INTERNATIONAL STANDARD

ISO/IEEE 11073-10201

First edition
2004-12-15

Health informatics — Point-of-care medical device communication —

Part 10201:
Domain information model

Informatique de santé — Communication entre dispositifs médicaux sur le site des soins —
Partie 10201: Modèle d'information du domaine

Reference number
ISO/IEEE 11073-10201:2004(E)

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

Sponsor

IEEE 1073™ Standard Committee

of the

IEEE Engineering in Medicine and Biology Society

Approved 24 June 2004

IEEE-SA Standards Board

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 object. All 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

Print: ISBN 0-7381-4089-9 SH95256
PDF: ISBN 0-7381-4090-2 SS95256

ISO Foreword

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

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

A pilot project between ISO and the IEEE has been formed to develop and maintain a group of ISO/IEEE standards in the field of medical devices as approved by Council resolution 43/2000. Under this pilot project, IEEE is responsible for the development and maintenance of these standards with participation and input from ISO member bodies.

Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. Neither ISO nor the IEEE shall be held responsible for identifying any or all such patent rights.

ISO/IEEE 11073-10201:2004(E) was prepared by IEEE 1073 Committee of the IEEE Engineering in Medicine and Biology Society.

IEEE Introduction

This introduction is not part of ISO/IEEE 11073-10201:2004(E), Health informatics — Point-of-care medical device communication — Part 10201: Domain information model.

ISO/IEEE 11073 standards enable communication between medical devices and external computer systems. They provide automatic and detailed electronic data capture of patient vital signs information and device operational data. The primary goals are to:

— Provide real-time plug-and-play interoperability for patient-connected medical devices

— Facilitate the efficient exchange of vital signs and medical device 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 all the clinician has to do is make the connection — the systems automatically detect, configure, and communicate without 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. The standards are especially targeted at acute and continuing care devices, such as patient monitors, ventilators, infusion pumps, ECG devices, etc. They comprise a family of standards that can be layered together to provide connectivity optimized for the specific devices being interfaced.

Notice to users

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 with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents or patent applications for which a license may be required by to implement an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Errata

Errata, if any, for this and all other standards can be accessed at the following URL: http://standards.ieee.org/reading/ieee/updates/errata/index.html. Users are encouraged to check this URL for errata periodically.

Interpretations

Current interpretations can be accessed at the following URL: http://standards.ieee.org/reading/ieee/interp/index.html.

Participants

At the time this standard was completed, the working group of the IEEE 1073 Standard Committee had the following membership:

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

When the IEEE-SA Standards Board approved this standard on 24 June 2004, it had the following membership:

Contents

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

1. Scope

Within the context of the ISO/IEEE 11073 family of standards, this standard addresses the definition and structuring of information that is communicated or referred to in communication between application entities.

This standard provides a common representation of all application entities present in the application processes within the various devices independent of the syntax.

The definition of association control and lower layer communication is outside the scope of this standard.

2. Normative references

The following referenced documents are indispensable for the application of this standard. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

CEN EN 1064, Medical informatics — Standard communication protocol — computer-assisted electrocardiography.[1]

CEN ENV 12052, Medical informatics — Medical imaging communication (MEDICOM).

IEEE Std 1073™, IEEE Standard for Medical Device Communications—Overview and Framework.[2]

[1]CEN publications are available from the European Committee for Standardization (CEN), 36, rue de Stassart, B-1050 Brussels, Belgium (http://www.cenorm.be).

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

IETF RFC 1155, Structure and Identification of Management Information for TCP/IP-Based Internets.[3]

ISO 639-1, Code for the representation of names of languages — Part 1: Alpha-2 code.[4]

ISO 639-2, Codes for the representation of names of languages — Part 2: Alpha-3 code.

ISO 3166-1, Codes for the representation of names of countries and their subdivisions — Part 1: Country codes.

ISO 3166-2, Codes for the representation of names of countries and their subdivisions — Part 2: Country subdivision code.

ISO 3166-3, Codes for the representation of names of countries and their subdivisions — Part 3: Code for formerly used names of countries.

ISO 8601, Data elements and interchange formats — Information interchange — Representation of dates and times.

ISO 15225, Nomenclature — Specification for a nomenclature system for medical devices for the purpose of regulatory data exchange.

ISO/IEC 646, Information technology — ISO 7-bit coded character set for information interchange.[5]

ISO/IEC 2022, Information technology — Character code structure and extension techniques.

ISO/IEC 5218, Information technology — Codes for the representation of human sexes.

ISO/IEC 7498-1, Information technology — Open systems interconnection — Basic reference model — Part 1: The basic model.

ISO/IEC 7498-2, Information processing systems — Open systems interconnection — Basic reference model — Part 2: Security architecture.

ISO/IEC 7498-3, Information processing systems — Open systems interconnection — Basic reference model — Part 3: Naming and addressing.

ISO/IEC 7498-4, Information processing systems — Open systems interconnection — Basic reference model — Part 4: Management framework.

ISO/IEC 8649, Information processing systems — Open systems interconnection — Service definition for the Association Control Service Element.

ISO/IEC 8650-1, Information technology — Open systems interconnection — Connection-Oriented Protocol for the Association Control Service Element — Part 1: Protocol.

[3]Internet requests for comment (RFCs) are available from the Internet Engineering Task Force at http://www.ietf.org/.

[4]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/).

[5]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/).

ISO/IEC 8650-2, Information technology — Open systems interconnection — Protocol Specification for Association Control Service Element — Part 2: Protocol Implementation Conformance Statements (PICS) proforma.

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

ISO/IEC 8824-2, Information technology — Abstract Syntax Notation One (ASN.1) — Part 2: Information object specification.

ISO/IEC 8859-*n*, Information processing — 8-bit single-byte coded graphic character sets — Part 1 to Part 15: Various alphabets.

ISO/IEC 9545, Information technology — Open systems interconnection — Application layer structure.

ISO/IEC 9595, Information technology — Open systems interconnection — Common management information service definition.

ISO/IEC 9596-1, Information technology — Open systems interconnection — Common Management Information Protocol — Part 1: Specification.

ISO/IEC 10040, Information technology — Open systems interconnection — Systems management overview.

ISO/IEC 10164-1, Information technology — Open systems interconnection — Systems management — Part 1: Object management function.

ISO/IEC 10164-2, Information technology — Open systems interconnection — Systems management — Part 2: State management function.

ISO/IEC 10164-3, Information technology — Open systems interconnection — System management — Part 3: Attributes for representing relationships.

ISO/IEC 10164-4, Information technology — Open systems interconnection — Systems management — Part 4: Alarm reporting function.

ISO/IEC 10164-5, Information technology — Open systems interconnection — Systems management — Part 5: Event management function.

ISO/IEC 10164-6, Information technology — Open systems interconnection — Systems management — Part 6: Log control function.

ISO/IEC 10164-7, Information technology — Open systems interconnection — Systems management — Part 7: Security alarm reporting function.

ISO/IEC 10164-8, Information technology — Open systems interconnection — Systems management — Part 8: Security audit trail function.

ISO/IEC 10164-9, Information technology — Open systems interconnection — Systems management — Part 9: Objects and attributes for access control.

ISO/IEC 10164-10, Information technology — Open systems interconnection — Systems management — Part 10: Usage metering function for accounting purposes.

ISO/IEC 10164-11, Information technology — Open systems interconnection — Systems management — Part 11: Metric objects and attributes.

ISO/IEC 10164-12, Information technology — Open systems interconnection — Systems management — Part 12: Test management function.

ISO/IEC 10164-13, Information technology — Open systems interconnection — Systems management — Part 13: Summarization function.

ISO/IEC 10164-14, Information technology — Open systems interconnection — Systems management — Part 14: Confidence and diagnostic test categories.

ISO/IEC 10165-1, Information technology — Open systems interconnection — Structure of management information — Part 1: Management information model.

ISO/IEC 10165-2, Information technology — Open systems interconnection — Structure of management information — Part 2: Definition of management information.

ISO/IEC 10646-1, Information technology — Universal multiple-octet coded character set (UCS) — Part 1: Architecture and basic multilingual plane.

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.

NEMA PS 3, Digital imaging and communications in medicine (DICOM).[6]

3. Definitions

For the purpose of this standard, the following definitions apply. *The Authoritative Dictionary of IEEE Standards Terms*, Seventh Edition, should be referenced for terms not defined in this clause.

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

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

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

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

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

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

[6]NEMA publications are available from Global Engineering Documents, 15 Inverness Way East, Englewood, Colorado 80112, USA (http://global.ihs.com/).

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

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

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

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

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

3.12 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.

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

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

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

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

3.17 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.

3.18 discrete parameter: Vital signs measurement that can be expressed as a single numeric or textual value.

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

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

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

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

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

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

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

3.26 information service element: Instances in the medical data information base (MDIB).

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

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

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

3.30 interoperability: Idealized scheme where medical devices of differing types, models, or manufacturers are capable of working with each other, whether connected to each other directly or through a communication system.

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

3.32 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.

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

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

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

3.36 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.

3.37 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 objects that are defined in this standard.

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

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

3.40 object attributes: Data that, together with methods, define an object.

3.41 object class: A descriptor used in association with a group of objects with similar properties (i.e., attributes), common behavior (i.e., operations), common relationships to other objects, and common semantics.

3.42 object diagram: Diagram showing connections between objects in a system.

3.43 object method: A procedure or process acting upon the attributes and states of an object class.

3.44 object-oriented analysis: Method of analysis where the problem domain is modelled in the form of objects and their interactions.

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

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

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

3.48 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.

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

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

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

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

3.53 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.

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

3.55 top object: The ultimate base class for all other objects belonging to one model.

3.56 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.

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

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

3.59 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.

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

4. 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
EEG	electroencephalogram
EBWW	eyeball and wristwatch
FSM	finite state machine
GMDN	Global Medical Device Nomenclature
GMT	Greenwich mean time
IANA	Internet Assigned Numbers Authority
ICS	implementation conformance statement
ICU	intensive care unit
ID	identifier or identification
LAN	local area network
LSB	least significant bit
MDIB	medical data information base
MDS	medical device system
MEDICOM	medical imaging communication
MIB or Mib	management information base
MOC	managed object class
OID	object identifier
OR	operating room
OSI	open systems interconnection
PC	personal computer
PDU	protocol data unit
PM	persistent metric
SCADA	supervisory control and data acquisition
SCP ECG	Standard Communications Protocol [for computer-assisted] Electrocardiography
SNMP	Simple Network Management Protocol
SNTP	Simple Network Time Protocol
UML	unified modeling language
UTC	coordinated universal time
VMD	virtual medical device
VMO	virtual medical object
VMS	virtual medical system

5. General requirements

The ISO/IEEE 11073 family of standards is intended to enable medical devices to interconnect and interoperate 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 information objects that can be accessed and manipulated using an object access service protocol.

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

The top-level user requirements are defined in IEEE Std 1073,[7] 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 information objects defined in the object-oriented model, including their attributes and methods.
- Define a service model for communicating medical devices that provides access to the information objects, 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.

6. DIM

6.1 General

6.1.1 Modeling concept

The DIM is an object-oriented model that consists of objects, 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. Objects defined in the information model are considered managed (here, medical) objects. For the most part, they are directly available to management (i.e., access) services provided by the common medical device information service element (CMDISE) as defined in this standard.

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

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 objects defined here represent generalized vital signs data and support information. Specialization of these objects is achieved by defining appropriate attributes. Object hierarchies and relations between objects are used to express device configuration and device capabilities.

> **Example:** A generalized object is defined to represent vital signs in the form of a real-time waveform. A set of object 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 6.1 shows the relation between managed medical objects, MDIB, CMDISE, application processes, and communication systems.

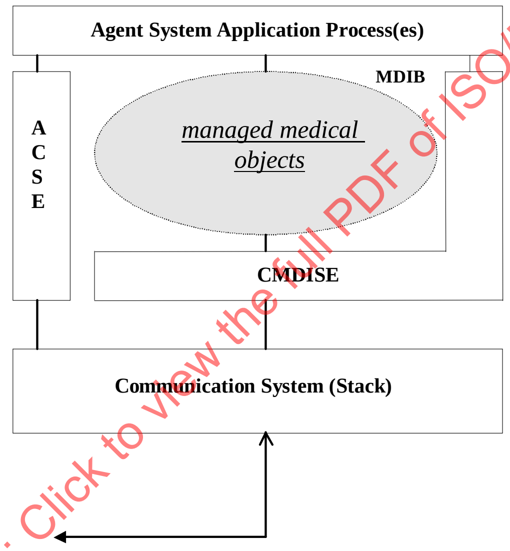
Agent System Application Process(es)
MDIB
A C S E
managed medical objects
CMDISE
Communication System (Stack)

Figure 6.1—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 6.2 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.

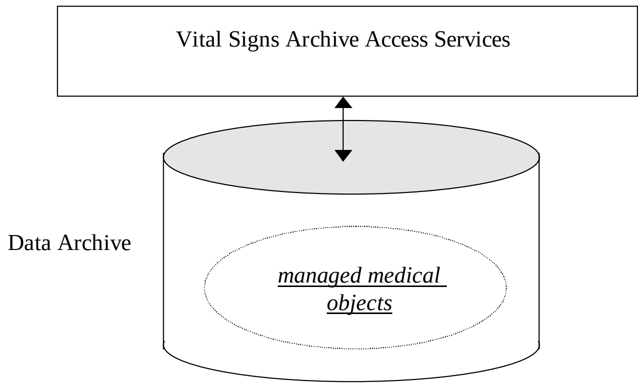

Figure 6.2—Managed medical objects in a vital signs archive

6.1.2 Scope of the DIM

6.1.2.1 General

Vital signs information objects 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.

