporting-Interval

The following ASN.1 data type definitions apply:

```
--
-- Scan-Extensibility attribute defines if the scanner includes all observations in the ScanReportInfo event
-- parameter or if it includes just the latest observation (i.e., superpositive)
--
ScanExtend ::= INT-U16 {
      extensive(0),                 -- all attribute changes in the scan period are included
      superpositive(1),             -- only the last attribute change is included
      superpositive-avg(2)          -- superpositive, but all values in period are averaged
}
```

6.7.4.2 Behavior

The PeriCfgScanner class does not define any special methods.

6.7.4.3 Notifications

The PeriCfgScanner class defines the events in Table 78.

Table 78—PeriCfgScanner events

Event	Mode	Event ID	Event parameter	Event result
Buf-Scan-Report	Confirmed/Unconfirmed	MDC_NOTI_BUF_SCAN_RPT	ScanReportInfo	—

6.7.5 FastPeriCfgScanner class

Class: FastPeriCfgScanner
Description: "The FastPeriCfgScanner class is a specialized class for scanning the observed value attribute of a RealTimeSampleArray object. This special Scanner class is further optimized for low-latency reporting and efficient communication bandwidth utilization, which is required to access real-time waveform data."
Superclass: PeriCfgScanner
Subclasses: --
Name Binding: Handle
Registered As: MDC_MOC_SCAN_CFG_PERI_FAST

6.7.5.1 Attributes

The FastPeriCfgScanner class does not define any addtional attributes.

6.7.5.2 Behavior

The FastPeriCfgScanner class does not define any special methods.

6.7.5.3 Notifications

The FastPeriCfgScanner class defines the events in Table 79.

Table 79—FastPeriCfgScanner events

Event	Mode	Event ID	Event parameter	Event result
Fast-Buf-Scan-Report	Confirmed/Unconfirmed	MDC_NOTI_FAST_BUF_SCAN_RPT	FastScanReportInfo	—

The following ASN.1 data type definitions apply:

```
--
-- Event report contains the observed values of scanned RealTimeSampleArray objects
--
FastScanReportInfo ::= SEQUENCE {
      scan-report-no                INT-U16,
      glb-scan-info                 SingleCtxtFastScanList
}

SingleCtxtFastScanList ::= SEQUENCE OF SingleCtxtFastScan

SingleCtxtFastScan ::= SEQUENCE {
      context-id    MdsContext,
      scan-info     RtsaObservationScanList
}

RtsaObservationScanList ::= SEQUENCE OF RtsaObservationScan
```

```
RtsaObservationScan ::= SEQUENCE {
    handle                  HANDLE,
    observation             SaObsValue
}
```

A FastPeriCfgScanner object is a dedicated scanner for RealTimeSampleArray objects. For performance reasons, the sample arrays do not carry a separate timestamp in each observation scan structure. For time synchronization and timestamping of specific samples, two different methods can be supported:

a) The default method assumes that the timestamp provided by the EVENT REPORT service is the time of the first sample value in each RtsaObservationScan::SaObsValue data structure
b) For higher precision time synchronization, RealTimeSampleArray objects may support the Average-Reporting-Delay and Sample-Time-Sync attributes. The support for this method is signaled by the presence of the Time-Support::time-capability-time-capab-rtsa-time-sync-highprecision flag in the Clock object. If this method is used, the individual sample times are determined by these attributes and they are independent of the timestamp provided by the EVENT REPORT service.

6.7.6 UcfgScanner class

Class: UcfgScanner
Description: "A UcfgScanner object scans a predefined set of managed medical objects that cannot be modified. In other words, a UcfgScanner object typically is a reporting object that is specialized for one specific purpose. It has the following properties:
— Scanner event reports are typically used in confirmed mode because the data they contain are not stateless.
— The list of scanned objects/attributes is fixed (i.e., cannot be configured).The UcfgScanner object is an abstract class; it cannot be instantiated."
Superclass: Scanner
Subclasses: AlertScanner, ContextScanner, OperatingScanner
Name Binding: Handle
Registered As: MDC_MOC_SCAN_UCFG

6.7.6.1 Attributes

The UcfgScanner class defines the attributes in Table 80.

Table 80—UcfgScanner class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Confirm-Mode	MDC_ATTR_ CONFIRM_MODE	ConfirmMode	Default is confirmed mode.	O
Confirm-Timeout	MDC_ATTR_ CONFIRM_ TIMEOUT	RelativeTime	Determines when a confirmed event report is resent in case of a missing response.	O
Transmit-Window	MDC_ATTR_TX_ WIND	INT-U16	Maximum number of not-yet-acknowledged event reports at one time.	O

The UcfgScanner class defines in Table 81 the attribute groups or extensions to inherited attribute groups.

Table 81—UcfgScanner class attribute groups

Attribute group	Attribute group ID	Group elements
Scanner Attribute Group (extensible attribute group)	MDC_ATTR_GRP_SCAN	from Scanner: Handle, Instance-Number, Operational-State from UcfgScanner: Confirm-Mode, Confirm-Timeout, Transmit-Window

6.7.6.2 Behavior

The UcfgScanner class does not define any special methods.

6.7.6.3 Notifications

The UcfgScanner class does not generate any special notifications.

6.7.7 ContextScanner class

Class: ContextScanner
Description: "ContextScanner objects are responsible for observing device configuration changes. After instantiation, a ContextScanner object is responsible for announcing the object instances in the device's MDIB. The scanner provides the object instance containment hierarchy and static attribute values. In case of dynamic configuration changes, the ContextScanner object sends notifications about new object instances or deleted object instances."
Superclass: UcfgScanner
Subclasses: --
Name Binding: Handle
Registered As: MDC_MOC_SCAN_UCFG_CTXT

6.7.7.1 Attributes

The ContextScanner class defines the attributes in Table 82.

Table 82—ContextScanner class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Context-Mode	MDC_ATTR_SCAN_CTXT_MODE	ContextMode	Default is dynamic.	M

The ContextScanner class uses the Scanner Attribute Group that is defined by the Scanner class.

The following ASN.1 data type definitions apply:

```
--
-- Context-Mode attribute determines if the context scanner sends create notifications for the maximum set of
-- object instances in the MDIB (and requires no delete notifications) or for active objects only
--
ContextMode ::= INT-U16 {
      static-mode(0),
      dynamic-mode(1)
}
```

6.7.7.2 Behavior

The ContextScanner class defines the methods in Table 83.

Table 83—ContextScanner instance methods

Action	Mode	Action ID	Action parameter	Action result
Refresh-Context	Confirmed	MDC_ACT_REFR_CTXT	RefreshObjList	ObjCreateInfo (scan report no is 0)

The Refresh-Context method returns configuration information for all object instances currently in the MDIB.

6.7.7.3 Notifications

The ContextScanner class defines the events in Table 84.

Table 84—ContextScanner events

Event	Mode	Event ID	Event parameter	Event result
Object-Create-Notification	Confirmed/Unconfirmed	MDC_NOTI_OBJ_CREAT	ObjCreateInfo	—
Object-Delete-Notification	Confirmed	MDC_NOTI_OBJ_DEL	ObjDeleteInfo	—

The following ASN.1 data type definitions apply:

```
--
-- Object-Create-Notification event contains type, ID, and attribute information about new object instances in the
-- MDIB
--
ObjCreateInfo ::= SEQUENCE {
      scan-report-no                INT-U16,
      scan-report-info              CreateEntryList
}

CreateEntryList ::= SEQUENCE OF CreateEntry

--
-- A single new entry for one parent object, necessary to construct hierarchy in MDIB
--
CreateEntry ::= SEQUENCE {
      superior-object               ManagedObjectId,
      created-object                CreatedObjectList
}

CreatedObjectList ::= SEQUENCE OF CreatedObject

--
-- Now finally the new object itself
--
CreatedObject ::= SEQUENCE {
      class-id      ManagedObjectId,
      attributes    AttributeList
```

```
}

--
-- Object-Delete-Notification event implicitly deletes all child objects as well
--
ObjDeleteInfo ::= SEQUENCE {
    scan-report-no          INT-U16,
    object-list             ManagedObjectIdList
}

ManagedObjectIdList ::= SEQUENCE OF ManagedObjectId
```

6.7.8 AlertScanner class

Class: AlertScanner

Description: "An AlertScanner object is responsible for observing the alert-related attribute groups of objects defined in the Alert Package. As alarming in general is security-sensitive, the scanner is not configurable (i.e., all or no Alert objects are scanned). An AlertScanner object sends event reports periodically so that timeout conditions can be checked."

Superclass: UcfgScanner

Subclasses: --

Name Binding: Handle

Registered As: MDC_MOC_SCAN_UCFG_ALSTAT

6.7.8.1 Attributes

The AlertScanner class defines the attributes in Table 85.

Table 85—AlertScanner class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Reporting-Interval	MDC_ATTR_SCAN_REP_PD	RelativeTime	Period of reports.	M

The AlertScanner class uses the Scanner Attribute Group that is defined by the Scanner class.

6.7.8.2 Behavior

The AlertScanner class does not define any special methods.

6.7.8.3 Notifications

The AlertScanner class defines the events in Table 86.

Table 86—AlertScanner events

Event	Mode	Event ID	Event parameter	Event result
Alert-Scan-Report	Confirmed/Unconfirmed	MDC_NOTI_AL_STAT_ SCAN_RPT	ScanReportInfo	—

6.7.9 OperatingScanner class

Class: OperatingScanner

Description: "The OperatingScanner class is responsible for providing all information about the operating and control system of the medical device. This information mainly includes SCO-contained Operation objects, which are considered SCO properties, not separate managed medical objects. The operating scanner

- Sends CREATE events for Operation object instances
- Scans Operation attributes together with attributes of the SCO Transaction Group (see6.6.1.1)
- Provides a refresh mechanism for Operation attributes."

Superclass: UcfgScanner
Subclasses: --
Name Binding: Handle
Registered As: MDC_MOC_SCAN_UCFG_OP

6.7.9.1 Attributes

The OperatingScanner class does not define any additional attributes.

The OperatingScanner class uses the Scanner Attribute Group that is defined by the Scanner class.