6.1.2.2 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.

6.1.2.3 Archived vital signs

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

6.1.3 Approach

For the object-oriented modeling, the unified modeling language (UML) technique 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 object diagrams. Objects are briefly introduced, and their relations and hierarchies are defined in the object diagram.

For the object 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). Object behavior and notifications generated by objects are also defined in definition tables. These definitions directly relate to the service model specified in Clause 8.

6.1.4 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 objects 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 (see 7.1.2.20).

In other cases, it may be necessary to define completely new objects or to add new attributes, new methods, or new events to already defined objects. 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, objects, attributes, and methods are identified by nomenclature codes. 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 objects, attributes, etc., with unknown identifiers (i.e., nomenclature codes).

6.2 Package diagram–overview

The package diagram organizes the problem domain into separate groups. It shows the major objects inside each package and defines the relationships between these packets.

The package diagram depicted in Figure 6.3 contains only a small subset of all objects defined in the DIM. Common base objects except the Top object are not shown in this diagram. Also, not all relations are shown between the different packages. Refer to the detailed package diagrams for more information.

The numbers in the packages refer to the corresponding subclauses in this clause about models and in Clause 7 about the object definitions.

The Top object is an abstract base class and at the same time the ultimate base class for all objects defined in the model. For editorial convenience, the modeling diagrams in this standard do not show this inheritance hierarchy.

The more detailed models for these packages are contained in 6.3 through 6.10.

6.3 Model for the Medical Package

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

Figure 6.4 shows the object model of the Medical Package.

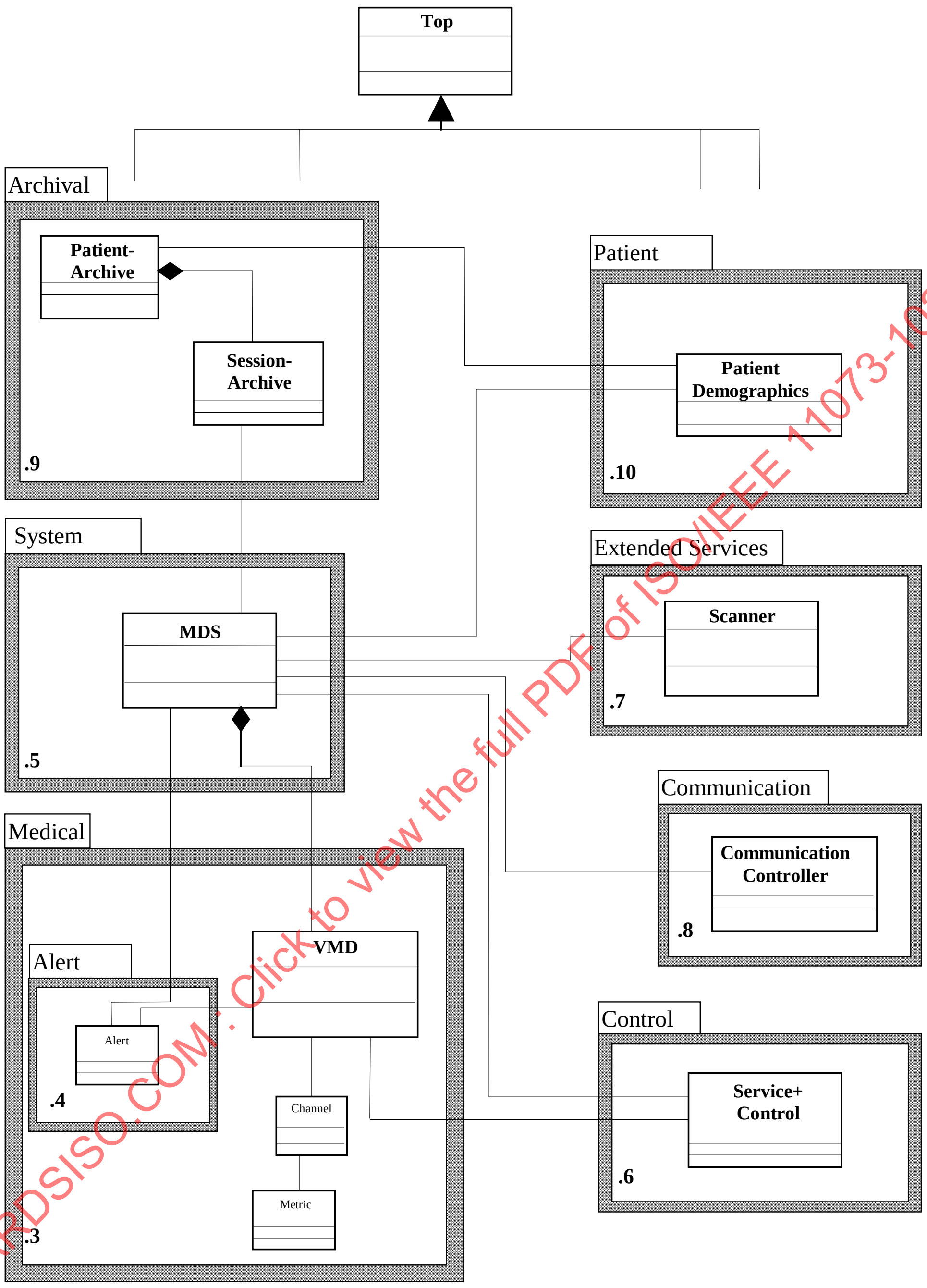

Figure 6.3—Packages of the DIM

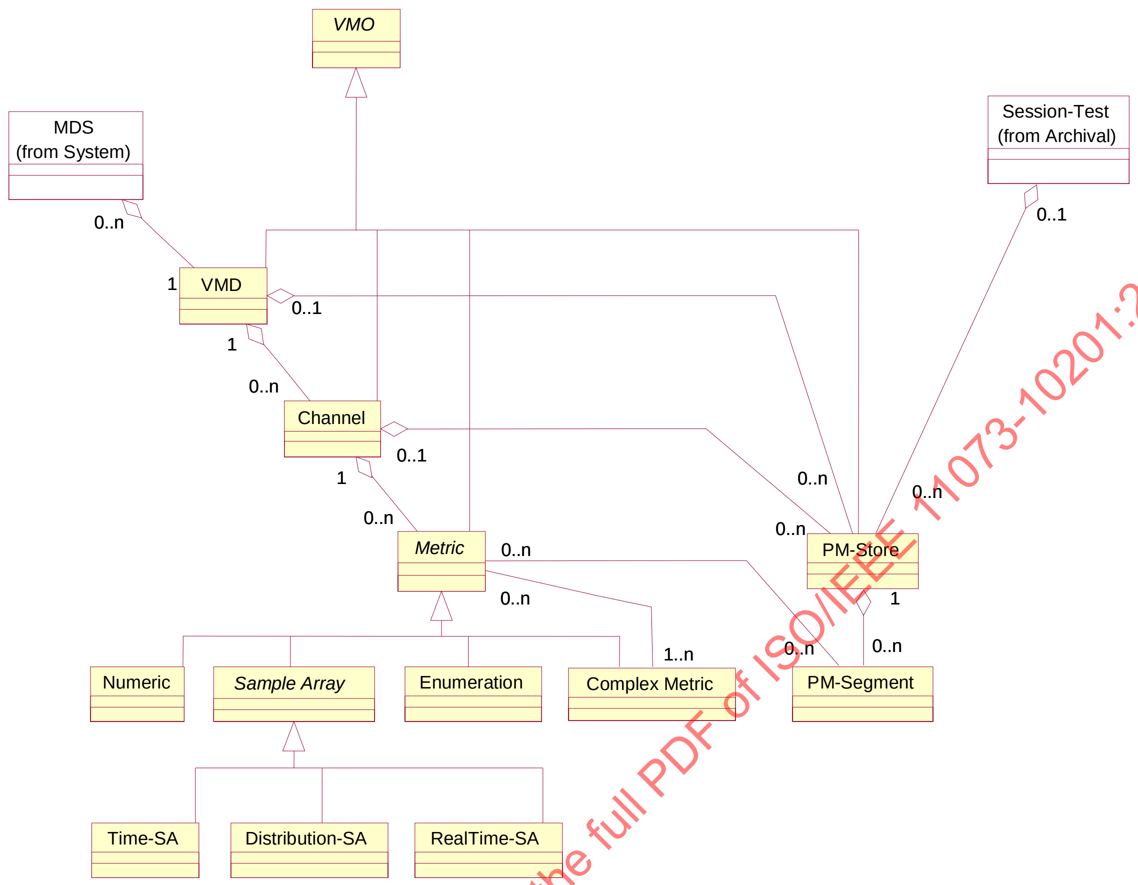

Figure 6.4—Medical Package model

NOTE—Instances of the Channel object and the PM-Store object shall be contained in exactly one superior object instance. Refer to the Alert Package for information about alarm-related objects.[8]

The Medical Package model contains the objects described in 6.3.1 through 6.3.13.

6.3.1 VMO (i.e., virtual medical object)

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

As a base class, the VMO cannot be instantiated.

6.3.2 VMD (i.e., virtual medical device) object

The VMD object is an abstraction for a medical-related subsystem (e.g., hardware or even pure software) of a medical device. Characteristics of this subsystem (e.g., modes, versions) are captured in this object. At the same time, the VMD object is a container for objects representing measurement and status information.

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

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

6.3.3 Channel object

The Channel object is used for grouping Metric objects and, thus, allows hierarchical information organization. The Channel object is not mandatory for representation of Metric objects in a VMD.

Example: 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.

6.3.4 Metric object

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

Specializations of the Metric object 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.

As a base class, the Metric object cannot be instantiated.

6.3.5 Numeric object

The Numeric object represents 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 in the Metric object.

6.3.6 Sample Array object

The Sample Array object 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.

As a base class, the Sample Array object cannot be instantiated.

6.3.7 Real Time Sample Array object

The Real Time Sample Array object is a sample array that represents a real-time continuous waveform. As such, it has special requirements in communicating systems, e.g., processing power, low latency, high bandwidth. Therefore, it requires the definition of a specialized object.

Example: An electrocardiogram (ECG) real-time wave is represented as a Real Time Sample Array object.

6.3.8 Time Sample Array object

The Time Sample Array object is a sample array that represents 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 Time Sample Array object 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 s.

6.3.9 Distribution Sample Array object

The Distribution Sample Array object is a sample array that represents linear value distributions in the form of arrays containing scaled sample values. The index of a value within an observation array denotes a spatial value, not a time point.

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 the Distribution Sample Array object to represent that spectrum in the MDIB.

6.3.10 Enumeration object

The Enumeration object 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 nomenclature scheme) 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.

6.3.11 Complex Metric object

In special cases, the Complex Metric 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. The Complex Metric 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 Complex Metric object instance, which is updated upon each breath.

6.3.12 PM-Store (i.e., persistent metric) object

The PM-Store object provides long-term storage capabilities for metric data. It contains a variable number of PM-Segment objects that can be accessed only through the PM-Store object. Without further specialization, the 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. 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).

6.3.13 PM-Segment object

The 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).

The PM-Segment object is accessible only through the PM-Store object (e.g., for retrieving stored data, the PM-Store object has to be accessed).

6.4 Model for the Alert Package

The Alert Package deals with objects 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 and, therefore, requires special handling.

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 is required to prioritize all device alarms, to latch alarms if needed (a latched alarm does not stop when the alarm condition goes away), and to produce 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 Alert Status or only Alert Monitor; 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 6.5 shows the object model of the Alert Package.

NOTE—Instances of objects in the Alert Package area shall be contained in exactly one superior object.

The Alert Package model contains the objects described in 6.4.1 through 6.4.3.

6.4.1 Alert object

The Alert object stands for the status of a simple alarm condition check. As such, it 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]. If a device instantiates an Alert object, it shall not instantiate the Alert Status or the Alert Monitor object. A single Alert object is needed for each alarm condition that the device is able to detect.

The Alert object has a reference to an object instance in the Medical Package to which the alarm condition relates.

NOTE—An Alert object instance is not dynamically created or deleted in cases where alarm conditions start or stop. Rather, an existing Alert object instance changes attribute values 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 alarm condition in the form of attribute value changes.

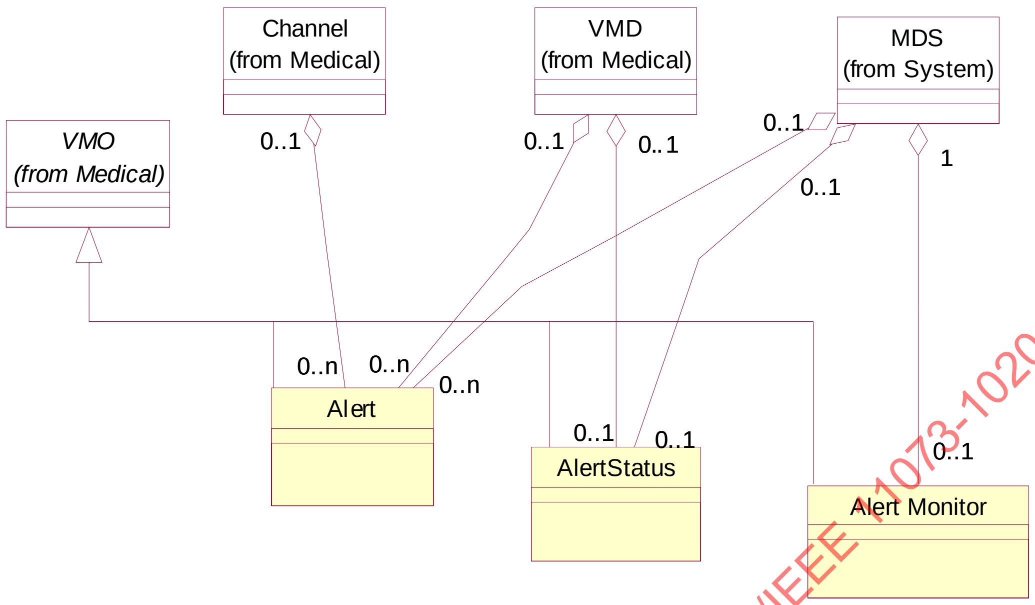

Figure 6.5—Alert Package model

6.4.2 Alert Status object

The Alert Status object represents the output of an alarm process that considers all alarm conditions in a scope that spans one or more objects. In contrast to the Alert object, the Alert Status object collects all alarm 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 alarm conditions and to suppress known false alarm indications.

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

If a device instantiates an Alert Status object, it shall not instantiate the Alert or the Alert Monitor object. Each VMD or MDS in the MDIB is able to contain at most one Alert Status object instance.

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 Alert Status object reports only a technical alarm and suppresses a heart rate limit violation alarm in this case.

6.4.3 Alert Monitor object

The Alert Monitor object represents the output of a device or system alarm processor. As such, it represents the overall device or system alarm condition and provides a list of all alarm 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 instantiates an Alert Monitor object, it shall not instantiate the Alert or the Alert Status object. An MDS shall contain not more than one Alert Monitor object instance.

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

6.5 Model for the System Package

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 6.6 shows the object model for the System Package.

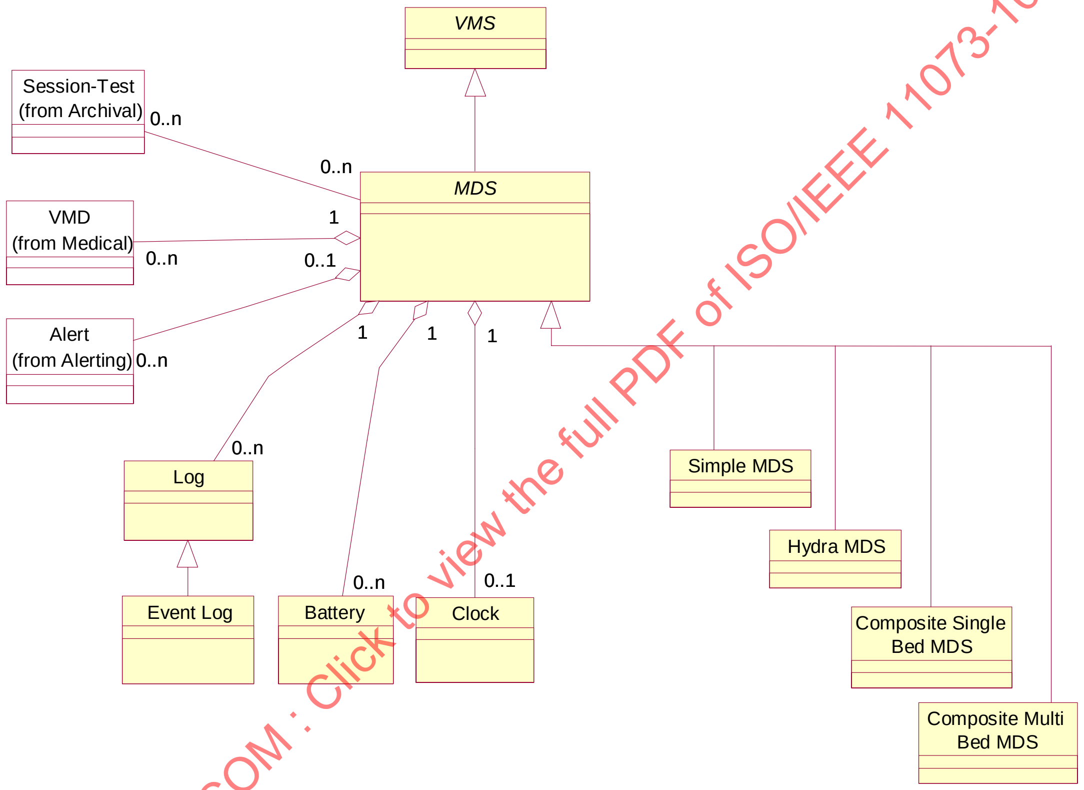

Figure 6.6—System Package model

The System Package model contains the objects described in 6.5.1 through 6.5.10.

6.5.1 VMS (i.e., virtual medical system) object

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

As a base class, the VMS object cannot be instantiated.

6.5.2 MDS object

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

The MDS object 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.

As a base class, the MDS object cannot be instantiated.

6.5.3 Simple MDS object

The Simple MDS object represents a medical device that contains a single VMD instance only (i.e., a single-purpose device).

6.5.4 Hydra MDS object

The Hydra MDS object represents a device that contains multiple VMD instances (i.e., a multipurpose device).

6.5.5 Composite Single Bed MDS object

The Composite Single Bed MDS object represents a device that contains (or interfaces with) one or more Simple or Hydra MDS objects at one location (i.e., a bed).

6.5.6 Composite Multiple Bed MDS object

The Composite Multiple Bed MDS object represents a device that contains (or interfaces with) multiple MDS objects at multiple locations (i.e., multiple beds).

6.5.7 Log object

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

As a base class, the Log object cannot be instantiated.

6.5.8 Event Log object

The Event Log object is 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.

6.5.9 Battery object

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., a smart battery) or manageable, then a special Battery object is provided.

6.5.10 Clock object

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

6.6 Model for the Control Package

The Control Package contains objects 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 an 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.

Different specializations of the Operation object define how the virtual attribute is modified. A Select Item operation, for example, allows the selection of an item from a given list of possible item values for the attribute. A Set Value operation 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 8. Instead, all controls shall be routed through the SCO (i.e., service and control object). This object supports 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 6.7 shows the object model for the Control Package:

The Control Package model contains the objects described in 6.6.1 through 6.6.9.

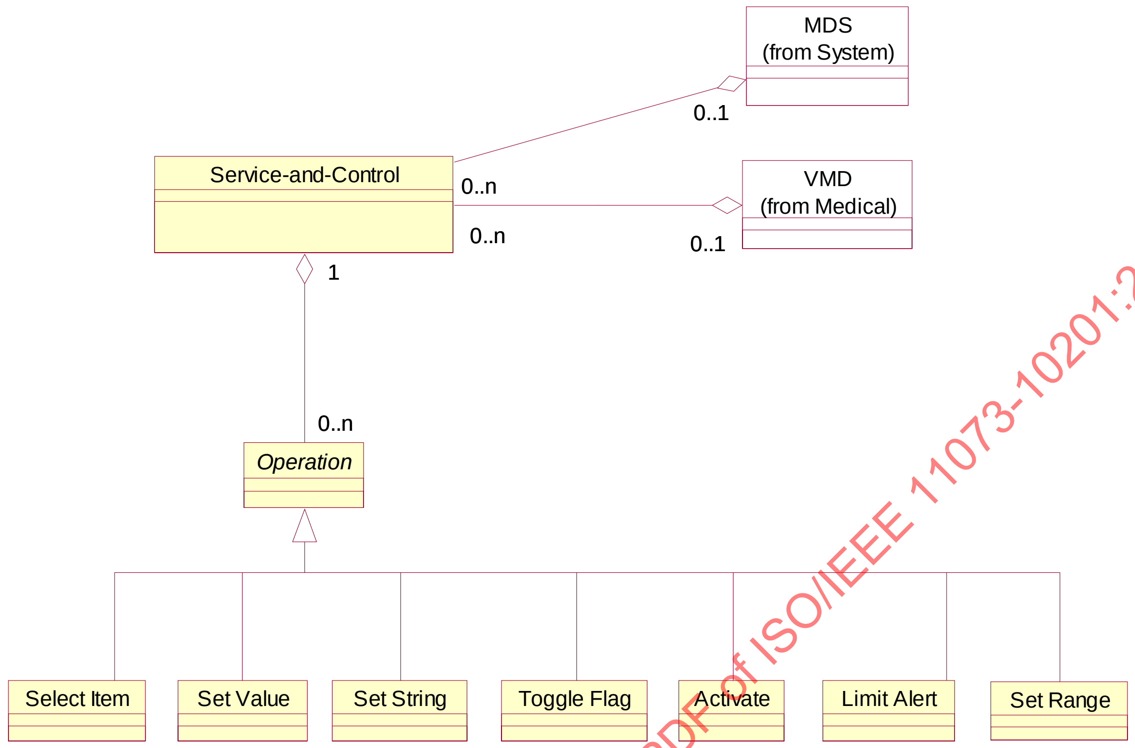

Figure 6.7—Control Package model

6.6.1 SCO

The SCO 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:

a) 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.

b) State indications, which allows local and remote indication of ongoing controls.

6.6.2 Operation object

The Operation object is the abstract base class for classes that represent 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 8. 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. This way a host system is able to discover the remote control capabilities of a device in the configuration phase.

6.6.3 Select Item Operation object

The Select Item Operation object allows the selection of one item out of a given list.

Example: The invasive pressure VMD may allow modification of its label. It uses a Select Item Operation 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.

6.6.4 Set Value Operation object

The Set Value Operation object allows 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 the Set Value Operation object for this function. The operation provides the supported value range and step width within this range.

6.6.5 Set String Operation object

The Set String Operation object 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 Set String Operation object for this function. The operation specifies the maximum string length and the character format so that the device is able to show the drug name on a small display.

6.6.6 Toggle Flag Operation object

The Toggle Flag Operation object 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 the Toggle Flag Operation object for switching the filter on or off.

6.6.7 Activate Operation object

The Activate Operation object 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 Activate Operation object.

6.6.8 Limit Alert Operation object

The Limit Alert Operation object allows adjustment of the limits of a limit alarm detector and the switching of the limit alarm to on or off.

6.6.9 Set Range Operation object

The Set Range Operation object 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 Set Range Operation object.

6.7 Model for the Extended Services Package

The Extended Services Package contains objects 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 objects 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 6.8 shows the object model for the Extended Services Package.

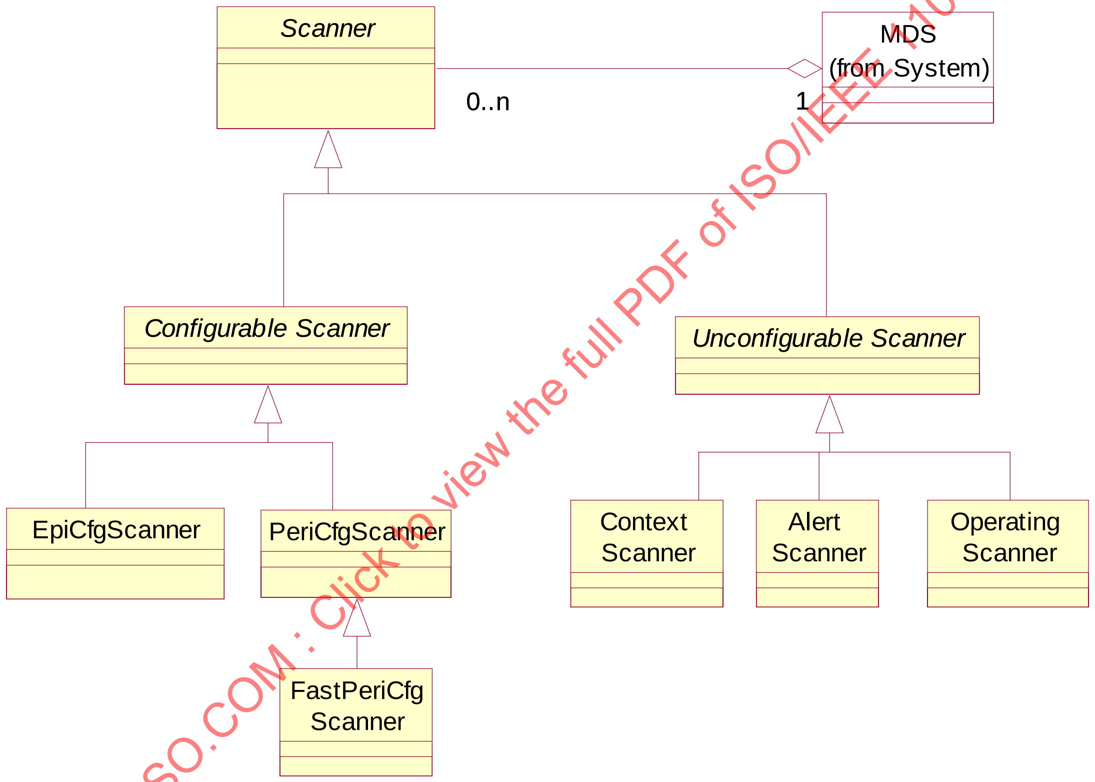

Figure 6.8—Extended Services Package model

The Extended Services Package model contains the objects described in 6.7.1 through 6.7.9.

6.7.1 Scanner object

A Scanner object is a base class that is an observer and “summarizer” of object attribute values. It 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.

Objects derived from the Scanner object may be instantiated either by the agent system itself or by the manager system (e.g., dynamic scanner creation by using the CREATE service).

As a base class, the Scanner object cannot be instantiated.

6.7.2 CfgScanner (i.e., configurable scanner) object

The CfgScanner object is a base class that has a special attribute (i.e., the ScanList attribute) that allows the system to configure which object attributes are scanned. The ScanList 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:

a) Attribute group (i.e., a defined set of attributes): The ScanList attribute contains the identifiers (IDs) of attribute groups, and all attributes in the group are scanned.

b) Individual attribute: The ScanList attribute contains the IDs of all attributes that are scanned.