6.7.9.2 Behavior

The OperatingScanner class defines the methods in Table 87.

Table 87—OperatingScanner instance methods

Action	Mode	Action ID	Action parameter	Action result
Refresh-Operation-Context	Confirmed	MDC_ACT_REFR_OP_CTXT	RefreshObjList	OpCreateInfo (scan report no is 0)
Refresh-Operation-Attributes	Confirmed	MDC_ACT_REFR_OP_ATTR	RefreshObjList	—

NOTE—The RefreshObjList action parameter for the Refresh-Operation-Attributes method may identify both SCO attributes and Operation attributes.

6.7.9.3 Notifications

The OperatingScanner class defines the events in Table 88.

Table 88—OperatingScanner events

Event	Mode	Event ID	Event parameter	Event result
Oper-Create-Notification	Confirmed/Unconfirmed	MDC_NOTI_OP_CREAT	OpCreateInfo	—
Oper-Delete-Notification	Confirmed	MDC_NOTI_OP_DEL	OpDeleteInfo	—
Oper-Attribute-Update	Confirmed/Unconfirmed	MDC_NOTI_OP_ATTR _UPDT	OpAttributeInfo	—

The following ASN.1 data type definitions apply:

--

```
-- Support data types
--
OpElemAttr ::= SEQUENCE {
    op-class-id             OID-Type,
    op-instance-no          InstNumber,
    attributes              AttributeList
}

OpElemAttrList ::= SEQUENCE OF OpElemAttr

--
-- Create and delete operations
--
OpCreateInfo ::= SEQUENCE {
    scan-report-no          INT-U16,
    scan-info               OpCreateEntryList
}

OpCreateEntryList ::= SEQUENCE OF OpCreateEntry

OpCreateEntry ::= SEQUENCE {
    sco-glb-handle          GLB-HANDLE,
    created-op-list         OpElemAttrList
}

OpDeleteInfo ::= SEQUENCE {
    scan-report-no          INT-U16,
    deleted-op-list         OpDeleteEntryList
}

OpDeleteEntryList ::= SEQUENCE OF OpDeleteEntry

OpDeleteEntry ::= SEQUENCE {
    sco-glb-handle          GLB-HANDLE,
    deleted-op-list         OpElemList
}

OpElemList ::= SEQUENCE OF OpElem

OpElem ::= SEQUENCE {
    op-class-id             OID-Type,
    op-instance-no          InstNumber
}

--
-- Report of Operation attributes (from multiple contexts, if necessary)
--
OpAttributeInfo ::= SEQUENCE {
    scan-report-no          INT-U16,
    glb-scan-info           SingleCtxtOperScanList
}

SingleCtxtOperScanList ::= SEQUENCE OF SingleCtxtOperScan

SingleCtxtOperScan ::= SEQUENCE {
    context-id   MdsContext,
    scan-info    OpAttributeScanList
}

OpAttributeScanList ::= SEQUENCE OF OpAttributeScan
```

```
--
-- The scanned information contains SCO transaction attributes and Operation attributes
--
OpAttributeScan ::= SEQUENCE {
    sco-handle              HANDLE,
    invoke-cookie           INT-U32,
    lock-state              AdministrativeState,
    op-elem-updt-list       OpElemAttrList
}
```

6.8 Communication package

6.8.1 CommunicationController class

Class: CommunicationController
Description: "The CommunicationController class represents the upper layer and lower layer communication profile (i.e., the application profile, the format profile, and the transport profile) and provides access methods for obtaining management information related to data communications."
Superclass: Top
Subclasses: BCC, DCC
Name Binding: Handle
Registered As: MDC_MOC_CC (from object-oriented nomenclature partition)

6.8.1.1 Attributes

The CommunicationController class defines the attributes in Table 89.

Table 89—CommunicationController class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Handle	MDC_ATTR_ID_HANDLE	HANDLE	ID for referencing the object	M
Capability	MDC_ATTR_CC_CAPAB	CcCapability	Bit field indicating specific capabilities of the CommunicationController implementation.	M
Cc-Type	MDC_ATTR_CC_TYPE	CC-Oid	Could be used to specify variants, e.g., ISO/IEEE 11073, local area network (LAN), combinations.	O
Number-Of-Difs	MDC_ATTR_CC_NUM_DIFS	INT-U16	Number of device interfaces; defaults to 1 if not present. DeviceInterface objects are identified by their index. The index is a 16-bit number between 1 and the Number-OfDifs attribute value. The list is statically configured at CommunicationController configuration time.	O
This-Connection-Dif-Index	MDC_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

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Cc-Ext-Mgmt-Proto-Id	MDC_ATTR_CC_EXT_MNG_PROT	CcExtMgmtProto	Specifies ID for an external management protocol, e.g., Simple Network Management Protocol (SNMP) or Common Management Information Protocol (CMIP).	O

The CommunicationController class defines in Table 90 the attribute groups or extensions to inherited attribute groups.

Table 90—CommunicationController class attribute groups

Attribute group	Attribute group ID	Group elements
CommunicationController Attribute Group (extensible attribute group)	MDC_ATTR_GRP_CC	from CommunicationController: Handle, Capability, Cc-Type, Number-Of-Difs, This-Connection-Dif-Index, Cc-Ext-Mgmt-Proto-Id

The following ASN.1 data type definitions apply:

```
--
-- Capability attribute specifies the CommunicationController object
--
CcCapability ::= BITS-32 {
      cc-sup-ext-mgmt-protocol(0)    -- CommunicationController object supports an external management
                                     -- protocol (e.g., SNMP); if this bit is set, then the presence of the CC-Ext-
                                     -- Mgmt-Proto-Id attribute is required
}

--
-- CC-OID data type is a regular 16-bit OID from the infrastructure elements nomenclature partition
--
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-proto-snmp-v1(1),         -- Simple Network Management Protocol Version 1
      mgmt-proto-snmp-v2(2),         -- Simple Network Management Protocol Version 2
      mgmt-proto-snmp-v3(3),         -- Simple Network Management Protocol Version 3
      mgmt-proto-cmip(16)            -- Common Management Information Protocol
}
```

6.8.1.2 Behavior

The ~~AlertScanner~~ CommunicationController class defines the events in Table 91.

Table 91—CommunicationController instance methods

Action	Mode	Action ID	Action parameter	Action result
Get-Mib-Data	Confirmed	MDC_ACT_GET_MIB_DATA	GetMibDataRequest	GetMibDataResult

The following ASN.1 data type definitions apply:

```
--
-- Data type for the ACTION service
-- One request can retrieve data for one device interface only
--
-- NOTE--If the mib-id-list type is empty, no MibElement data are returned in the response; valid entries in the
-- mib-id-list type are defined in the Mib-Element-List attribute of the DeviceInterfaceMibElement object
--
GetMibDataRequest ::= SEQUENCE {
      dif-index     INT-U16,
      mib-id-list   MibIdList
}

MibIdList ::= SEQUENCE OF CC-Oid

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

MibDataList ::= SEQUENCE OF MibDataEntry

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

6.8.1.3 Notifications

The CommunicationController class does not generate any special notifications.

6.8.2 DCC class

Class:	DCC
Description:	"The DCC class is a CommunicationController class used by medical devices operating as agent systems (i.e., association responders)."
Superclass:	CommunicationController
Subclasses:	--
Name Binding:	Handle
Registered As:	MDC_MOC_DCC (from object-oriented nomenclature partition)

The DCC class does not define any attributes, methods, or notifications.

6.8.3 BCC class

Class:	BCC
Description:	"The BCC class is a CommunicationController class used by medical devices operating as manager systems (i.e., association requestors)."
Superclass:	CommunicationController
Subclasses:	--
Name Binding:	Handle
Registered As:	MDC_MOC_BCC (from object-oriented nomenclature partition)

The BCC class does not define any attributes, methods, or notifications.

6.8.4 DeviceInterface class

Class: DeviceInterface
Description: "A DeviceInterface object represents a BCC or DCC communication port that is an end point of a single association for which (e.g., statistical) data are independently collected by the CommunicationController object. DeviceInterface objects are not accessible by CMDISE services."
Superclass: --
Subclasses: --
Name Binding: -
Registered As: MDC_CC_DIF (from infrastructure nomenclature partition)

6.8.4.1 Attributes

The DeviceInterface class does not define any attributes. All its properties are captured in the DeviceInterfaceMibElement class. A MibElement instance is mandatory for each instance of DeviceInterface that is supported by a device's CommunicationController instance.

6.8.4.2 Behavior

The DeviceInterface class does not define any special methods.

6.8.4.3 Notifications

The DeviceInterface class does not generate any special notifications.

6.8.5 MibElement class

Class: MibElement
Description: "A MibElement object represents management information about a specific physical or logical port of a DeviceInterface object. The MibElement class is an abstract base class only."
Superclass: --
Subclasses: DeviceInterfaceMibElement, GeneralCommunicationStatisticsMibElement
Name Binding: -
Registered As: MDC_CC_MIB_ELEM (from infrastructure nomenclature partition)

6.8.5.1 Attributes

The MibElement class defines the attributes in Table 92.

Table 92—MibElement class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Mib-Ext-Oid	MDC_CC_MIB_DATA_EXT_OID	OCTET STRING	The OCTET STRING contains a registered ISO OID that is fully encoded by basic encoding rules (BER) if the MibElement is a registered concept. The size of the OCTET STRING shall be even and may require a padding byte. This attribute allows the inclusion of management information base (MIB) definitions from other standards here.	O

The attributes can only be retrieved by the Get-Mib-Data method defined on the CommunicationController class.

The MibElement class does not define any attribute groups.

6.8.5.2 Behavior

The MibElement class does not define any special methods.

6.8.5.3 Notifications

The MibElement class does not generate any special notifications.

6.8.6 DeviceInterfaceMibElement class

Class: DeviceInterfaceMibElement
Description: "A DeviceInterfaceMibElement object describes the properties of a DeviceInterface object. This MibElement object is mandatory for each DeviceInterface object of the CommunicationController object."
Superclass: MibElement
Subclasses: --
Name Binding: -
Registered As: MDC_CC_MIB_ELEM_DIF (from infrastructure nomenclature partition)

6.8.6.1 Attributes

The DeviceInterfaceMibElement class defines the attributes in Table 93.

Table 93—DeviceInterfaceMibElement class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Dif-Id	MDC_CC_MIB_DATA_DIF_ID	INT-U16	Between 1 and the Number-OfDifs attribute value in the CommunicationController object.	M
Port-State	MDC_CC_MIB_DATA_DIF_PORT_ST	DifMibPortState	State information about the port.	M
Port-Number	MDC_CC_MIB_DATA_DIF_PORT_NO	INT-U16	Logical port number of this device interface.	O
Dif-Type	MDC_CC_MIB_DATA_DIF_TYPE	CC-Oid	Assumes entries in infrastructure nomenclature partition.	O
Active-Profile	MDC_CC_MIB_DATA_PROFILE_ID	OID-Type	This ID should contain the Profile Support Attribute ID (see ISO/IEEE 11073-20101) as used in the ACSE user information 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	MDC_CC_MIB_DATA_SUPP_PROFILES	SupportedProfileList	See below	O

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
MTU	MDC_CC_MIB_ DATA_MTU	INT-U32	Maximum transmit unit, in bytes	O
Link-Speed	MDC_CC_MIB_ DATA_LINK_ SPEED	INT-U32	In bits per second	O
Mib-Element-List	MDC_CC_MIB_ DATA_MIB_ ELEM_LIST	MibElementList	A list of MibElements supported by the DeviceInterface object (in addition to this mandatory MibElement). Assumes that the MibElements are registered in the nomenclature.	M

The attributes can only be retrieved by the Get-Mib-Data method defined on the CommunicationController class.

The DeviceInterfaceMibElement class does not define any attribute groups.

The following ASN.1 data type definitions apply:

```
--
-- Supported-Profiles attribute defines which profiles are supported by the DeviceInterface object; entries in the
-- list are Profile Support Attribute IDs as used in the ACSE user information structure used for negotiating the
-- active profile (see definition of application profiles) (entries in the list are from the infrastructure nomenclature
-- partition)
--
SupportedProfileList ::= SEQUENCE OF CC-Oid

--
-- The Mib-Element-List attribute defines which MibElement objects are supported by the DeviceInterface object
-- (entries in the list are from the infrastructure nomenclature partition)
--
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
}
```

6.8.6.2 Behavior

The DeviceInterfaceMibElement class does not define any special methods.

6.8.6.3 Notifications

The DeviceInterfaceMibElement class does not generate any special notifications.

6.8.7 GeneralCommunicationStatisticsMibElement class

Class: GeneralCommunicationStatisticsMibElement

Description: "The GeneralCommunicationStatisticsMibElement class represents generic communication statistics for one device interface."
Superclass: MibElement
Subclasses: --
Name Binding: -
Registered As: MDC_CC_MIB_ELEM_GEN_COMM_STATS (from infrastructure elements nomenclature table)

6.8.7.1 Attributes

The GeneralCommunicationStatisticsMibElement class defines the attributes in Table 94.

Table 94—GeneralCommunicationStatisticsMibElement class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Packets-In	MDC_CC_MIB_DATA_PACK_IN	MibCcCounter	The number of packets received.	O
Packets-Out	MDC_CC_MIB_DATA_PACK_OUT	MibCcCounter	The number of packet issued.	O
Octets-In	MDC_CC_MIB_DATA_OCT_IN	MibCcCounter	The number of payload bytes received at transport interface (e.g., without framing).	O
Octets-Out	MDC_CC_MIB_DATA_OCT_OUT	MibCcCounter	The number of payload bytes sent at transport interface (e.g., without framing).	O
Discarded-Packets-In	MDC_CC_MIB_DATA_DISC_PACK_IN	MibCcCounter	Received packets not delivered to upper layers.	O
Discarded-Packets-Out	MDC_CC_MIB_DATA_DISC_PACK_OUT	MibCcCounter	Packets from upper layers not sent to network interface.	O
Unknown-Protocol-Packets-In	MDC_CC_MIB_DATA_UNK_PROT_PACK_IN	MibCcCounter	Received packets with unknown protocol.	O
Queue-Len-In	MDC_CC_MIB_DATA_QUEUE_LEN_IN	MibCcGauge	Size of output packet queue in bytes.	O
Queue-Len-Out	MDC_CC_MIB_DATA_QUEUE_LEN_OUT	MibCcGauge	Size of input packet queue in bytes.	O
Dif-Admin-Status	MDC_CC_MIB_DATA_DIF_STATE	OperationalState	Desired device interface state.	O
Dif-Oper-Status	MDC_CC_MIB_DATA_CUR_DIF_STATE	OperationalState	Current device interface status.	O
Dif-Last-Change	MDC_CC_MIB_DATA_TIME_DIF_LAST_CHANGE	AbsoluteTime	The time when the device interface last changed state.	O
Errors-In	MDC_CC_MIB_DATA_ERRS_IN	MibCcCounter	Corrupt received packets.	O
Errors-Out	MDC_CC_MIB_DATA_ERRS_OUT	MibCcCounter	Corrupt sent packets.	O

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Generic-Mode	MDC_CC_MIB_DATA_COMM_MODE	MibCcCommMode	The mode of the communication.	O
Average-Speed	MDC_CC_MIB_DATA_AVG_SPEED	INT-U32	In bits per second.	O
Maximum-Speed	MDC_CC_MIB_DATA_MAX_SPEED	INT-U32	In bits per second.	O

The attributes can only be retrieved by the Get-Mib-Data method defined on the CommunicationController class.

The GeneralCommunicationStatisticsMibElement class does not define any attribute groups.

The following ASN.1 data type definitions apply:

```
--
-- The gauge type (from IETF RFC 1155) represents a non-negative integer that may increase or decrease, but that
-- latches at a maximum value
--
MibCcGauge ::= INT-U32

--
-- The counter type (from IETF RFC 1155) represents a non-negative integer that monotonically increases until it
-- reaches a maximum value, at which time it wraps around and starts increasing again from 0
--
MibCcCounter ::= INT-U32

--
-- The communication mode type represents the communication modes that are supported by the device interface.
--
MibCcCommMode ::= BITS-32 {
      comm-mode-simplex(0),
      comm-mode-half-duplex(1),
      comm-mode-full-duplex(2)
}
```

6.8.7.2 Behavior

The GeneralCommunicationStatisticsMibElement class does not define any special methods.

6.8.7.3 Notifications

The GeneralCommunicationStatisticsMibElement class does not generate any special notifications.

6.9 Archival package

6.9.1 MultiPatientArchive class

Class: MultiPatientArchive

Description: "A MultipatientArchive object groups together one or more PatientArchive objects."
Superclass: Top
Subclasses: --
Name Binding: Handle
Registered As: MDC_MOC_ARCHIVE_MULTI_PT

6.9.1.1 Attributes

The MultiPatientArchive class defines the attributes in Table 95.

Table 95—MultiPatientArchive class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Handle	MDC_ATTR_ID_ HANDLE	HANDLE	Name binding attribute.	M
System-Id	MDC_ATTR_SYS_ID	OCTET STRING		M
Location	MDC_ATTR_ LOCATION	OCTET STRING	Example: name of hospital.	M
Study-Name	MDC_ATTR_ STUDYNAME	OCTET STRING		M
Version_dim	MDC_ATTR_ ARCHIVE_VERS	OCTET STRING	Example: ADS version 1.0.	M

The MultiPatientArchive class defines in Table 96 the attribute groups or extensions to inherited attribute groups.

Table 96—MultiPatientArchive class attribute groups

Attribute group	Attribute group ID	Group elements
Archival Attribute Group	MDC_ATTR_GRP_ARCHIVE	from MultiPatientArchive: Handle, System-Id, Location, Study-Name, Version_dim

6.9.1.2 Behavior

The MultiPatientArchive class does not define any special methods.

6.9.1.3 Notifications

The MultiPatientArchive class does not generate any special notifications.

6.9.2 PatientArchive class

Class: PatientArchive
Description: "A PatientArchive object groups together vital signs and other information about a single patient."
Superclass: Top
Subclasses: --
Name Binding: Handle
Registered As: MDC_MOC_ARCHIVE_PT

6.9.2.1 Attributes

The PatientArchive class defines the attributes in Table 97.

Table 97—PatientArchive class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Handle	MDC_ATTR_ID_HANDLE	HANDLE	Name binding attribute.	M
System-Id	MDC_ATTR_SYS_ID	OCTET STRING		M
System-Name	MDC_ATTR_NAME_SYS	OCTET STRING	Example: filename.	M
Processing-History	MDC_ATTR_PROC_HIST	OCTET STRING	Example: not processed.	M
Protection	MDC_ATTR_PROTECTION	ArchiveProtection	Example: original recording.	M

The PatientArchive class defines in Table 98 the attribute groups or extensions to inherited attribute groups.

Table 98—PatientArchive class attribute groups

Attribute group	Attribute group ID	Group elements
Archival Attribute Group	MDC_ATTR_GRP_ARCHIVE	from PatientArchive: Handle, System-Id, System-Name, Processing-History, Protection

The following ASN.1 data type definitions apply:

```
--
-- Protection attribute defines the mechanism used for access control; this mechanism is vendor-specific or
-- implementation-specific
--
ArchiveProtection ::= SEQUENCE {
      protection-type                PrivateOid,
      protection-key                 ANY DEFINED BY  protection-type
}
```

6.9.2.2 Behavior

The PatientArchive class does not define any special methods.

6.9.2.3 Notifications

The PatientArchive class does not generate any special notifications.

6.9.3 SessionArchive class

Class: SessionArchive
Description: "A SessionArchive object contains information on a single patient that is collected during one stay or visit."
Superclass: Top
Subclasses: --
Name Binding: Handle

Registered As: MDC_MOC_ARCHIVE_SESSION

6.9.3.1 Attributes

The SessionArchive class defines the attributes in Table 99.

Table 99—SessionArchive class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Handle	MDC_ATTR_ID_ HANDLE	HANDLE	Name binding attribute.	M
S-Archive-Id	MDC_ATTR_ID_ SESS_ARCHIVE	OCTET STRING		M
S-Archive-Name	MDC_ATTR_NAME_ SESS_ARCHIVE	OCTET STRING	Example: study name.	M
S-Archive-Comments	MDC_ATTR_SESS_ ARCHIVE_ COMMENTS	OCTET STRING	Example: part one of MSLT test.	O
Start-Time	MDC_ATTR_TIME_ START	AbsoluteTime		M
Stop-Time	MDC_ATTR_TIME_ STOP	AbsoluteTime		M
Protection	MDC_ATTR_ PROTECTION	ArchiveProtection	Example: original recording.	C

The SessionArchive class defines in Table 100 the attribute groups or extensions to inherited attribute groups.

Table 100—SessionArchive class attribute groups

Attribute group	Attribute group ID	Group elements
Archival Attribute Group	MDC_ATTR_GRP_ARCHIVE	from SessionArchive: Handle, S-Archive-Id, S-Archive-Name, S-Archive-Comments, Start-Time, Stop-Time, Protection

6.9.3.2 Behavior

The SessionArchive class does not define any special methods.

6.9.3.3 Notifications

The SessionArchive class does not generate any special notifications.

6.9.4 Physician class

Class: Physician
Description: "The Physician class represents a physician."
Superclass: Top
Subclasses: --
Name Binding: Handle
Registered As: MDC_MOC_PHYSICIAN

6.9.4.1 Attributes

The Physician class defines the attributes in Table 101.

Table 101 —Physician class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Handle	MDC_ATTR_ID_HANDLE	HANDLE	Name binding attribute.	M
Physician-Id	MDC_ATTR_ID_PHYSICIAN	OCTET STRING		M
Authorization-Level	MDC_ATTR_AUTH_LEVEL	Authorization		C
Name	MDC_ATTR_PHYSICIAN_NAME	OCTET STRING	Unstructured physician name.	O
Given-Name	MDC_ATTR_PHYSICIAN_NAME_GIVEN	OCTET STRING		O
Family-Name	MDC_ATTR_PHYSICIAN_NAME_FAMILY	OCTET STRING		O
Middle-Name	MDC_ATTR_PHYSICIAN_NAME_MIDDLE	OCTET STRING		O
Title-Name	MDC_ATTR_PHYSICIAN_NAME_TITLE	OCTET STRING	Example: Professor.	O

The Physician class defines in Table 102 the attribute groups or extensions to inherited attribute groups.

Table 102 —Physician class attribute groups

Attribute group	Attribute group ID	Group elements
Physician Attribute Group	MDC_ATTR_GRP_PHYSICIAN	from Physician: Handle, Physician-Id, Authorization-Level, Name, Given-Name, Family-Name, Middle-Name, Title-Name

The following ASN.1 data type definitions apply:

```
--
-- Authorization-Level attribute defines the access rights used for access control; this mechanism is vendor-specific or
-- implementation-specific --
Authorization ::= SEQUENCE {
      authorization-type          PrivateOid,
      authorization-key           ANY DEFINED BY  authorization-type
}
```

6.9.4.2 Behavior

The Physician class does not define any special methods.

6.9.4.3 Notifications

The Physician class does not generate any special notifications.

6.9.5 SessionTest class

Class: SessionTest
Description: "A SessionTest object contains vital signs information of a single patient that is recorded during a single examination or diagnostic treatment. This object contains vital signs metrics in the form of PM-Store objects. It also may contain information about equipment that was used for recording (in the form of relations to MDS and Ancillary objects)."
Superclass: Top
Subclasses: --
Name Binding: Handle
Registered As: MDC_MOC_SESSION_TEST

6.9.5.1 Attributes

The SessionTest class defines the attributes in Table 103.

Table 103—SessionTest class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Handle	MDC_ATTR_ID_ HANDLE	HANDLE	Name binding attribute.	M
St-Archive-Id	MDC_ATTR_ID_ SESS_TEST_ ARCHIVE	OCTET STRING		M
St-Archive-Name	MDC_ATTR_NAME_ SESS_TEST_ ARCHIVE	OCTET STRING	Example: study name.	M
St-Archive-Comments	MDC_ATTR_SESS_ TEST_ARCHIVE_ COMMENTS	OCTET STRING		O
Start-Time	MDC_ATTR_TIME_ START	AbsoluteTime		M
Stop-Time	MDC_ATTR_TIME_ STOP	AbsoluteTime		M
Protection	MDC_ATTR_ PROTECTION	ArchiveProtection		C

The SessionTest class defines in Table 104 the attribute groups or extensions to inherited attribute groups.

Table 104—SessionTest class attribute groups

Attribute group	Attribute group ID	Group elements
Archival Attribute Group	MDC_ATTR_GRP_ARCHIVE	from SessionTest: Handle, St-Archive-Id, St-Archive-Name, St-Archive-Comments, Start-Time, Stop-Time, Protection

6.9.5.2 Behavior

The SessionTest class does not define any special methods.

6.9.5.3 Notifications

The SessionTest class does not generate any special notifications.

6.9.6 SessionNotes class

Class: SessionNotes
Description: "A SessionNotes object contains diagnostic data, patient care details, and treatment-related information in the form of textual data."
Superclass: Top
Subclasses: --
Name Binding: Handle
Registered As: MDC_MOC_SESSION_NOTES

6.9.6.1 Attributes

The SessionNotes class defines the attributes in Table 105.

Table 105—SessionNotes class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Handle	MDC_ATTR_ID_HANDLE	HANDLE	Name binding attribute.	M
Sn-Id	MDC_ATTR_ID_SESS_NOTES_ARCHIVE	OCTET STRING		M
Sn-Name	MDC_ATTR_NAME_SESS_NOTES_ARCHIVE	OCTET STRING		M
Sn-Comments	MDC_ATTR_SESS_NOTES_ARCHIVE_COMMENTS	OCTET STRING		O
Start-Time	MDC_ATTR_TIME_START	AbsoluteTime		M
Stop-Time	MDC_ATTR_TIME_STOP	AbsoluteTime		M
Findings	MDC_ATTR_FINDINGS	OCTET STRING		O
Diagnostic-Codes	MDC_ATTR_CODE_DIAGNOSTIC	ExtNomenRefList	Diagnostic codes are specified in a nomenclature scheme not defined in this standard.	M
Diagnosis-Description	MDC_ATTR_DESC_DIAGNOSTIC	OCTET STRING		O
Procedure-Code	MDC_ATTR_CODE_PROCEDURE	ExtNomenRefList	Procedure codes are specified in a nomenclature scheme not defined in this standard.	M
Procedure-Description	MDC_ATTR_DESC_PROCEDURE	OCTET STRING		O
Protection	MDC_ATTR_PROTECTION	ArchiveProtection		C

The SessionNotes class defines in Table 106 the attribute groups or extensions to inherited attribute groups.

Table 106—SessionNotes class attribute groups

Attribute group	Attribute group ID	Group elements
Archival Attribute Group	MDC_ATTR_GRP_ARCHIVE	from SessionNotes: Handle, Sn-Id, Sn-Name, Sn-Comments, Start-Time, Stop-Time, Findings, Diagnostic-Codes, Diagnosis-Description, Procedure-Code, Procedure-Description, Protection

The following ASN.1 data type definitions apply:

```
ExtNomenRefList ::= SEQUENCE OF ExtNomenRef
```

6.9.6.2 Behavior

The SessionNotes class does not define any special methods.

6.9.6.3 Notifications

The SessionNotes class does not generate any special notifications.

6.10 Patient package

6.10.1 PatientDemographics class

Class: PatientDemographics
Description: "A PatientDemographics object contains minimal patient information as required by medical devices."
Superclass: Top
Subclasses: --
Name Binding: Handle
Registered As: MDC_MOC_PT_DEMOG

6.10.1.1 Attributes

The PatientDemographics class defines the attributes in Table 107.

Table 107—PatientDemographics class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Handle	MDC_ATTR_ID_HANDLE	HANDLE	Name binding attribute.	M
Pat-Demo-State	MDC_ATTR_PT_DEMOG_ST	PatDemoState	As a container, this object has a state.	M
Patient-Id	MDC_ATTR_PT_ID	OCTET STRING		O
Name	MDC_ATTR_PT_NAME	OCTET STRING		O

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Given-Name	MDC_ATTR_PT_NAME_GIVEN	OCTET STRING		O
Family-Name	MDC_ATTR_PT_NAME_FAMILY	OCTET STRING		O
Middle-Name	MDC_ATTR_PT_NAME_MIDDLE	OCTET STRING		O
Birth-Name	MDC_ATTR_PT_NAME_BIRTH	OCTET STRING	Maiden name.	O
Title-Name	MDC_ATTR_PT_NAME_TITLE	OCTET STRING	Example: Professor.	O
Sex	MDC_ATTR_PT_SEX	PatientSex		O
Race	MDC_ATTR_PT_RACE	PatientRace		O
Patient-Type	MDC_ATTR_PT_TYPE	PatientType		O
Date-Of-Birth	MDC_ATTR_PT_DOB	Date		O
Patient-Gen-Info	MDC_ATTR_PT_GEN_INFO	OCTET STRING	Textual patient-related information.	O
Patient-Age	MDC_ATTR_PT_AGE	PatMeasure	For neonatal, e.g., in hours or in weeks.	O
Gestational-Age	MDC_ATTR_PT_AGE_GEST	PatMeasure	For neonatal.	O
Patient-Height	MDC_ATTR_PT_HEIGHT	PatMeasure		O
Patient-Weight	MDC_ATTR_PT_WEIGHT	PatMeasure		O
Patient-Birth-Length	MDC_ATTR_PT_BIRTH_LENGTH	PatMeasure	For neonatal.	O
Patient-Birth-Weight	MDC_ATTR_PT_BIRTH_WEIGHT	PatMeasure	For neonatal.	O
Mother-Patient-Id	MDC_ATTR_ID_PT_MOTHER	OCTET STRING	For neonatal.	O
Mother-Name	MDC_ATTR_PT_NAME_MOTHER	OCTET STRING	For neonatal.	O
Patient-Head-Circumference	MDC_ATTR_CIRCUM_HEAD	PatMeasure		O
Patient-Bsa	MDC_ATTR_PT_BSA	PatMeasure	Body surface area; can be calculated.	O
Patient-Lbm	MDC_ATTR_PT_LBM	PatMeasure	Lean body mass; used for drug dosage calculations.	O
Bed-Id	MDC_ATTR_ID_BED	OCTET STRING		O
Diagnostic-Info	MDC_ATTR_DIAGNOSTIC_INFO	OCTET STRING	Free text for diagnosis.	O
Diagnostic-Codes	MDC_ATTR_CODE_DIAGNOSTIC	ExtNomenRefList	Diagnostic codes are specified in a nomenclature scheme not defined in this standard.	O
Admitting-Physician	MDC_ATTR_PHYSICIAN_ADMIT	OCTET STRING	For ICU.	O

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Attending-Physician	MDC_ATTR_PHYSICIAN_ATTEND	OCTET STRING	For ICU.	O
Date-Of-Procedure	MDC_ATTR_PROCEDURE_DATE	Date	For operating room (OR).	O
Procedure-Description	MDC_ATTR_DESC_PROCEDURE	OCTET STRING	For OR.	O
Procedure-Codes	MDC_ATTR_CODE_PROCEDURE	ExtNomenRefList	For OR; procedure codes are specified in a nomenclature scheme not defined in this standard.	O
Anaesthetist	MDC_ATTR_ANAESTHETIST	OCTET STRING	For OR.	O
Surgeon	MDC_ATTR_SURGEON	OCTET STRING	For OR.	O

The PatientDemographics class defines in Table 108 the attribute groups or extensions to inherited attribute groups.

Table 108 —PatientDemographics class attribute groups

Attribute group	Attribute group ID	Group elements
PatientDemographics Attribute Group	MDC_ATTR_GRP_PT_DEMOG	from PatientDemographics: Handle, Pat-Demo-State, Patient-Id, Name, Given-Name, Family-Name, Middle-Name, Birth-Name, Title-Name, Sex, Race, Patient-Type, Date-Of-Birth, Patient-Gen-Info, Patient-Age, Gestational-Age, Patient-Height, Patient-Weight, Patient-Birth-Length, Patient-Birth-Weight, Mother-Patient-Id, Mother-Name, Patient-Head-Circumference, Patient-Bsa, Patient-Lbm, Bed-Id, Diagnostic-Info, Diagnostic-Codes, Admitting-Physician, Attending-Physician, Date-Of-Procedure, Procedure-Description, Procedure-Codes, Anaesthetist, Surgeon

The following ASN.1 data type definitions apply:

```
--
-- State of the PatientDemographics object
--
PatDemoState ::= INT-U16 {
      empty(0),
      pre-admitted(1),
      admitted(2),
      discharged(8)
}

--
-- Patient demographics measured value
--
PatMeasure ::= SEQUENCE {
      value         FLOAT-Type,
```

```
        m-unit      OID-Type            -- code for units of measure
    }

    --
    -- Patient sex according to ISO/IEC 5218
    --
    PatientSex ::= INT-U16 {
        sex-unknown(0),
        male(1),
        female(2),
        sex-unspecified(9)
    }

    --
    -- Patient-Type attribute
    --
    PatientType ::= INT-U16 {
        pt-unspecified(0),
        adult(1),
        pediatric(2),
        neonatal(3)
    }

    --
    -- Patient race according to the Standard Communications Protocol [for computer-assisted]
    -- Electrocardiography(SCP ECG) (see CEN EN 1064)
    --
    PatientRace ::= INT-U16 {
        race-unspecified(0),
        race-caucasian(1),
        race-black(2),
        race-oriental(3)
    }
```

6.10.1.2 Behavior

The PatientDemographics class defines the methods in Table 109.

Table 109—PatientDemographics instance methods

Action	Mode	Action ID	Action parameter	Action result
Discharge-Patient	Confirmed	MDC_ACT_DISCH_PT		PatDemoState
Admit-Patient	Confirmed	MDC_ACT_ADMIT_PT	AdmitPatInfo	PatDemoState
Pre-Admit-Patient	Confirmed	MDC_ACT_PRE_ADMIT_PT	AdmitPatInfo	PatDemoState

The following ASN.1 data type definitions apply:

```
    --
    -- Admit-Patient method
    --
    AdmitPatInfo ::= AttributeList
```

6.10.1.3 Notifications

The PatientDemographics class defines the events in Table 110.

Table 110 —PatientDemographics events

Event	Mode	Event ID	Event parameter	Event result
Patient-Demographics-Modified	Confirmed/Unconfirmed	MDC_NOTI_PT_DEMOG_MOD	AttributeList	—
Patient-Demographics- State-Change	Confirmed/Unconfirmed	MDC_NOTI_PT_DEMOG_ST_MOD	AttributeList	—

7. Service model for communicating systems

7.1 General

This clause defines the basic application layer services provided by communicating systems that comply with this standard. The services are used by application processes to exchange vital signs information and commands for device and measurement control.

7.2 Communicating systems

The communication architecture that is assumed here is based on the agent-manager concept found in ISO systems management. It is possible to distinguish three types of systems that communicate and process vital signs information:

1. Vital signs information agent (i.e., an agent system that provides vital signs information in the form of managed medical objects)
2. Vital signs information manager (i.e., a manager system that consumes and acts upon vital signs information in the form of managed medical objects)
3. Vital signs information hybrid system (i.e., a system that both provides and consumes vital signs information)

NOTE—The term vital signs refers to the scope of this standard, not to the type or timeliness of information. Archived vital sign information can, for example, be provided on a vital signs information agent to supply manager applications with the requested data using remote database access.

A single communication application may consist of two or more of the above-mentioned systems.

Example: A central arrhythmia review system may consist of an ECG monitor (i.e., a vital signs information agent), an arrhythmia computer (i.e., a vital signs information hybrid system), and a central display and storage device (i.e., a vital signs information manager).

7.3 General service model overview

The range of devices, system configurations, and applications in the scope of this standard (i.e., vital signs) is very wide. From a simple device providing a single numerical

measurement to a system that consists of a number of dynamically reconfigurable measurement and processing devices, there is a large variation in complexity. Over time, it is likely that new classes providing specific functionality will have to be added to the DIM to cope with ongoing developments in the field of medical devices and measurements.Therefore, specialized messages for all possible vital signs and each possible application cannot be defined without causing penalties for small-scale devices and difficulties in the future maintenance of this standard and any implementations that are based on it. These obstacles necessitate the definition of a generalized service model that is largely independent of the DIM and does not require modification if new classes are needed.

NOTE—ISO/IEEE 11073-20601 defines protocols that are an alternative to this standard. ISO/IEEE 11073-20601 is tailored specifically for personal health devices, devices which are typically smaller and simpler than the point-of-care devices with which this standard is primarily concerned.

The DIM is strictly based on object-oriented methodology. It defines vital signs information in the form of classes and object containment hierarchies (i.e., managed medical objects). The classes each have identifiers, attributes, and methods.

The service model for communicating systems provides access to these managed medical objects by means of basic object management services that are independent of specific information object definitions. Such object management services make it possible to extend the information model by adding additional classes in subsequent standards without affecting the service model.

General object management services as defined in this standard are conceptually based on OSI system management in general (i.e., ISO/IEC 10040, the ISO/IEC 10164 family of standards) and specifically on the ISO/IEC common management information service element (CMISE) (i.e., ISO/IEC 9595).

NOTE—Classes that provide extended management services as defined in ISO/IEC 10164 family of standards are defined in the DIM, in particular in the ExtendedServices Package. Extended services defined by objects are invoked by the general object management services defined in this clause. Unless otherwise noted, in this clause the term services refers to the application layer services defined here.

The services enable the exchange of information about managed medical objects defined in the DIM between two peer entities. In a communicating system, the services are provided by the CMDISE. This name is chosen to make the functional similarity to CMISE obvious, but at the same time leave the definition of a cost-/performance-improved service model implementation to the interoperability work item instead of requiring a ISO/IEC CMIP-conformant implementation.

Service calls shall be mapped to a protocol, the Common Medical Device Information Protocol (CMDIP), the definition of which is outside the scope of this standard. The CMDIP is defined in ISO/IEEE 11073-20101.

ISO/IEC CMIP (i.e., ISO/IEC 9596-1) is considered a valid protocol implementation of the generalized object management services provided by the CMDISE defined in this standard. Additional (e.g., cost and/or performance optimized) implementations are explicitly not excluded and may be found in dependent standard(s).

7.3.1 Conceptual architecture of communicating systems

The conceptual architecture of communicating devices for which this standard provides class definitions is expressed in Figure 17. This figure shows how the services provided by CMDISE relate to class definitions defined in this standard and to applications using these class definitions.

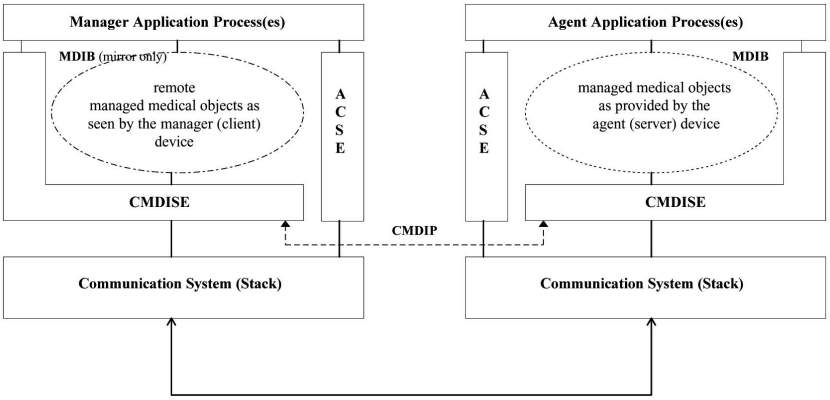

Figure 17—**Conceptual communicating systems architecture**

The application processes shown in Figure 17 are service users; the CMDISE and the ACSE are service providers.

This architectural model shows the following components that are used or referenced in this standard:

- A CMDISE, which provides the services defined in Clause 7 to application processes. The services are mapped to messages defined by the CMDIP.
- ACSE, which provides services to establish logical connections between MDSs. ACSE and the corresponding protocol are defined in ISO/IEC 8649 and ISO/IEC 8650-1. It provides service primitives for the following:
  - Requesting and accepting an association
  - Releasing an association and accepting the release
  - Association abort in case of a failure
- A standardized MDIB that contains managed medical object instances as defined in this standard (i.e., in the DIM).
- A standardized communication system (i.e., communication stack or profile, defining Layer 1 to Layer 6 of the OSI reference model). In addition to basic communication, this system may also provide services for synchronization between multiple devices, fragmentation of large messages, flow control, etc.

NOTE—The definition of association control and lower layer communication profiles is outside the scope of this standard. They are shown in Figure 17 for completeness only.

The manager system may use CMDISE services to build and to maintain a local copy of the agent's MDIB. As a conceptual illustration, Figure 17 shows a mirror of the agent's MDIB in the manager system. Note that object manipulation is always carried out by agent application processes.

An agent application populates and updates objects and attribute values in the agent MDIB. Applications running in the manager system may access the local copy of the MDIB in the manager for efficient data access (but services always act upon object instances in the agent's MDIB).

Objects in the MDIB can be remotely accessed only by using CMDISE services.

NOTE—Application access to the MDIB inside a specific system is a local implementation issue and is outside the scope of this standard. The actual implementation of the MDIB is also a local implementation issue and, therefore, outside the scope of this standard.

The following groups of services for the management of medical information are defined in this standard:

1. *Operational services*: This standard defines the following operational services on managed medical objects:
    1) Retrieve attribute value
    2) Modify attribute value
    3) Invoke instance methods
    4) Create and delete object instances
2. *Notification services*: This standard defines the following service that makes it possible to convey event notifications between communicating systems:
    1) Report asynchronous events that occurred within an object
3. *Services used by a manager system*: The manager system (i.e., client) invokes operational services to determine the agent (i.e., server) configuration, to retrieve medical object attribute values (e.g., measurements), and to control the agent. The manager system responds to notification services, if required, by providing an acknowledgment.
4. *Services used by an agent system*: The agent system invokes notification services to report the occurrence of defined events. The agent system responds to operational services by providing a result.
5. *Services used by a hybrid system*: A hybrid system invokes both operational and notification services as appropriate for a particular application. A hybrid system responds to both operational and notification services.

7.4 General object management services definition

Communicating systems that comply with the definitions in this standard provide or make use of the object management services defined in this subclause. The extent to which these

services are used by any particular communicating system depends on its role and its scope. Classes defined in the DIM specify the extent to which object instances use these services and the extent to which they can be accessed (i.e., controlled) by these services.

Service parameters defined in this subclause represent minimum requirements for communicating systems. In other words, an implementation of the service model may add extended functionality (e.g., authentication, access control, extended object selection) that requires additional parameters on top of the definitions in this standard. A protocol definition shall define the actual parameter data types and their usage in protocol messages.

7.4.1 EVENT REPORT service

The EVENT REPORT service is used to report an event about a managed object instance. The service may be used in confirmed mode or in unconfirmed mode. In confirmed mode, an EVENT REPORT service call requires a response.

> **Example**: An SpO2 monitor (i.e., an agent in a data logger application) may detect a transducer failure. The application process uses the EVENT REPORT service provided by the CMDISE to notify an associated manager of this technical alarm condition.

Unlike all other management services, the EVENT REPORT service, as a notification service, is initiated by an agent application process; the manager application process is the receiver and responder.

The EVENT REPORT service has the parameters in Table 111.

Table 111—EVENT REPORT service parameters

Parameter	Description
Invoke Identifier	Unique ID (e.g., a sequence number) assigned to a specific instance of the service so that it can be distinguished from other service invocations that a service provider may have in progress.
Mode	Confirmed or unconfirmed; confirmed mode requires a response.
Object Class	Identifies class of the object that generates the event (with the values defined in nomenclature/dictionary).
Object Instance	Identifies instance of the object that generates the event.
Event Time	Time the event was generated.
Event Type	Identifies the type of event (with the values defined in nomenclature/dictionary).
Event Information	(Optional) Additional information about the event, as defined by the Event Type parameter. The Event Information parameter is defined by the object that generates the event.

The confirmed EVENT REPORT service returns a response that has the parameters in Table 112.

Table 112—EVENT REPORT service result parameters

Parameter	Description
Invoke Identifier	Returns the unique Invoke Identifier parameter of the EVENT REPORT service so that the response can be related to the request.
Object Class	Same value as in EVENT REPORT service (optional).
Object Instance	Same value as in EVENT REPORT service (optional).
Current Time	(Optional).
Event Type	Same value as in EVENT REPORT service (optional).
Event Reply Information	(Optional) Additional information, as defined by the Event Type parameter.

If the EVENT REPORT service call cannot be processed, an error is returned indicating the type of failure.

NOTE— Error responses and error values are described in ISO/IEEE 11073-20101.

7.4.2 GET service

The GET service allows the retrieval of attribute data from managed object instances. The GET service is always used in confirmed mode. The GET service response contains the requested data (or an error notification).

Example: A data storage manager application may use the GET service to retrieve the serial number and revision information from a connected measurement device.

A manager process invokes the GET service (i.e., sends a GET service request message) to retrieve one, several, or all attributes of a selected managed object instance in an agent. The GET service result returns a list containing the requested attribute values.

The GET service has the parameters in Table 113.

Table 113—GET service parameters

Parameter	Description
Invoke Identifier	Unique ID (e.g., a sequence number) assigned to a specific instance of the service.
Object Class	Identifies class of the object that contains the requested attributes (i.e., values defined in nomenclature/dictionary).
Object Instance	Identifies instance of the object that contains the requested attributes.
Attribute Identifier List	List of attribute IDs (i.e., values defined in nomenclature/dictionary) for which values are to be retrieved.

The GET service returns a response that has the parameters in Table 114.

Table 114—GET service result parameters

Parameter	Description
Invoke Identifier	Returns the unique Invoke Identifier parameter of the GET service.
Object Class	Same value as in GET service.
Object Instance	Same value as in GET service.
Attribute List	A list of attribute ID–attribute value pairs.

If the GET service call cannot be processed, an error is returned indicating the type of failure.

7.4.3 SET service