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

As a base class, the CfgScanner object cannot be instantiated.

6.7.3 EpiCfgScanner (i.e., episodic configurable scanner) object

The EpiCfgScanner object 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 object attribute value changes. If the 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 ScanList attribute. The scanner instance afterwards sends a notification when the Enumeration object reports a heart beat.

6.7.4 PeriCfgScanner (i.e., periodic configurable scanner) object

The PeriCfgScanner object 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 object and configures the scanner so that it sends an update of the observed value attributes of all Numeric objects in the MDIB every 15 s.

6.7.5 FastPeriCfgScanner (i.e., fast periodic configurable scanner) object

The FastPeriCfgScanner object is a specialized object class for scanning the observed value attribute of the Real Time Sample Array object. This special scanner object 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 creates a FastPeriCfgScanner object on the agent system (e.g., server device) and requests periodic updates of all ECG leads.

6.7.6 UcfgScanner (i.e., unconfigurable scanner) object

The UcfgScanner object is a base class that 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.

As a base class, the UcfgScanner object cannot be instantiated.

6.7.7 Context Scanner object

The Context Scanner object is responsible for observing device configuration changes. After instantiation, the Context Scanner object is responsible for announcing the object instances in the device's MDIB. The scanner provides the object instance containment hierarchy and static object attribute values.

In case of dynamic configuration changes, the Context Scanner object sends notifications about new object instances or deleted object instances.

Example: A data logger creates an instance of the Context Scanner object 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).

6.7.8 Alert Scanner object

The Alert Scanner object is responsible for observing the alert-related attribute groups of objects 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).

The Alert Scanner object sends event reports periodically so that timeout conditions can be checked.

6.7.9 Operating Scanner object

The Operating Scanner object is responsible for providing all information about the operating and control system of a medical device.

In other words, the scanner 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 object attributes. Because SCOs and Operation objects may have dependencies, the scanner is not configurable.

6.8 Model for the Communication Package

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

Figure 6.9 shows the object model for the Communication Package.

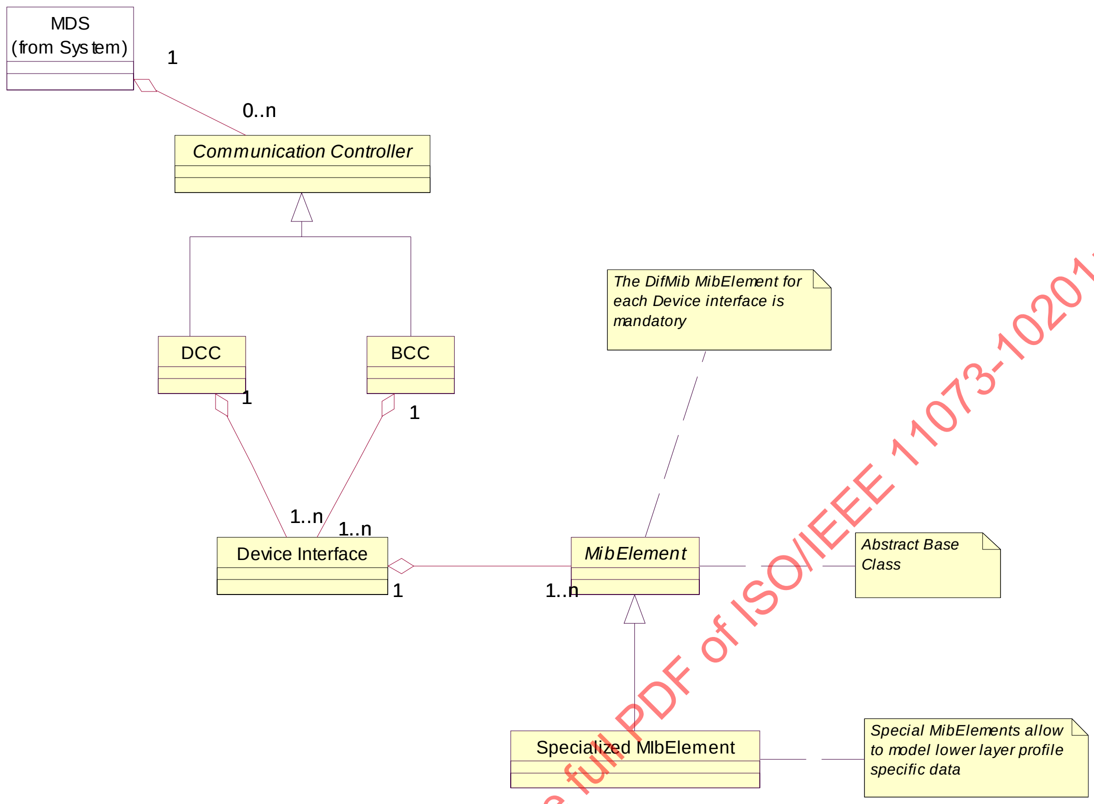

Figure 6.9—Communication Package model

The Communication Package model contains the objects described in 6.8.1 through 6.8.6.

6.8.1 Communication Controller object

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

The Communication Controller object is the access point for retrieving Device Interface attributes and management information base element (MibElement) attributes for obtaining management information related to data communications.

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

6.8.2 DCC (i.e., device communication controller) object

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

The DCC object shall contain one or more Device Interface objects.

6.8.3 BCC (i.e., bedside communication controller) object

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

The BCC object shall contain one or more Device Interface objects.

6.8.4 Device Interface object

The Device Interface object represents 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.

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

6.8.5 MibElement object

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

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

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

6.8.6 Specialized MibElement object

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 generic MibElement objects:

- The mandatory Device Interface MibElement object, which describes properties of the device interface
- An optional general communication statistics MibElement object, which models typical communication statistics that are generally applicable

Specialized MibElement objects are defined in IEEE P1073.2.1.2.

6.9 Model for the Archival Package

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

Figure 6.10 shows the object model of the Archival Package.

The Archival Package model contains the objects described in 6.9.1 through 6.9.7.

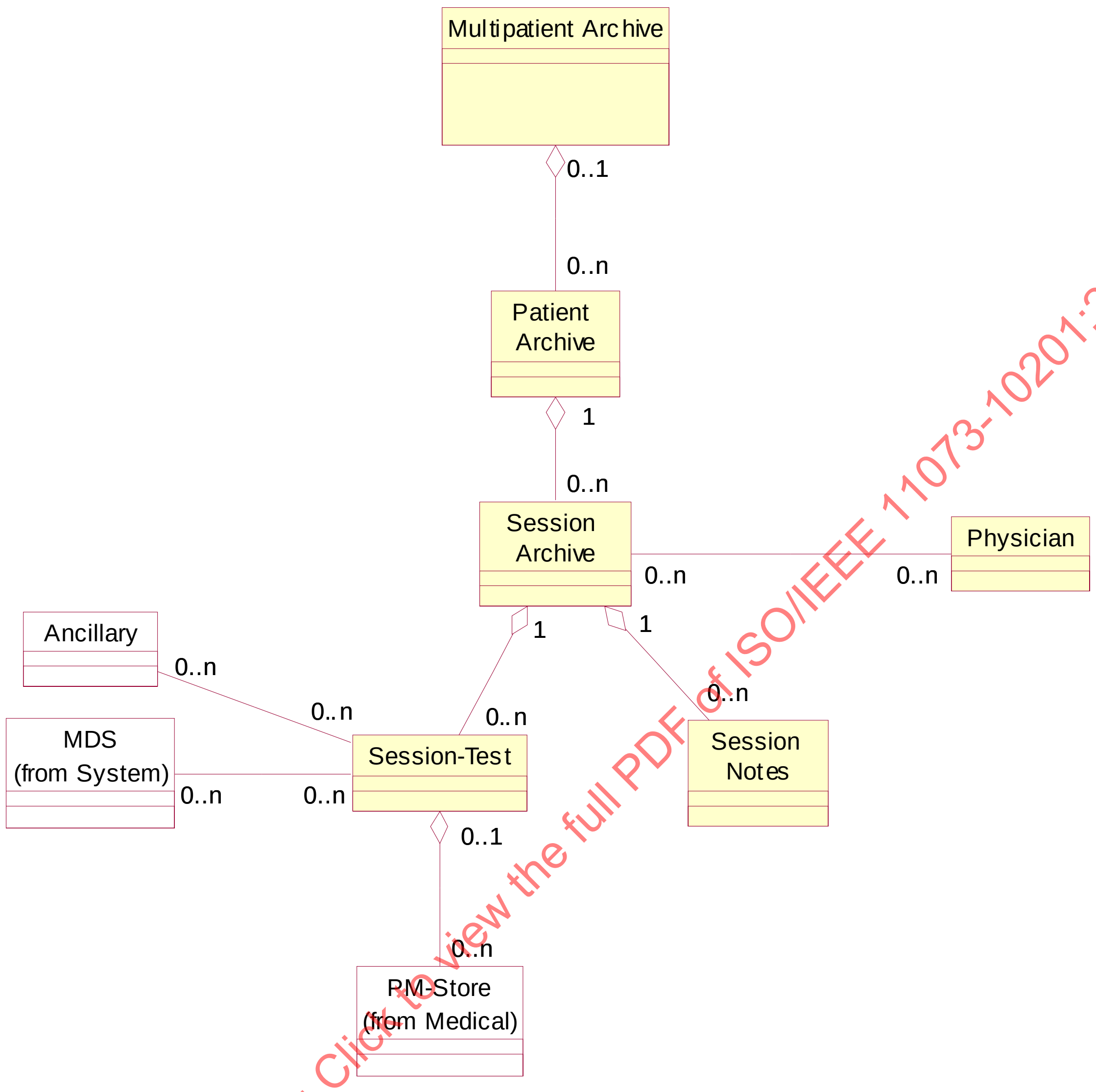

Figure 6.10—Archival Package model

6.9.1 Multipatient Archive object

The Multipatient Archive object groups together multiple Patient Archive ojbects referring to different patients.

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

6.9.2 Patient Archive object

The Patient Archive object groups 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 Patient Demographics object only.

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

6.9.3 Session Archive object

The Session Archive object represents a patient visit or a continuous stay in the a hospital or hospital department. Diagnostic treatments performed during this time period are represented by Session Test objects contained in the Session Archive object. The Session Archive object refers to dynamic (i.e., variant) data in a Patient Demographics object.

6.9.4 Physician object

The Physician object represents the physician responsible for the set of diagnostic and therapeutic activities during the time period represented by the Session Archive object.

6.9.5 Session Test object

The Session Test 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 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 Session Test object.

6.9.6 Session Notes object

The Session Notes object is a container for diagnostic data, patient care details, and treatment-related information in the form of textual data.

6.9.7 Ancillary object

The Ancillary object is not further defined in this standard. This object 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 Session Test object.

Example: Image data that complies with the MEDICOM (CEN ENV 12052) or DICOM (NEMA PS 3) standard are permitted to be included in the Session Test object as ancillary data.

6.10 Model for the Patient Package

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 6.11 shows the object model for the Patient Package:

The Patient Package model contains one object (see 6.10.1).

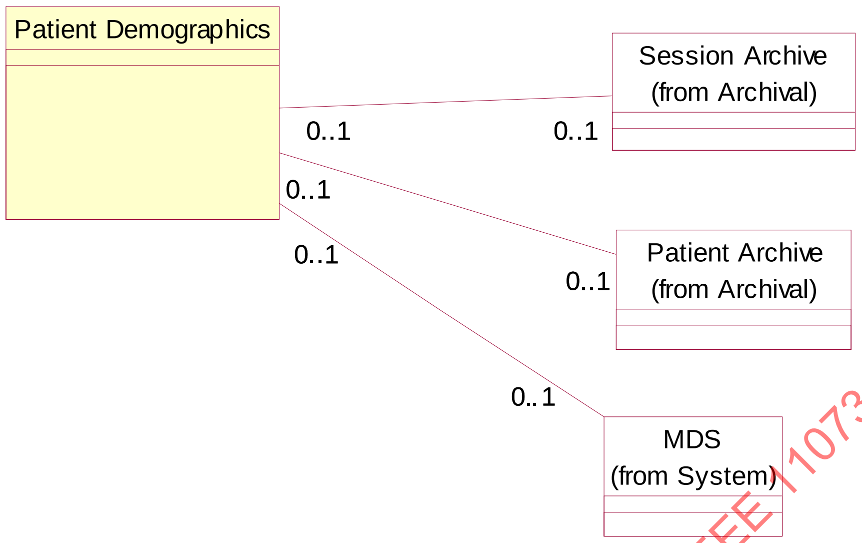

Figure 6.11—Patient Package model

6.10.1 Patient Demographics object

The Patient Demographics object stores 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.

6.11 DIM—dynamic model

6.11.1 General

Subclause 6.11 defines global dynamic system behavior.

Note that dynamic object behavior resulting from invocation of object management services is part of the object definitions in Clause 7.

6.11.2 MDS communication finite state machine (FSM)

Figure 6.12 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.