The SET service allows the modification of attribute data contained in managed object instances. The SET service may be used in confirmed mode or in unconfirmed mode. In confirmed mode, a SET service call requires a response.

Example: A central computer may use the SET service to set the ~~current date and time in a device that has been newly connected to a network~~value of the Scan-List attribute of a CfgScanner object.

The manager process invokes the SET service (i.e., sends a SET service request message) to modify one or several attributes of a selected object instance in an agent. For each attribute that is to be modified, the request contains the attribute ID, a modify operator (to select whether the attribute value should be replaced, added to a list, deleted from a list, or set to a default value), and (optionally) the attribute value.

The SET service has the parameters in Table 115.

Table 115—SET service parameters

Parameter	Description
Invoke Identifier	Unique ID (e.g., a sequence number) assigned to a specific instance of the service.
Mode	Confirmed or unconfirmed.
Object Class	Identifies class of the object that contains the attributes to be modified (i.e., values defined in nomenclature/dictionary).
Object Instance	Identifies instance of the object that contains the attributes to be modified.
Modification List	List of (modify operator - attribute ID - attribute value) records. The modify operator may be replace, addValues, removeValues, setToDefault.

The confirmed SET service returns a response that has the parameters in Table 116.

Table 116—SET service result parameters

Parameter	Description
Invoke Identifier	Returns the unique Invoke Identifier parameter of the SET service.
Object Class	Same value as in SET service.
Object Instance	Same value as in SET service.
Attribute List	A list of attribute ID–attribute value pairs (optional).

If the SET service call cannot be processed, an error is returned indicating the type of failure.

7.4.4 ACTION service

The ACTION service makes it possible to invoke a predefined method (i.e., procedure) of a managed medical object. The ACTION service may be used in confirmed mode or in unconfirmed mode. In confirmed mode, an ACTION service call requires a response.

Example: ~~A monitoring system may use the ACTION service to start a calibration procedure on a measurement device~~A central computer may use the SetTime action service to set the current date and time in a device that has been newly connected to a network.

The definition of instance methods and the consequent object behavior are dependent on the specification of the managed object, not on the ACTION service. The class specification in the DIM defines all available instance methods that can be invoked by the ACTION service, along with their specific parameter and result data types.

The ACTION service has the parameters in Table 117.

Table 117—ACTION service parameters

Parameter	Description
Invoke Identifier	Unique ID (e.g., a sequence number) assigned to a specific instance of the service.
Mode	Confirmed or unconfirmed.
Object Class	Identifies class of the object that should execute the action (with the values defined in nomenclature/dictionary).
Object Instance	Identifies instance of the object that should execute the action.
Action Type	Identifies the type of the action (with the values defined in nomenclature/dictionary).
Action Information	Additional parameters for the action, as defined by the action type.

The confirmed ACTION service returns a response that has the parameters in Table 118.

Table 118—ACTION service result parameters

Parameter	Description
Invoke Identifier	Returns the unique Invoke Identifier parameter of the ACTION service.
Object Class	Same value as in ACTION service.
Object Instance	Same value as in ACTION service.
Action Type	Identifies the type of the action; same value as in ACTION service.
Action Reply	(Optional) Result of the action, as defined by the action type.

If the ACTION service call cannot be processed, an error is returned indicating the type of failure.

7.4.5 CREATE service

The CREATE service is used to create a new instance of a managed medical object. Attributes of the new object can be specified when using this service. The CREATE service is always used in confirmed mode and requires a response.

> **Example**: A data logger application may use the CREATE service to create an extended service object (e.g., a Scanner object) in a monitoring measurement agent (i.e., the agent system). This scanner processes all numerical measurement data and sends a report (i.e., event report) to the charting application every minute.

The CREATE service does not permit the creation of instances of arbitrary objects in the MDIB of the agent system. A system that complies with the definitions in this standard has to specify which object classes can be dynamically created.

The CREATE service has the parameters in Table 119.

Table 119—CREATE service parameters

Parameter	Description
Invoke Identifier	Unique ID (e.g., a sequence number) assigned to a specific instance of the service.
Object Class	Identifies class of the object that should be created (with the values defined in nomenclature/dictionary).
Superior Object Class	Identifies class of the superior (with respect to containment hierarchy) object instance.
Superior Object Instance	Identifies instance of the superior (with respect to containment hierarchy) object.
Attribute List	A list of attribute ID–attribute value pairs (optional) to set the initial values of attributes.

The CREATE service returns a response that has the parameters in Table 120.

Table 120 —CREATE service result parameters

Parameter	Description
Invoke Identifier	Returns the unique Invoke Identifier parameter of the CREATE service.
Object Class	Same value as in CREATE service.
Object Instance	Assigned by CMDISE according to object name binding.

If the CREATE service call cannot be processed, an error is returned indicating the type of failure.

7.4.6 DELETE service

The DELETE service is used to delete instances of managed objects. The delete service is always used in confirmed mode and requires a response.

Example: When the data logger application from the previous example no longer needs the measurement data provided by the agent system, it uses the DELETE service to delete the extended services object (i.e., the Scanner object instance).

The DELETE service does not permit the deletion of instances of arbitrary objects in the MDIB of the agent system. A system that complies with the definitions in this standard has to specify which object classes can be dynamically deleted.

The DELETE service has the parameters in Table 121.

The DELETE service returns a response that has the parameters in Table 122.

If the DELETE service call cannot be processed, an error is returned indicating the type of failure.

Table 121 —DELETE service parameters

Parameter	Description
Invoke Identifier	Unique ID (e.g., a sequence number) assigned to a specific instance of the service.
Object Class	Identifies class of the object that should be deleted (with the values defined in nomenclature/dictionary).
Object Instance	Identifies instance of the object that should be deleted.

Table 122 —DELETE service result parameters

Parameter	Description
Invoke Identifier	Returns the unique Invoke Identifier parameter of the DELETE service.
Object Class	Same value as in DELETE service.
Object Instance	Same value as in DELETE service.

8. MDIB nomenclature

The set of objects occurring in any device of the communicating system as described in the DIM forms the MDIB. Each instantiation of a class defined in the DIM needs a unique identification. The total set of terms forms the MDIB nomenclature or the data dictionary. Because a large number of instantiations exist, a structured identification scheme, i.e., a nomenclature, is necessary. The nomenclature for the MDIB comprises several thousand terms concerning the object-oriented modeling elements, demographic patient data, device descriptions, measurement values, measurement methods, measurement locations, alarm information, etc. It is open for extensions due to progress in medicine and technology without the need to change structures and terms within an established set of terms.

The nomenclature also supports the development of a dictionary that is language independent with a coding scheme for easy and fast computer access.

The MDIB nomenclature(i.e., data dictionary) is presented in ISO/IEEE 11073-10101. It contains the terms (i.e., systematic names), descriptions, and codes for the following target categories:

- Object-oriented modeling elements resulting from the DIM
- Medical devices and device systems
- Units of measurements
- Metrics (measurements and enumerations)
- Body sites (i.e., specifications for measurement locations)
- Alerts
- External nomenclatures

Each of the respective clauses starts with a detailed description on how to build the systematic name for the target category concerned.

9. Conformance model

9.1 Applicability

It is expected that this standard will be used together with other base standards or referenced by other standards in the ISO/IEEE 11073 family of standards to define applications (e.g., for the exchange of vital signs measurement databases) or to define functional communication profiles (e.g., medical device interoperability profiles).

Such additional specifications or standards are necessary to fully enable an implementation or a system using this standard.

It is possible for an implementation or a system to conform to the following element of this standard, which contains concrete definitions:

— DIM class hierarchy and class definitions (i.e., attributes, notifications, methods, and data type definitions).

However, conformance to this element alone does not provide interoperability of applications or medical devices.

Standards for specific applications or functional communication profiles are expected to define an appropriate conformance model that includes specific conformance requirements for this standard on vital signs representation. They also need to define additional conformance criteria for semantic and dynamic behavior of the implementation, which are out of the scope of this standard.

Conformance to definitions of this standard is specified at the appropriate application interface or system interface only. Only the behavior at this interface is considered for conformance. Implementation details that cannot be perceived externally are not subject to conformance specification.

Example: A communicating medical device uses the classes and type definitions as defined in this standard to distribute data to other devices. It may be compliant to this standard, even if it does not use any object-oriented implementation internally.

9.2 Conformance specification

This standard on vital signs representation offers a high degree of flexibility in how the model is applied for a certain medical device, particularly in the following areas:

— Information model of a specific device
— Use of attributes, value ranges, and access
— Use of extended communication services (i.e., scanners), scan periods, and scanner configurability

To support interoperability of applications and systems, an implementation based on this standard shall provide specific details about the way that the definitions of this standard are applied.

These specifications have to be provided in form of a set of implementation conformance statements (ICSs). An ICS is a form of data sheet that discloses details of a specific implementation and specifies which features are provided. Specific applications or functional communication profiles that are based on this standard shall define more specific conformance requirements in addition to or as a replacement of the ICS defined here.

NOTE— The ICSs defined in 9.3 provide understanding of the details of an implementation. However, they are not sufficient to provide interoperability of devices or applications. For such interoperability, additional specifications (e.g., timing, latencies, system loading assumptions) shall be taken into account. These specifications are not within the scope of this standard.

9.3 ICSs

9.3.1 General format

The ICSs have to be supplied in the form of tables. Templates for these ICS tables are given in 9.3.2 through 9.3.7 The tables have to be filled out and provided as an overall conformance statement document.

Generally the column headers of an ICS table contain the following information:

- Index, which is an identifier (e.g., a number) of a specific feature.
- Feature, which briefly describes the characteristic for which a conformance statement must be made.
- Reference, which is a reference to the definition of the feature (may be empty).
- Status, which specifies the conformance requirement (i.e., the requirements for a conforming implementation regarding the feature). In some cases, this standard does not specify conformance requirements, but still wants a definition of the status of a particular feature.
- Support, which is filled out by the implementer and specifies the characteristics of the feature in the implementation.
- Comment, which contains additional information provided by the implementer.

The value of the Status and Support columns are permitted to range from simple to complex entries. Examples of simple values are as follows:

m mandatory
o optional
x prohibited
c conditional
n/a not applicable

More complex expressions or specific lists of items are defined in the specific ICS table.

9.3.2 General ICS

In a top-level General ICS, the implementer specifies the versions/revisions that are supported by the implementation as well as some high-level system behavior definitions.

Table 123 shows the General ICS.

Table 123 —General ICS

Index	Feature	Reference	Status	Support	Comment
GEN-1	Implementation Description	—	Identification of the device/application. Description of functionality.		

Index	Feature	Reference	Status	Support	Comment
GEN-2	Standard Document Revision	(Standard documents)	(Set of existing revisions)	(Set of supported revision)	
GEN-3	Conformance Deviation	—	Provides information about possible deviations from the DIM (e.g., nonstandard attributes, objects)	(Set of deviations)	
GEN-4	Object Containment Tree	5.2	Provides object containment diagram showing relations between object instances used by the application. A conforming implementation uses object relations only as defined in the DIM.		
GEN-5	Nomenclature Revision	(Standard documents)	(Set of existing revisions)	(Set of supported revision)	
GEN-6	Use of other Nomenclature Schemes	—	Are nomenclature codes from other standard coding schemes used in the implementation?	Yes/No (If yes: list of other nomenclatures)	Note that the use of other nomenclatures severely impacts interoperability.