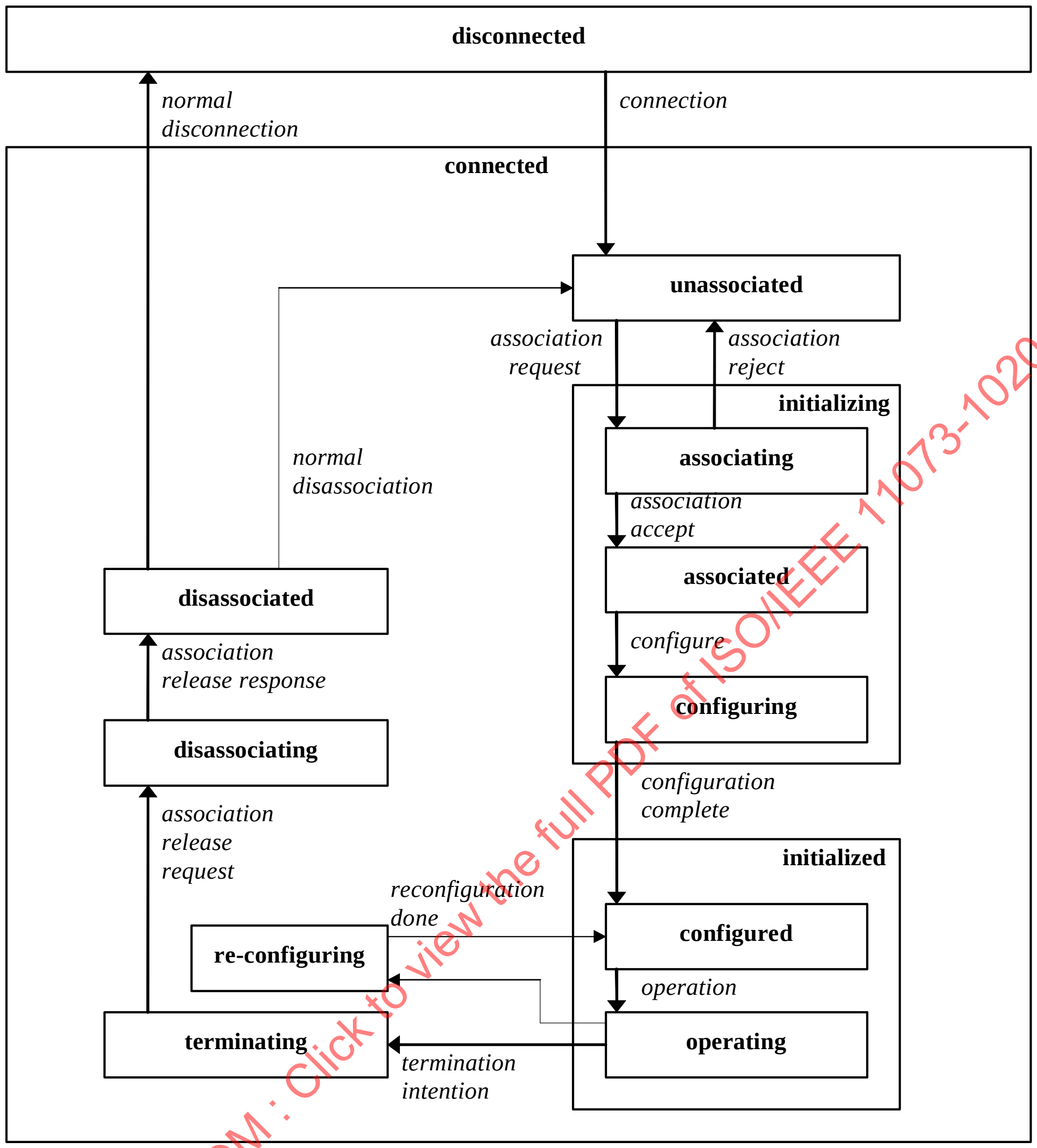

Figure 6.12—MDS FSM

After successful association, configuration data (i.e., the MDIB structure) is exchanged by the use of services and extended services (in particular, the Context Scanner 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 object. 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.

6.11.3 Communicating systems—startup object interaction diagram

Figure 6.13 presents the object interaction diagram that visualizes the startup phase after connecting two devices.

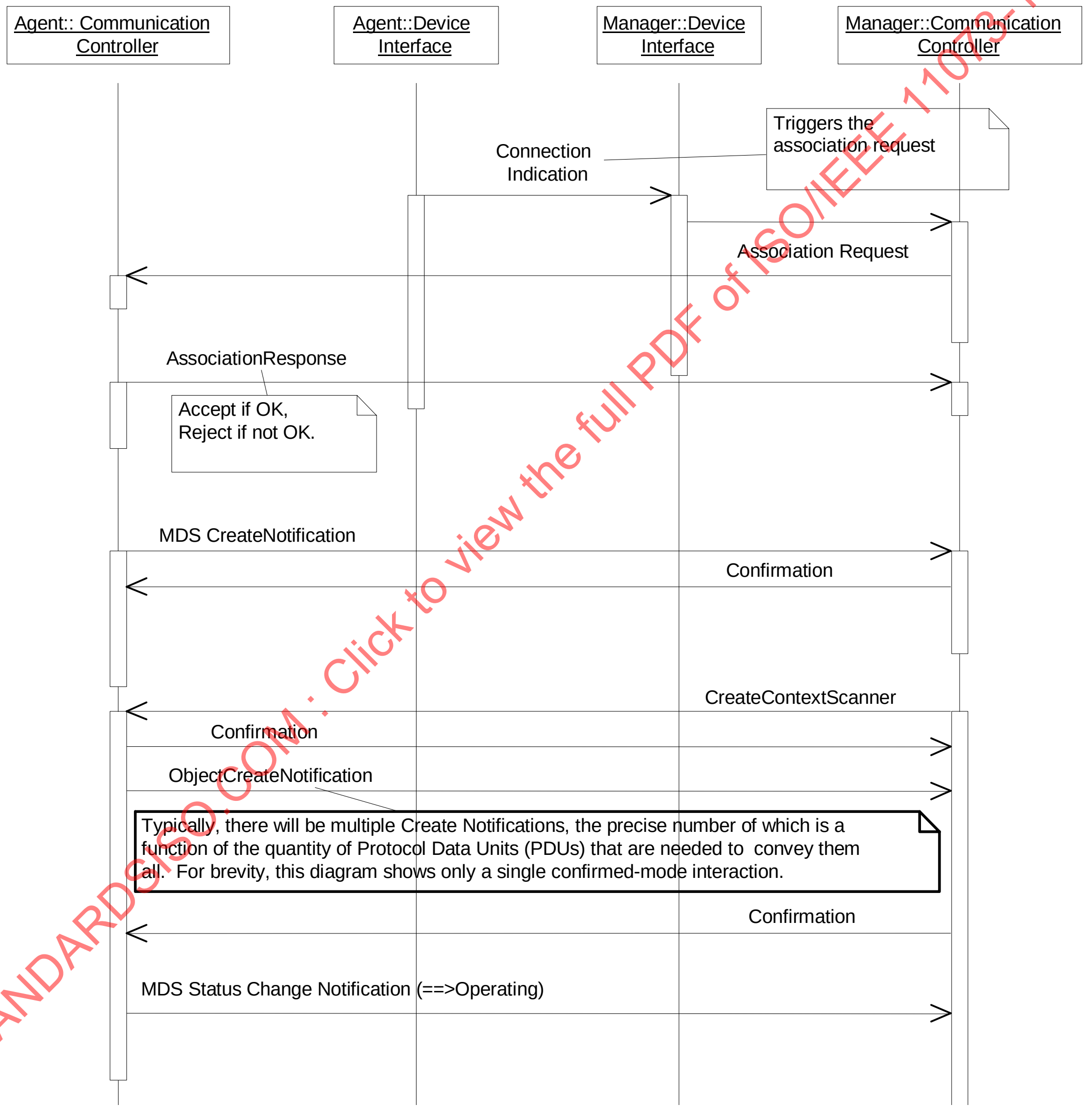
Agent:: Communication Controller
Agent::Device Interface
Manager::Device Interface
Manager::Communication Controller
Triggers the association request
Connection Indication
Association Request
AssociationResponse
Accept if OK, Reject if not OK.
MDS CreateNotification
Confirmation
CreateContextScanner
Confirmation
ObjectCreateNotification
Typically, there will be multiple Create Notifications, the precise number of which is a function of the quantity of Protocol Data Units (PDUs) that are needed to convey them all. For brevity, this diagram shows only a single confirmed-mode interaction.
Confirmation
MDS Status Change Notification (==>Operating)

Figure 6.13—Startup after connection

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

Some form of connection indication is necessary to make the manager system aware of a new agent on the network. This mechanism is dependent on the specific lower layer implementation; 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.

6.11.4 Communication Package—MibElement data access

Figure 6.14 presents the object interaction diagram that shows how a manager system accesses the MibElement data using the objects defined in the Communication Package (see 6.8).

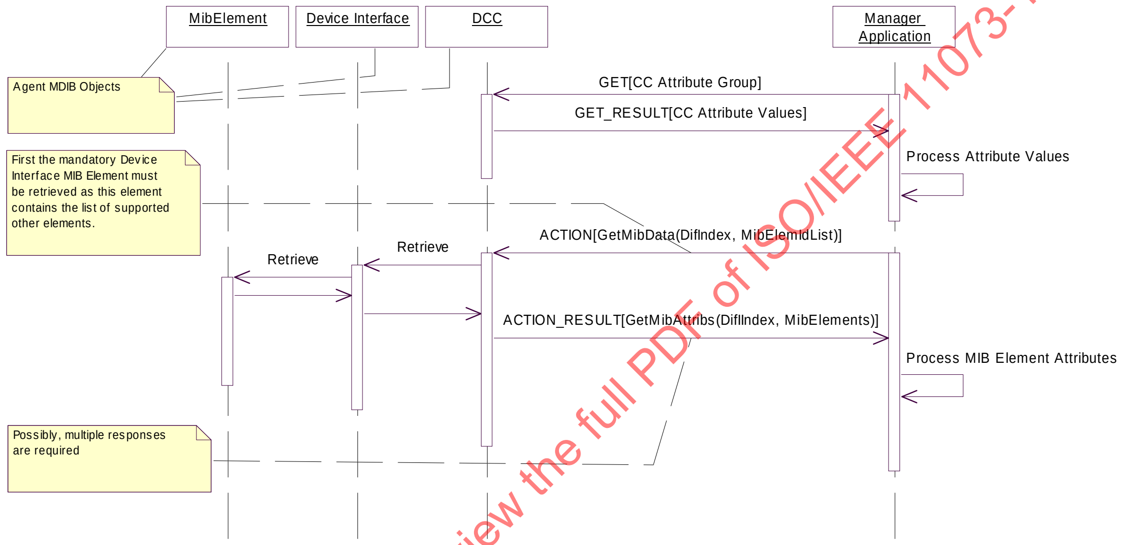
MibElement
Device Interface
DCC
Manager Application
Agent MDIB Objects
GET[CC Attribute Group]
GET_RESULT[CC Attribute Values]
Process Attribute Values
First the mandatory Device Interface MIB Element must be retrieved as this element contains the list of supported other elements.
ACTION[GetMibData(DifIndex, MibElemIdList)]
Retrieve
Retrieve
ACTION_RESULT[GetMibAttribs(DifIndex, MibElements)]
Process MIB Element Attributes
Possibly, multiple responses are required

Figure 6.14—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 object attributes and their values. The attributes specify how many Device Interface objects exist.

The manager uses the ACTION service with the Communication Controller object-defined method to retrieve the attributes of the mandatory Device Interface MibElement 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.

6.11.5 Dynamic object relations

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

Generally, the relationships between object instances 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 Session Archive object 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 Session Archive object shall not delete recorded data when the recording device is disconnected.

Thus, in certain applications (e.g., archival applications), object instance 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 6.15.

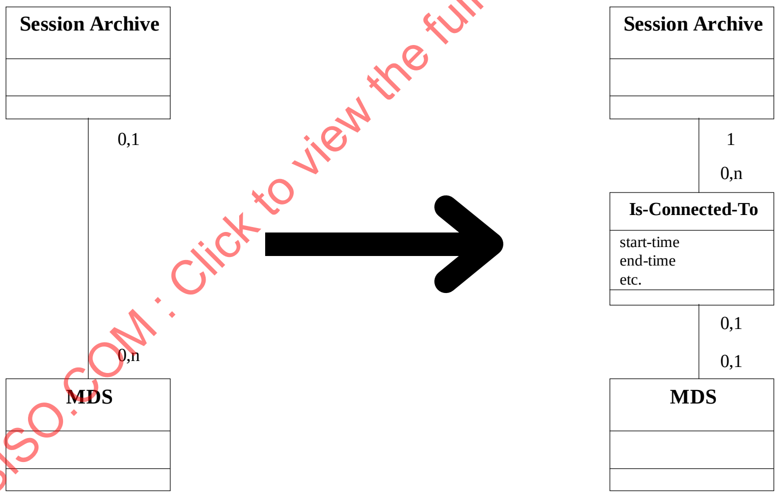

Figure 6.15—Example of a relationship represented by an object

The example in Figure 6.15 shows a relation between a Session Archive 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 6.11.5.1 and 6.11.5.2.

6.11.5.1 Dynamic object relations in communicating systems

Relations between object instances that are defined in the package models are considered configuration information. The Context Scanner object 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 object instances (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.

6.11.5.2 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 6.11.5.

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.

7. DIM object definitions

7.1 General

Clause 7 contains the object definitions for all objects in the DIM. The packages defined in the model are used to categorize objects. Attributes, behavior, and notifications are defined for each object class.

7.1.1 Notation

Each object is defined in a separate subclause (see 7.2 through 7.10). Further subclauses define attributes, behavior, and notifications for the objects.

The object is defined in a subclause as follows:

Object:	Defines the name of the object.
Description:	“Gives a short, informative textual description of the object.”
Derived From:	Defines potential base classes of the object.
Name Binding:	Defines the attribute that uniquely identifies an instance of the object 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 7.1.2.5.
Registered As:	Defines a term that is defined in the nomenclature to allow unique identification [e.g., object identifier (OID), code] of the object.

Each object attribute is 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 base classes have to be checked for a complete list of object 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 object 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 methods or functions that are provided by an object class are defined in a behavior subclause. These methods can be invoked by the CMDISE ACTION service.

Events generated by an object class (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.

7.1.2 Common data types

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

7.1.2.1 Integer and bit string data types

For representing integer numbers, the object 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)

--
-- 16-bit bit string
--
BITS-16 ::= BIT STRING (SIZE(16))
--
-- 32-bit bit string
--
BITS-32 ::= BIT STRING (SIZE(32))
```

NOTES

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).

2—In the object definitions, integer and bit string 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.

7.1.2.2 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)
--
OID-Type ::= INT-U16                          -- 16-bit integer type
```

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
```

7.1.2.3 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
```

7.1.2.4 Instance number data type

The instance number data type is used to distinguish class or 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
```

7.1.2.5 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 Context Scanner object create notifications.

Assumptions and possible restrictions about different naming contexts within an MDIB are profile dependent. They are, therefore, defined in the ISO/IEEE P11073-202xx series.

7.1.2.6 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-ext-nom(256),              -- IDs of other nomenclatures and dictionaries
        nom-part-priv(1024)                 -- private partition
    }
```

7.1.2.7 Attribute value assertion data type

A number of services defined in the service model in Clause 8 provide access to object attributes (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
    }
```

7.1.2.8 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
```

7.1.2.9 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
```

7.1.2.10 Floating point type data type

For performance and efficiency, the object 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 floating point number format is either explicitly negotiated at association or implicitly defined by the association application context.

7.1.2.11 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.

7.1.2.12 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 (SIZE(8))
```

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

7.1.2.13 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 hexadecimal 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          -- 1/10 and 1/100 of second if available
}
```

7.1.2.14 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
}
```

7.1.2.15 Operational state data type

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)
}
```

7.1.2.16 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)
}
```

7.1.2.17 Color data type

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

```
--
SimpleColour ::= INT-U16 {                  -- RGB
    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
    }
```

7.1.2.18 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 00_h

The field Locale::country shall represent the uppercase ISO/IEC 646 representation of a two-character country 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 00_h

The field Locale::charset denotes the encoding scheme of the characters used in string data types representing 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, representing 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 certain character repertoire.

7.1.2.19 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
}
```

7.1.2.20 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 objects.

7.2 Top object

Object:	Top
Description:	“The Top object is the common inheritance base for all objects in the DIM.”
Derived From:	—
Name Binding:	—
Registered As:	MDC_MOC_TOP

7.2.1 Attributes

The Top object class defines the attributes in Table 7.1.

Table 7.1—Top object 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.	n/a
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.	n/a

The Top object class does not define any attribute groups.

7.2.2 Behavior

The Top object does not define any special methods.

7.2.3 Notifications

The Top object defines the events in Table 7.2.

Table 7.2—Top object 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, the Scanner objects should be used.

7.3 Objects in the Medical Package

The definitions of objects in the Medical Package are given in 7.3.1 through 7.3.13.

7.3.1 VMO

Object:	VMO
Description:	"The VMO is the base class for all medical-related objects in the model. It provides consistent naming and identification across the Medical Package model. As a base class, the VMO cannot be instantiated."
Derived From:	Top
Name Binding:	Handle (the value of the Handle attribute is sufficient for unique identification of an instance of a VMO-derived object in a device system)
Registered As:	MDC_MOC_VMO

7.3.1.1 Attributes

The VMO class defines the attributes in Table 7.3.

Table 7.3—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	ExtObjRelation-List	Relations to objects that are not defined in the DIM.	O

The VMO class defines in Table 7.4 the attribute groups or extensions to inherited attribute groups.

Table 7.4—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

Note that the Relationship Attribute Group is not shown again in the definitions of derived classes.

7.3.1.2 Behavior

The VMO does not define any special methods.

7.3.1.3 Notifications

The VMO does not generate any special notifications.

7.3.2 VMD object

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

7.3.2.1 Attributes

The VMD object class defines the attributes in Table 7.5.

Table 7.5—VMD object 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
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

Table 7.5—VMD object class attributes *(continued)*

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
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

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

The VMD object class defines in Table 7.6 the attribute groups or extensions to inherited attribute groups.

Table 7.6—VMD object 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: Parameter-Group, Instance-Number, Compatibility-Id, Measurement-Principle, Locale
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_DYN	from VMO: Label-String from VMD: Vmd-Status
VMD Application Attribute Group	MDC_ATTR_GRP_VMD_APPL	from VMD: Position, Operating-Hours, Operation-Cycles
VMD Production Attribute Group	MDC_ATTR_GRP_VMD_PROD	from VMD: Vmd-Model, Production-Specification

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

The following 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)
}
```

7.3.2.2 Behavior

The VMD object does not define any special methods.

7.3.2.3 Notifications

The VMD object does not generate any special notifications.

7.3.3 Channel object

Object:	Channel
Description:	"The Channel object is used for grouping Metric objects and, thus, allows hierarchical information organization."
Derived From:	VMO
Name Binding:	Handle (VMO inherited)
Registered As:	MDC_MOC_VMO_CHAN

7.3.3.1 Attributes

The Channel object class defines the attributes in Table 7.7.

Table 7.7—Channel object 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 object class defines in Table 7.8 the attribute groups or extensions to inherited attribute groups.

Table 7.8—Channel object 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:
		Parameter-Group, Physical-Channel-No, Measurement-Principle

VMO Dynamic Context Group
(extensible attribute group)
	MDC_ATTR_GRP_VMO_
	DYN
		from VMO:
		Label-String
		from Channel:
		Channel-Id, Channel-Status, Color, Logical-Channel-No

The following 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)
}
```

7.3.3.2 Behavior

The Channel object does not define any special methods.

7.3.3.3 Notifications

The Channel object does not generate any special notifications.

7.3.4 Metric object

Object:	Metric
Description:	“The Metric object is the base class for all objects representing direct and derived, quantitative and qualitative biosignal measurement, status, and context data. It is a base class that is used for inheritance only.”
Derived From:	VMO
Name Binding:	Handle (VMO inherited)
Registered As:	MDC_MOC_VMO_METRIC

7.3.4.1 Attributes

The Metric object class defines the attributes in Table 7.9.

The Metric object class defines in Table 7.10 the attribute groups or extensions to inherited attribute groups.

Table 7.9—Metric object 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	Measurement-Status	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 Metric-Specification. OID is from VMO::Type or Metric-Id-Partition partition. Usually this attribute is part of an observed value, not an individual attribute.	O
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 MDC_ATTR_SITE_LIST_MSMT_EXT	SiteList SiteListExt	Measurement sites, specified in internal or external nomenclature.	O
Body-Site-List	MDC_ATTR_SITE_LIST_BODY MDC_ATTR_SITE_LIST_BODY_EXT	SiteList SiteListExt	Body sites, specified in internal or external nomenclature.	O
Metric-Calibration	MDC_ATTR_METRIC_CALIB	MetricCalibration	Indicates type and last time of calibration.	O

Table 7.9—Metric object class attributes *(continued)*

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
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-Label-String	MDC_ATTR_ID_ SUBSTANCE_LABEL _STRING	OCTET STRING	Textual attribute that identifies the substance.	O

Table 7.10—Metric object 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: Vmo-Source-List, Metric-Source-List, Unit-Code, Unit-LabelString, Msmt-Site-List, Body-Site-List, Metric-Status, Measure-Period, Averaging-Period, Start-Time, Stop-Time, Measure-Mode, Metric-Calibration, Color, Measurement-Status, Metric-Id, Metric-Id-Ext, Metric-Info-LabelString, Substance, Substance-LabelString
Metric Observed Value Group (extensible attribute group)	MDC_ATTR_GRP_ METRIC_VAL_OBS	from Metric: Metric-Id-Partition

The following 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                   INT-U8 {
          simple(0),
          compound(1),                                 -- multiple observed values, same dynamic context
          complex(2)                                   -- multiple observed values, multiple dynamic contexts
          },
      ms-comp-no                  INT-U8               -- maximum number of components in compound/complex
}

--
-- 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
--
SiteList ::= SEQUENCE OF OID-Type           -- entries are from body site nomenclature partition

--
-- 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
--
MetricSourceList ::= SEQUENCE OF OID-Type       -- OIDs from VMO::Type or Metric-Id-Partition partition

--
-- 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 alarm 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
}
```

7.3.4.2 Behavior

The Metric object does not define any special methods.

7.3.4.3 Notifications

The Metric object does not emit any special notifications.

7.3.5 Numeric object

Object:	Numeric
Description:	"The Numeric object represents numerical measurements and status information, e.g., amplitude measures, counters."
Derived From:	Metric
Name Binding:	Handle (VMO inherited)
Registered As:	MDC_MOC_VMO_METRIC_NU

7.3.5.1 Attributes

The Numeric object class defines the attributes in Table 7.11.

Table 7.11—Numeric object 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 object class defines in Table 7.12 the attribute groups or extensions to inherited attribute groups.

Table 7.12—Numeric object 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 Numeric:
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_ DYN	from VMO: Label-String from Metric: Vmo-Source-List, Metric-Source-List, Unit-Code, Unit-LabelString, Msmt-Site-List, Body-Site-List, Metric-Status, Measure-Period, Averaging-Period, Start-Time, Stop-Time, Measure-Mode, Metric-Calibration, Color, Measurement-Status, Metric-Id, Metric-Id-Ext, Metric-Info-LabelString, Substance, Substance-LabelString from Numeric: Nu-Measure-Range, Nu-Physiological-Range, Accuracy, Nu-Measure-Resolution, Display-Resolution
Metric Observed Value Group (extensible attribute group)	MDC_ATTR_GRP_ METRIC_VAL_OBS	from Metric: Metric-Id-Partition from Numeric: Nu-Observed-Value, Compound-Nu-Observed-Value, Absolute-Time-Stamp, Relative-Time-Stamp, HiRes-Time-Stamp

The following 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 class
    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
}
```

7.3.5.2 Behavior

The Numeric object does not define any special methods.

7.3.5.3 Notifications

The Numeric object does not generate any special notifications.

7.3.6 Sample Array object

Object:	Sample Array
Description:	"The Sample Array object 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."
Derived From:	Metric
Name Binding:	Handle (VMO inherited)
Registered As:	MDC_MOC_VMO_METRIC_SA

7.3.6.1 Attributes

The Sample Array object class defines the attributes in Table 7.13.

Table 7.13—Sample Array object 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	MDC_ATTR_SCALE_SPECN_I8 MDC_ATTR_SCALE_SPECN_I16 MDC_ATTR_SCALE_SPECN_I32	ScaleRangeSpec8 ScaleRange-Spec16 ScaleRange-Spec32	Defines mapping between samples and actual values as well as measurement range; type depends on sample size.	M
Sa-Physiological-Range	MDC_ATTR_SA_RANGE_PHYS_I8 MDC_ATTR_SA_RANGE_PHYS_I16 MDC_ATTR_SA_RANGE_PHYS_I32	ScaledRange8 ScaledRange16 ScaledRange32	For optimum display scaling, the physiologically meaningful range is specified.	O
Visual-Grid	MDC_ATTR_GRID_VIS_I8 MDC_ATTR_GRID_VIS_I16 MDC_ATTR_GRID_VIS_I32	SaVisualGrid8 SaVisualGrid16 SaVisualGrid32	Defines gridline positions on displays and recorders; type depends on sample size.	O

Table 7.13—Sample Array object class attributes *(continued)*

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Sa-Calibration-Data	MDC_ATTR_SA_ CALIB_I8 MDC_ATTR_SA_ CALIB_I16 MDC_ATTR_SA_ CALIB_I32	SaCalData8 SaCalData16 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 "Linear-Phase."	O
Sa-Signal-Frequency	MDC_ATTR_SA_ FREQ_SIG	SaSignal-Frequency	Maximum signal frequency.	O
Sa-Measure-Resolution	MDC_ATTR_SA_ MSMT_RES	FLOAT-Type		O
Sa-Marker-List	MDC_ATTR_SA_ MARKER_LIST_I8 MDC_ATTR_SA_ MARKER_LIST_I16 MDC_ATTR_SA_ MARKER_LIST_I32	MarkerListSaVal8 MarkerListSa-Val16 MarkerListSa-Val32		

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

The Sample Array object class defines in Table 7.14 the attribute groups or extensions to inherited attribute groups.

Table 7.14—Sample Array object 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 Sample Array: Sa-Specification, Compression, Sa-Marker-List
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_ VMO_DYN	from VMO: Label-String from Metric: Vmo-Source-List, Metric-Source-List, Unit-Code, Unit-LabelString, Msmt-Site-List, Body-Site-List, Metric-Status, Measure-Period, Averaging-Period, Start-Time, Stop-Time, Measure-Mode, Metric-Calibration, Color, Measurement-Status, Metric-Id, Metric-Id-Ext, Metric-Info-LabelString, Substance, Substance-LabelString from Sample Array: Scale-and-Range-Specification, Sa-Physiological-Range, Visual-Grid, Sa-Calibration-Data, Filter-Specification, Filter-Label-String , Sa-Signal-Frequency, Sa-Measure-Resolution

Table 7.14—Sample Array object class attribute groups *(continued)*

Attribute group	Attribute group ID	Group elements
Metric Observed Value Group (extensible attribute group)	MDC_ATTR_GRP_ METRIC_VAL_OBS	<u>from Metric:</u> Metric-Id-Partition <u>from Sample Array:</u> Sa-Observed-Value, Compound-Sa-Observed-Value

The following 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 object
      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 8-bit samples, 16 for 16-bit samples,
                                                      -- shall be divisible by 8
      significant-bits             INT-U8             -- defines significant bits in one sample
          { signed-samples(255)}                      -- if value is 255, the samples are signed; all bits are
                                                      -- significant; samples are interpreted in twos complement
}

--
-- 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                 INT-U16 {other(0), low-pass(1), high-pass(2), notch(3) },
      filter-frequency            FLOAT-Type,
      filter-order                INT-I16,              -- e.g., -1: 6 dB/octet
}

--
-- 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 ::= {
      low-edge-freq                 FLOAT-Type,
      high-edge-freq                FLOAT-Type     -- both in hertz
}

--
-- 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
}
```

7.3.6.2 Behavior

The Sample Array object does not define any special methods.