GEN-7	Data Structure Encoding	—	—	Description of encoding method for ASN.1 data structures	
GEN-8	Dynamic Object Instances	—	Is the set of object instances at run-time static or dynamic?	Static/Dynamic	
GEN-9	Use of Private Objects	—	Does the implementation use objects that are not defined in the DIM?	Yes/No [If yes: explain in DIM MOC ICS (see 9.3.4)]	
GEN-10	Use of Private Nomenclature Extensions	—	Does the implementation use private extensions to the nomenclature? Private nomenclature extensions are allowed only if the standard nomenclature does not include the specific terms required by the application.	Yes/No (If yes: explain in the appropriate ICS)	
GEN-11	Communication Profile and Hardware	—	Description of communication profile and hardware requirements for interfacing (only applicable for communicating devices).		

Index	Feature	Reference	Status	Support	Comment
GEN-12	File Format and Storage Media	—	Description of file formats used for archiving vital signs data; definition of supported storage media (only applicable for archival applications).		
GEN-13	ACSE	ISO/IEC 8649 ISO/IEC 8650-1	Use of ACSE protocol (only applicable for communicating systems).	Specify use of optional fields in ACSE protocol data units (PDUs).	

For each implementation, one General ICS shall be provided.

9.3.3 Service Support ICS

The Service Support ICS defines which services that are defined in the service model are implemented. This ICS needs only be supplied for communicating devices.

Table 124 shows the Service Support ICS.

Table 124 —Service Support ICS

Index	Feature	Reference	Status	Support	Comment
SRV-1	GET Service	7.4.2	o		
SRV-2	SET Service	7.4.3	o		
SRV-3	Confirmed SET Service	7.4.3	o		
SRV-4	EVENT REPORT Service	7.4.1	m		
SRV-5	Confirmed EVENT REPORT Service	7.4.1	o		
SRV-6	ACTION Service	7.4.4	o		
SRV-7	Confirmed ACTION Service	7.4.4	o		
SRV-8	Create Service	7.4.5	o		
SRV-9	Delete Service	7.4.6	o		

The Support column of the completed table should define if the implementation invokes the service (e.g., sends a GET PDU), provides the service (e.g., processes a received GET PDU), or does not implement the service at all.

In addition, specific restrictions should be listed (e.g., if a specific service is restricted to objects of only one class).

9.3.4 DIM managed object class (MOC) ICS

The DIM MOC ICS defines which managed medical objects (not base classes) are used by the implementation. Table 125 is a template only. For each object supported by the implementation, one row shall be filled out.

Table 125—Template for DIM MOC ICS

Index	Feature	Reference	Status	Support	Comment
MOC-[1-*n*]	Object Name and OID	Reference to the clause in this standard where the object is defined	Implemented	Specify restrictions, e.g., maximum number of supported instances CREATE/DELETE services are supported.	

If the implementation uses private objects, these objects should also be specified in the DIM MOC ICS. A separate definition should be appended to the conformance statement that can be referenced in the Reference column.

The Support column should indicate specific restrictions about the object implementation. In particular, it shall indicate whether object instances can be dynamically created/deleted using the CREATE/DELETE service.

In addition to the DIM MOC ICS, an object containment diagram (i.e., class instance diagram) should be provided that allows reviewing the class hierarchy used by the implementation.

9.3.5 MOC Attribute ICS

For each supported object as defined in the DIM MOC ICS, a MOC Attribute ICS has to be provided that defines which attributes are used/supported by the implementation, including any inherited attributes. Table 126 is a template only.

Table 126—Template for MOC Attribute ICS

Index	Feature	Reference	Status	Support	Comment
ATTR-*x-n*	Attribute Name and Attribute ID	Reference to the clause in this standard where the object is defined	m/o/c (see 6.1.2 for an explanation of these abbreviations)	Access (i.e., GET, GET-GRP, SET, SCAN, SCAN-GRP, ER, CRER; see third paragraph after this table) Value ranges Additional restrictions Static/dynamic value	

The *x* in the Index column is the ID of the managed object for which the table is supplied (i.e., the index of the managed object as specified in the DIM MOC ICS). There is one separate table for each supported managed object.

The *n* in the Index column is just a serial number (1..m).

The attribute access specification fields in the Support column have to be specified if the implementation provides access services for attributes. (In other words, the fields are not needed for a plain storage format.) The fields have the following meanings:

GET	Attribute can be individually accessed by the GET service.
GET-GRP	Attribute can be accessed by the GET service as part of an attribute group.
SET	Attribute can be individually modified by the SET service.
SCAN	Attribute can be individually accessed by a Scanner object (individual scan list entry).
SCAN-GRP	Attribute can be accessed by a Scanner object (attribute group scan list entry).
ER	Attribute changes are communicated by event reports from the container object itself.
CR-ER	Attribute value is provided within the notification that announces the availability of the container object (object create notification).

The Support column should also contain attribute value ranges (if applicable), hints about specific restrictions for attribute access or attribute availability and information, and an indication if the attribute value is static or dynamic in the implementation.

NOTE— The attribute definition tables in this standard define a minimum mandatory set of attributes for each object.

9.3.6 MOC Behavior ICS

The MOC Behavior ICS specifies all implemented instance methods that can be invoked by the ACTION service. Table 127 is a template only. One table has to be provided for each object that supports special methods.

Table 127—Template for MOC Behavior ICS

Index	Feature	Reference	Status	Support	Comment
ACT-*x-n*	Method Name and Method ID	Reference to the clause in this standard where the instance method is defined		Specific restrictions	

The *x* in the Index column is the ID of the managed object for which the table is supplied (i.e., the index of the managed object as specified in the DIM MOC ICS). There is one separate table for each managed object that supports specific instance methods (i.e., actions).

The n in the Index column is just a serial number (1..*m*).

The Support column should specify any restrictions for the method.

9.3.7 MOC Notification ICS

The MOC Notification ICS specifies all implemented notifications (typically in form of the EVENT REPORT service) that are emitted by supported objects. Table 128 is a template only. One table has to be provided for each object that supports special object notifications.

Table 128—Template for MOC Notification ICS

Index	Feature	Reference	Status	Support	Comment
NOTI-*x-n*	Notification Name and Notification ID	Reference to the clause in this standard where the event is defined		Specific restrictions	

The *x* in the Index column is the ID of the managed object for which the table is supplied (i.e., the index of the managed object as specified in the DIM MOC ICS). There is one separate table for each managed object that supports specific object notifications (i.e., events).

The *n* in the Index column is just a serial number (1..*m*).

The Support column should specify any restrictions for the notification.

Annex A (informative) Bibliography

Bibliographical references are resources that provide additional or helpful material but do not need to be understood or used to implement this guide. Reference to these resources is made for informational use only. To aid the reader, notes in parentheses have been added to indicate where bibliographical references are mentioned in this guide.

[B1] CEN EN 1064, Medical informatics — Standard communication protocol — computer-assisted electrocardiography.[8]

[B2] FDA Unique Device Identification System, A Rule by the Food and Drug Administration, accessed September 17, 2018, https://www.federalregister.gov/articles/2013/09/24/2013-23059/unique-device-identification-system

[B3] IETF RFC 1155, Structure and Identification of Management Information for TCP/IP-Based Internets.[9]

[B4] Integrating the Healthcare Enterprise. IHE Patient Care Device (PCD) technical framework (volume 1, revision 7.0), accessed September 17, 2018, https://www.ihe.net/Technical_Frameworks/#pcd

[B5] ISO 639-1, Code for the representation of names of languages — Part 1: Alpha-2 code.[10]

[B6] ISO 639-2, Codes for the representation of names of languages — Part 2: Alpha-3 code.

[B7] ISO/IEC 646, Information technology — ISO 7-bit coded character set for information interchange.

[B8] ISO 3166-1, Codes for the representation of names of countries and their subdivisions — Part 1: Country codes.

[B9] ISO 3166-2, Codes for the representation of names of countries and their subdivisions — Part 2: Country subdivision code.

[B10] ISO 3166-3, Codes for the representation of names of countries and their subdivisions — Part 3: Code for formerly used names of countries.

[B11] ISO 8601, Data elements and interchange formats — Information interchange — Representation of dates and times.

[B12] ISO 12052, Health informatics -- Digital imaging and communication in medicine (DICOM) including workflow and data management.

[8] CEN publications are available from the European Committee for Standardization (CEN), 36, rue de Stassart, B-1050 Brussels, Belgium (http://www.cenorm.be).
[9] Internet requests for comment (RFCs) are available from the Internet Engineering Task Force at http://www.ietf.org/.
[10] ISO publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse (http://www.iso.ch/). ISO publications are also available in the United States from the Sales Department, American National Standards Institute, 25 West 43rd Street, 4th Floor, New York, NY 10036, USA (http://www.ansi.org/).

[B13] ISO 15225, Nomenclature — Specification for a nomenclature system for medical devices for the purpose of regulatory data exchange.

[B14] ISO/IEC 2022, Information technology — Character code structure and extension techniques.

[B15] ISO/IEC 5218, Information technology — Codes for the representation of human sexes.

[B16] ISO/IEC 7498-1, Information technology — Open systems interconnection — Basic reference model — Part 1: The basic model.

[B17] ISO/IEC 7498-2, Information processing systems — Open systems interconnection — Basic reference model — Part 2: Security architecture.

[B18] ISO/IEC 7498-3, Information processing systems — Open systems interconnection — Basic reference model — Part 3: Naming and addressing.

[B19] ISO/IEC 7498-4, Information processing systems — Open systems interconnection — Basic reference model — Part 4: Management framework.

[B20] ISO/IEC 8649, Information processing systems — Open systems interconnection — Service definition for the Association Control Service Element.

[B21] ISO/IEC 8650-1, Information technology — Open systems interconnection — Connection-Oriented Protocol for the Association Control Service Element — Part 1: Protocol.

[B22] ISO/IEC 8650-2, Information technology — Open systems interconnection — Protocol Specification for Association Control Service Element — Part 2: Protocol Implementation Conformance Statements (PICS) proforma.

[B23] ISO/IEC 8824-2, Information technology — Abstract Syntax Notation One (ASN.1) — Part 2: Information object specification.

[B24] ISO/IEC 8859-n, Information processing — 8-bit single-byte coded graphic character sets — Part 1 to Part 15: Various alphabets.

[B25] ISO/IEC 9545, Information technology — Open systems interconnection — Application layer structure.

[B26] ISO/IEC 9595, Information technology — Open systems interconnection — Common management information service definition.

[B27] ISO/IEC 9596-1, Information technology — Open systems interconnection — Common Management Information Protocol — Part 1: Specification.

[B28] ISO/IEC 10040, Information technology — Open systems interconnection — Systems management overview.

[B29] ISO/IEC 10164-1, Information technology — Open systems interconnection — Systems management — Part 1: Object management function.

[B30] ISO/IEC 10164-2, Information technology — Open systems interconnection — Systems management — Part 2: State management function.

[B31] ISO/IEC 10164-3, Information technology — Open systems interconnection — System management — Part 3: Attributes for representing relationships.

[B32] ISO/IEC 10164-4, Information technology — Open systems interconnection — Systems management — Part 4: Alarm reporting function.

[B33] ISO/IEC 10164-5, Information technology — Open systems interconnection — Systems management — Part 5: Event management function.

[B34] ISO/IEC 10164-6, Information technology — Open systems interconnection — Systems management — Part 6: Log control function.

[B35] ISO/IEC 10164-7, Information technology — Open systems interconnection — Systems management — Part 7: Security alarm reporting function.

[B36] ISO/IEC 10164-8, Information technology — Open systems interconnection — Systems management — Part 8: Security audit trail function.

[B37] ISO/IEC 10164-9, Information technology — Open systems interconnection — Systems management — Part 9: Objects and attributes for access control.

[B38] ISO/IEC 10164-10, Information technology — Open systems interconnection — Systems management — Part 10: Usage metering function for accounting purposes.

[B39] ISO/IEC 10164-11, Information technology — Open systems interconnection — Systems management — Part 11: Metric objects and attributes.

[B40] ISO/IEC 10164-12, Information technology — Open systems interconnection — Systems management — Part 12: Test management function.

[B41] ISO/IEC 10164-13, Information technology — Open systems interconnection — Systems management — Part 13: Summarization function.

[B42] ISO/IEC 10164-14, Information technology — Open systems interconnection — Systems management — Part 14: Confidence and diagnostic test categories.

[B43] ISO/IEC 10165-1, Information technology — Open systems interconnection — Structure of management information — Part 1: Management information model.

[B44] ISO/IEC 10165-2, Information technology — Open systems interconnection — Structure of management information — Part 2: Definition of management information.

[B45] ISO/IEC 10646-1, Information technology — Universal multiple-octet coded character set (UCS) — Part 1: Architecture and basic multilingual plane.

[B46] ISO/IEEE 11073-20601, Health informatics — Personal health device communication — Part 20601: Application profile — Optimized Exchange Protocol

1

2

3