7.3.6.3 Notifications

The Sample Array object does not generate any special notifications.

7.3.7 Real Time Sample Array object

Object:	Real Time Sample Array
Description:	"The Real Time Sample Array object is a sample array that represents a real-time continuous waveform."
Derived From:	Sample Array
Name Binding:	Handle (VMO inherited)
Registered As:	MDC_MOC_VMO_METRIC_SA_RT

7.3.7.1 Attributes

The Real Time Sample Array object class defines the attributes in Table 7.15.

Table 7.15—Real Time Sample Array object 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

Table 7.15—Real Time Sample Array object class attributes *(continued)*

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
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 FastPeriCfg-Scanner 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	HighRes-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

NOTE—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.

See also 6.7.5 for the definition of the FastPeriCfgScanner object class.

The Real Time Sample Array object class defines in Table 7.16 the attribute groups or extensions to inherited attribute groups.

Table 7.16—Real Time Sample Array object 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 Sample Array:
		Sa-Specification, Compression, Sa-Marker-List
		from Real Time Sample Array:
		Sample-Period, Sweep-Speed, Average-Reporting-Delay

Table 7.16—Real Time Sample Array object class attribute groups *(continued)*

Attribute group	Attribute group ID	Group elements
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_DYN	from VMO: Label-String from Metric: Vmo-Source-List, Metric-Source-List, Unit-Code, Unit-LabelString, Msmt-Site-List, Body-Site-List, Metric-Status, Measure-Period, Averaging-Period, Start-Time, Stop-Time, Measure-Mode, Metric-Calibration, Color, Measurement-Status, Metric-Id, Metric-Id-Ext, Metric-Info-LabelString, Substance, Substance-LabelString from Sample Array: Scale-and-Range-Specification, Sa-Physiological-Range, Visual-Grid, Sa-Calibration-Data, Filter-Specification, Filter-Label-String, Sa-Signal-Frequency, Sa-Measure-Resolution From Real Time Sample Array: 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 Sample Array: Sa-Observed-Value, Compound-Sa-Observed-Value

No additional type definitions apply.

7.3.7.2 Behavior

The Real Time Sample Array object does not define any special methods.

7.3.7.3 Notifications

The Real Time Sample Array object does not generate any special notifications.

7.3.8 Time Sample Array object

Object:	Time Sample Array
Description:	“The Time Sample Array object is a sample array that represents noncontinuous wave-forms (i.e., a wave snippet).”
Derived From:	Sample Array
Name Binding:	Handle (VMO inherited)
Registered As:	MDC_MOC_VMO_METRIC_SA_T

7.3.8.1 Attributes

The Time Sample Array object class defines the attributes in Table 7.17.

The Time Sample Array object class defines in Table 7.18 the attribute groups or extensions to inherited attribute groups.

Table 7.17—Time Sample Array object 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	HighRes-RelativeTime	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

Table 7.18—Time Sample Array object 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 Sample Array: Sa-Specification, Compression, Sa-Marker-List
		from Time Sample Array: Sample-Period, Sweep-Speed

VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_DYN	from VMO: Label-String
		from Metric: Vmo-Source-List, Metric-Source-List, Unit-Code, Unit-LabelString, Msmt-Site-List, Body-Site-List, Metric-Status, Measure-Period, Averaging-Period, Start-Time, Stop-Time, Measure-Mode, Metric-Calibration, Color, Measurement-Status, Metric-Id, Metric-Id-Ext, Metric-Info-LabelString, Substance, Substance-LabelString
		from Sample Array: Scale-and-Range-Specification, Sa-Physiological-Range, Visual-Grid, Sa-Calibration-Data, Filter-Specification, Filter-Label-String, Sa-Signal-Frequency, Sa-Measure-Resolution

Metric Observed Value Group (extensible attribute group)	MDC_ATTR_GRP_METRIC_VAL_OBS	from Metric: Metric-Id-Partition
		from Sample Array: Sa-Observed-Value, Compound-Sa-Observed-Value
		from Time Sample Array: Absolute-Time-Stamp, Relative-Time-Stamp, HiRes-Time-Stamp, Tsa-Marker-List

The following 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
}
```

7.3.8.2 Behavior

The Time Sample Array object does not define any special methods.

7.3.8.3 Notifications

The Time Sample Array object does not generate any special notifications.

7.3.9 Distribution Sample Array object

Object:	Distribution Sample Array
Description:	“The Distribution Sample Array object is a sample array that represents linear value distributions in the form of arrays containing scaled sample values. 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 object and the *x* axis is specified by attributes defined in the Distribution Sample Array object.”
Derived From:	Sample Array
Name Binding:	Handle (VMO inherited)
Registered As:	MDC_MOC_VMO_METRIC_SA_D

7.3.9.1 Attributes

The Distribution Sample Array object class defines the attributes in Table 7.19.

Table 7.19—Distribution Sample Array object 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	HighRes-RelativeTime	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

Table 7.19—Distribution Sample Array object class attributes *(continued)*

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
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 Distribution Sample Array object samples	O

The Distribution Sample Array object class defines in Table 7.20 the attribute groups or extensions to inherited attribute groups.

Table 7.20—Distribution Sample Array object 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 Sample Array: Sa-Specification, Compression, Sa-Marker-List
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_ DYN	from VMO: Label-String from Metric: Vmo-Source-List, Metric-Source-List, Unit-Code, Unit-LabelString, Msmt-Site-List, Body-Site-List, Metric-Status, Measure-Period, Averaging-Period, Start-Time, Stop-Time, Measure-Mode, Metric-Calibration, Color, Measurement-Status, Metric-Id, Metric-Id-Ext, Metric-Info-LabelString, Substance, Substance-LabelString from Sample Array: Scale-and-Range-Specification, Sa-Physiological-Range, Visual-Grid, Sa-Calibration-Data, Filter-Specification, Filter-Label-String, Sa-Signal-Frequency, Sa-Measure-Resolution from Distribution Sample Array: 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 Sample Array: Sa-Observed-Value, Compound-Sa-Observed-Value from Distribution Sample Array: Absolute-Time-Stamp, Relative-Time-Stamp, HiRes-Time-Stamp, Dsa-Marker-List

The following 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
}
```

7.3.9.2 Behavior

The Distribution Sample Array object does not define any special methods.

7.3.9.3 Notifications

The Distribution Sample Array object does not generate any special notifications.

7.3.10 Enumeration object

Object:	Enumeration
Description:	“The Enumeration object 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) or in the form of free text.”
Derived From:	Metric
Name Binding:	Handle (VMO inherited)
Registered As:	MDC_MOC_VMO_METRIC_ENUM

7.3.10.1 Attributes

The Enumeration object class defines the attributes in Table 7.21.

Table 7.21—Enumeration object 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-Enum-Observed-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

Table 7.21—Enumeration object class attributes *(continued)*

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Relative-Time-Stamp	MDC_ATTR_TIME_STAMP_REL	RelativeTime		O
HiRes-Time-Stamp	MDC_ATTR_TIME_STAMP_REL_HI_RES	HighRes-RelativeTime	High-resolution timestamp.	O
Enum-Measure-Range	MDC_ATTR_ENUM_RANGE_MSMT	EnumMsmtRange	List of supported observed value OIDs. Optional if the OID type (EnumVal::enum-obj-id) 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	EnumMsmtRange Labels	Associates text strings with specific enumeration values.	O

The Enumeration object class defines in Table 7.22 the attribute groups or extensions to inherited attribute groups.

Table 7.22—Enumeration object 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, Enum-Measure-Range-Labels
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_DYN	from VMO: Label-String from Metric: Vmo-Source-List, Metric-Source-List, Unit-Code, Unit-LabelString, Msmt-Site-List, Body-Site-List, Metric-Status, Measure-Period, Averaging-Period, Start-Time, Stop-Time, Measure-Mode, Metric-Calibration, Color, Measurement-Status, Metric-Id, Metric-Id-Ext, Metric-Info-LabelString, Substance, Substance-LabelString from Enumeration: Enum-Measure-Range, Enum-Measure-Range-Bits

Table 7.22—Enumeration object class attribute groups *(continued)*

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 Enumeration: Enum-Observed-Value, Compound-Enum-Observed-Value, Absolute-Time-Stamp, Relative-Time-Stamp, HiRes-Time-Stamp

The following 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)
--
EnumMsmtRange ::= SEQUENCE OF
                                    OID-Type          -- from VMO::Type or Metric-Id-Partition partition

--
-- 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
}
```

7.3.10.2 Behavior

The Enumeration object does not define any special methods.

7.3.10.3 Notifications

The Enumeration object does not generate any special notifications.

7.3.11 Complex Metric object

Object:	Complex Metric
Description:	“The Complex Metric object acts as a container object for other Metric objects, enabling reporting of the collection as a single semantic entity.”
Derived From:	Metric
Name Binding:	Handle (VMO inherited)
Registered As:	MDC_MOC_VMO_METRIC_CMPLX

7.3.11.1 Attributes

The Complex Metric object class defines the attributes in Table 7.23.

Table 7.23—Complex Metric object 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 Complex Metric object.	O
Cmplx-Static-Attr	MDC_ATTR_CMPLX_STATIC_ATTR	CmplxStaticAttr	Static attributes of the individual objects within the Complex Metric object.	O
Cmplx-Recursion-Depth	MDC_ATTR_CMPLX_RECURSION_DEPTH	INT-U16	Mandatory if the Complex Metric object contains further Complex Metric objects (e.g., recursion). If so, the attribute defines the maximum recursion depth.	C

The Complex Metric object class shall set the Metric::MetricSpec::structure::ms-struct::complex flag.

The Complex Metric object class defines in Table 7.24 the attribute groups or extensions to inherited attribute groups.

Table 7.24—Complex Metric object 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 Complex Metric: 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: Vmo-Source-List, Metric-Source-List, Unit-Code, Unit-LabelString, Msmt-Site-List, Body-Site-List, Metric-Status, Measure-Period, Averaging-Period, Start-Time, Stop-Time, Measure-Mode, Metric-Calibration, Color, Measurement-Status, Metric-Id, Metric-Id-Ext, Metric-Info-LabelString, Substance, Substance-LabelString
		from Complex Metric: Cmplx-Dyn-Attr

Table 7.24—Complex Metric object class attribute groups *(continued)*

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 Complex Metric: Cmplx-Observed-Value

The following 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-U8 {
      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 defintion, 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 object attributes)
    }
```

7.3.11.2 Behavior

The Complex Metric object does not define any special methods.

7.3.11.3 Notifications

The Complex Metric object does not generate any special notifications.

7.3.12 PM-Store object

Object:	PM-Store
Description:	“The PM-Store object provides long-term storage capabilities for metric data. It contains a variable number of PM-Segment objects that can be accessed only through the PM-Store object.”
Derived From:	VMO
Name Binding:	Handle
Registered As:	MDC_MOC_VMO_PMSTORE

7.3.12.1 Attributes

The PM-Store object class defines the attributes in Table 7.25.

Table 7.25—PM-Store object class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Metric-Class	MDC_ATTR_METRIC _CLASS	OID-Type	Object 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_METRIC _STORE_USAGE_ CNT	INT-U32	Actual number of stored values.	O
Operational-State	MDC_ATTR_OP_ STAT	OperationalState	Indicates whether new samples are currently stored.	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 object class defines in Table 7.26 the attribute groups or extensions to inherited attribute groups.

Table 7.26—PM-Store object 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 PM-Store:
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_ DYN	from VMO: Label-String from PM-Store:
PM-Store Attribute Group	MDC_ATTR_GRP_ PMSTORE	from PM-Store: Metric-Class, Store-Sample-Algorithm, Storage-Format, Store-Capacity-Count, Store-Usage-Count, Operational-State, Sample-Period, Number-Of-Segments

The following type definitions apply:

```
--
-- The storage type 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 object;
                                              -- see 7.3.13)
      sto-t-nu-opt(2),                        -- implies optimized Numeric object format
      sto-t-rtsa-opt(3)                       -- implies optimized Real Time Sample Array 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)
}
```

7.3.12.2 Behavior

The PM-Store object defines the methods in Table 7.27.

Table 7.27—PM-Store object methods

Action	Mode	Action ID	Action parameter	Action result
Clear-Segments	Confirmed	MDC_ACT_SEG_CLEAR	SegmSelection	(empty)
Get-Segments	Confirmed	MDC_ACT_SEG_GET	SegmSelection	SegmentAttrList
Get-Segment-Info	Confirmed	MDC_ACT_SEG_GET_INFO	SegmSelection	Segment-InfoList

The following additional type definitions are needed:

```
--
-- 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
}
```

7.3.12.3 Notifications

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

7.3.13 PM-Segment object

Object:	PM-Segment
Description:	“The 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). The PM-Segment object is contained in a PM-Store object and is not directly accessible by management services.”
Derived From:	Top
Name Binding:	Instance Number (object not directly manageable; instance number unique within a single PM-Store instance)
Registered As:	MDC_MOC_PM_SEGMENT

7.3.13.1 Attributes

The PM-Segment object class defines the attributes in Table 7.28.

Table 7.28—PM-Segment object 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-Id-Partition partition).	M

Table 7.28—PM-Segment object class attributes *(continued)*

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
Vmo-Global-Reference	MDC_ATTR_VMO_ REF_GLB	GLB-HANDLE	Reference to stored Metric object.	O
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	MDC_ATTR_SEG_ DATA_GEN MDC_ATTR_SEG_ DATA_NU_OPT MDC_ATTR_SEG_ DATA_RTSA_OPT	SegDataGen SegDataNuOpt SegDataRtsaOpt	Segment data stored in a format as specified in the PM-Store object.	M
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 object class defines no attribute groups.

The following 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 Real Time Sample Array object format; a consecutive array of samples
--
SegDataRtsaOpt ::= OCTET STRING
```

7.3.13.2 Behavior

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

7.3.13.3 Notifications

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

7.4 Objects in the Alert Package

The definitions of objects in the Alert Package are given in 7.4.1 through 7.4.3.

7.4.1 Alert object

Object:	Alert
Description:	"The Alert object stands for the status of a simple alarm condition check. As such, it 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, VMD, Metric)."
Derived From:	VMO
Name Binding:	Handle
Registered As:	MDC_MOC_VMO_AL

7.4.1.1 Attributes

The Alert object class defines the attributes in Table 7.29.

Table 7.29—Alert object 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 object class defines in Table 7.30 the attribute groups or extensions to inherited attribute groups.

Table 7.30—Alert object 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

Table 7.30—Alert object class attribute groups *(continued)*

Attribute group	Attribute group ID	Group elements

VMO Dynamic Context Group
(extensible attribute group)
	MDC_ATTR_GRP_VMO_DYN	from VMO:
		Label-String
		from Alert:
		Limit-Specification

Alert Group	MDC_ATTR_GRP_AL	from Alert: Alert-Condition

The following 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 code comes from the events 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 comes the al-source (in the Alert Monitor object; see 7.4.3.1). If the last bit 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.

```
--
-- Alert controls define flags to communicate status information relevant for alarm processor; this structure is
-- reused in the Alert Status object
--
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 the
-- Alert Status object 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 7.6.8.1 for definition
    lim-al-val           CurLimAlVal    -- see 7.6.8.1 for definition
}
```

7.4.1.2 Behavior

The Alert object does not define any special methods.

7.4.1.3 Notifications

The Alert object does not generate any special notifications.

7.4.2 Alert Status object

Object:	Alert Status
Description:	“The Alert Status object represents the output of an alarm process that considers all alarm conditions in a scope that spans one or more objects. In contrast to the Alert object, the Alert Status object collects all alarm conditions related to a VMD object hierarchy or related to an MDS object and provides this information in list-structured attributes.”
Derived From:	VMO
Name Binding:	Handle
Registered As:	MDC_MOC_VMO_AL_STAT

7.4.2.1 Attributes

The Alert Status object class defines the attributes in Table 7.31.

Table 7.31—Alert Status object class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Alert-Capab-List	MDC_ATTR_AL_ STAT_AL_C_LIST	AlertCapabList	Capabilities of the Alert Status object.	M
Tech-Alert-List	MDC_ATTR_AL_ STAT_AL_T_LIST	AlertList	List of technical alert information.	O

Table 7.31—Alert Status object class attributes *(continued)*

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Physio-Alert-List	MDC_ATTR_AL_ STAT_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 Alert Status object class defines in Table 7.32 the attribute groups or extensions to inherited attribute groups.

Table 7.32—Alert Status object 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 Status: Alert-Capab-List
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_ DYN	from VMO: Label-String from Alert Status: Limit-Spec-List
Alert Status Group	MDC_ATTR_GRP_AL_ STAT	from Alert Status: Tech-Alert-List, Physio-Alert-List

The following type definitions apply:

```
--
-- The alert list is used to communicate alarm conditions derived by the Alert Status 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 code comes from the events 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 comes the al-source (in the Alert Monitor object; see 7.4.3.1). If the last bit 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.

```
--
-- Alert Status 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            BITS-16         -- defines how multiple alarms are communicated
        { dyn-inst-contents(1), rep-all-inst(2) },
    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
}

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

7.4.2.2 Behavior

The Alert Status object does not define any special methods.

7.4.2.3 Notifications

The Alert Status object does not generate any special notifications.

7.4.3 Alert Monitor object

Object:	Alert Monitor
Description:	"The Alert Monitor object represents the output of a device or system alarm processor. As such, it represents the overall device or system alarm condition and provides a list of all alarm 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."
Derived From:	VMO
Name Binding:	Handle
Registered As:	MDC_MOC_VMO_AL_MON

7.4.3.1 Attributes

The Alert Monitor object class defines the attributes in Table 7.33.

Table 7.33—Alert Monitor object class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Device-Alert-Condition	MDC_ATTR_DEV_AL_COND	DevAlert-Condition	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

Table 7.33—Alert Monitor object class attributes *(continued)*

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
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 Alert Monitor object class defines in Table 7.34 the attribute groups or extensions to inherited attribute groups.

Table 7.34—Alert Monitor object 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 Monitor:
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_DYN	from VMO: Label-String from Alert Monitor: Limit-Spec-List
Alert Monitor Group	MDC_ATTR_GRP_AL_MON	from Alert Monitor: Device-Alert-Condition, Device-P-Alarm-List, Device-T-Alarm-List, Device-Sup-Alarm-List, Suspension-Period

The following 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 code comes from the events 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 al-source comes. If the last bit 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.

7.4.3.2 Behavior

The Alert Monitor object does not define any special methods.

7.4.3.3 Notifications

The Alert Monitor object does not generate any special notifications.

7.5 Objects in the System Package

The definitions of objects in the System Package are given in 7.5.1 through 7.5.10.

7.5.1 VMS object

Object:	VMS
Description:	“The VMS object is the abstract base class for all System Package objects in this model. It provides consistent naming and identification of system-related objects.”
Derived From:	Top
Name Binding:	Handle
Registered As:	MDC_MOC_VMS

7.5.1.1 Attributes

The VMS object class defines the attributes in Table 7.35.

Table 7.35—VMS object class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Handle	MDC_ATTR_ID_ HANDLE	HANDLE	Name binding attribute.	M

Table 7.35—VMS object class attributes *(continued)*

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
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	Nomenclature-Version	Version of nomenclature 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
Ext-Obj-Relations	MDC_ATTR_EXT_OBJ_RELATION	ExtObjRelation-List	Relations to objects that are not defined in the DIM.	O

NOTE—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 object class defines in Table 7.36 the attribute groups or extensions to inherited attribute groups.

Table 7.36—VMS object 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
Relationship Attribute Group	MDC_ATTR_GRP_RELATION	from VMS: Ext-Obj-Relations

Note that the Relationship Attribute Group is not shown again in the definitions of derived classes.

The following 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                   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[9]
        },
    component-id                PrivateOid,
    prod-spec                   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
--
```

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

```
NomenclatureVersion ::= SEQUENCE {
     nom-major-version              BITS-16 {           -- major version identifier
         majorVersion1(0),
         majorVersion2(1),
         majorVersion3(2),
         majorVersion4(3)
         },
     nom-minor-version              INT-U16             -- counter to identify minor updates
}
```

7.5.1.2 Behavior

The VMS object does not define any special methods.

7.5.1.3 Notifications

The VMS object does not generate any special notifications.

7.5.2 MDS object

Object:	MDS
Description:	"The MDS object is an abstraction of a device that provides medical information in the form of objects 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."
Derived From:	VMS
Name Binding:	Handle
Registered As:	MDC_MOC_VMS_MDS

7.5.2.1 Attributes

The MDS object class defines the attributes in Table 7.37.

Table 7.37—MDS object 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[c]
Date-and-Time	MDC_ATTR_TIME_ABS	AbsoluteTime	MDS maintains device time.	O
Relative-Time	MDC_ATTR_TIME_REL	RelativeTime		O

Table 7.37—MDS object class attributes *(continued)*

Attribute name	Attribute ID[a]	Attribute type	Remark	Qualifier[b]
HiRes-Relative-Time	MDC_ATTR_TIME_REL_HI_RES	HighResRelativeTime		O
Power-Status	MDC_ATTR_POWER_STAT	PowerStatus	Either onBattery or onMains.	O[d]
Altitude	MDC_ATTR_ALTITUDE	INT-I16	Meters above or below sea level.	O
Battery-Level	MDC_ATTR_VAL_BAT_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 Patient Demographics 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 object class defines in Table 7.38 the attribute groups or extensions to inherited attribute groups.

Table 7.38—MDS object 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

Table 7.38—MDS object class attribute groups *(continued)*

Attribute group	Attribute group ID	Group elements
System Application Attribute Group (extensible attribute group)	MDC_ATTR_GRP_SYS_ APPL	from VMS: System-Capability, System-Specification from MDS: Mds-Status, Operating-Mode, Patient-Type, Date-and-Time, Power-Status, Battery-Level, Remaining-Battery-Time, Application-Area, Bed-Label, Relative-Time, HiRes-Relative-Time, Altitude, Line-Frequency
System Production Attribute Group (extensible attribute group)	MDC_ATTR_GRP_SYS_ PROD	from VMS: Production-Specification

The following 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),
      onBattery(1),
      chargingFull(8),
      chargingTrickle(9),
      chargingOff(10)
}

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

7.5.2.2 Behavior

The MDS object defines the methods in Table 7.39.

Table 7.39—MDS object methods

Action	Mode	Action ID	Action parameter	Action result
Mds-Set-Status	Confirmed	MDC_ACT_SET_MDS_ STATE	MdsSetStateInvoke	MdsSetState-Result

The following 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
```

7.5.2.3 Notifications

The MDS object defines the events in Table 7.40.

Table 7.40—MDS object 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	Mds-AttributeChange-Info	—

The following 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
```

7.5.3 Simple MDS object

Object:	Simple MDS
Description:	“The Simple MDS object represents a medical device that contains a single VMD instance only (i.e., single-purpose device).”
Derived From:	MDS
Name Binding:	Handle
Registered As:	MDC_MOC_VMS_MDS_SIMP

This MDS specialization does not define any specialized attributes, methods, and notifications.

7.5.4 Hydra MDS object

Object:	Hydra MDS
Description:	“The Hydra MDS object represents a device that contains multiple VMD instances (i.e., multipurpose device).”
Derived From:	MDS
Name Binding:	Handle
Registered As:	MDC_MOC_VMS_MDS_HYD

This MDS specialization does not define any specialized attributes, methods, and notifications.

7.5.5 Composite Single Bed MDS object

Object:	Composite Single Bed MDS
Description:	“The Composite Single Bed MDS object represents a device that contains (or interfaces with) one or more Simple or Hydra MDS objects at one location (i.e., a bed).”
Derived From:	MDS
Name Binding:	Handle
Registered As:	MDC_MOC_VMS_MDS_COMPOS_SINGLE_BED

This MDS specialization does not define any specialized attributes, methods, and notifications.

7.5.6 Composite Multiple Bed MDS object

Object:	Composite Multiple Bed MDS
Description:	“The Composite Multiple Bed MDS object represents a device that contains (or interfaces with) multiple MDS objects at multiple locations (i.e., multiple beds).”
Derived From:	MDS
Name Binding:	Handle
Registered As:	MDC_MOC_VMS_MDS_COMPOS_MULTI_BED

This MDS specialization does not define any specialized attributes, methods, and notifications.

7.5.7 Log object

Object:	Log
Description:	“The 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 object is an abstract base class and cannot be instantiated.”
Derived From:	Top

Name Binding:	Handle
Registered As:	MDC_MOC_LOG

7.5.7.1 Attributes

The Log object class defines the attributes in Table 7.41.

Table 7.41—Log object 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 object class does not define any attribute groups, and no additional type definitions are needed.

7.5.7.2 Behavior

The Log object defines the methods in Table 7.42.

Table 7.42—Log object methods

Action	Mode	Action ID	Action parameter	Action result
Clear-Log	Confirmed	MDC_ACT_CLEAR_ LOG	ClearLogRange-Invoke (optional)	ClearLog-RangeResult (optional)

The following 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              ClearLogOption,
      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
}
```

NOTE—The processing of the change counter in the clear command prevents race conditions where log entries that were not yet retrieved by the client are inadvertently cleared.

7.5.7.3 Notifications

The Log object does not generate any special notifications.

7.5.8 Event Log object

Object:	Event Log
Description:	“The Event Log object is a general Log object that stores system events in a free-text or in a binary representation.”
Derived From:	Log
Name Binding:	Handle
Registered As:	MDC_MOC_LOG_EVENT

7.5.8.1 Attributes

The Event Log object class defines the attributes in Table 7.43.

Table 7.43—Event Log object 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	EventLogEntry-List	Event entries; can be retrieved with GET service.	M
Event-Log-Info	MDC_ATTR_EVENT_LOG_INFO	EventLogInfo	Static and dynamic specifications.	O

The Event Log object class does not define any attribute groups.

The following 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
}
```

7.5.8.2 Behavior

The Event Log object defines the methods in Table 7.44.

Table 7.44—Event Log object methods

Action	Mode	Action ID	Action parameter	Action result
Get-Event-Log-Entries	Confirmed	MDC_ACT_GET_ EVENT_LOG_ ENTRIES	GetEventLogEntry-Invoke	GetEventLog-EntryResult

The following 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
}
```

7.5.8.3 Notifications

The Event Log object does not generate any special notifications.

7.5.9 Battery object

Object:	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."
Derived From:	Top
Name Binding:	Handle
Registered As:	MDC_MOC_BATT

7.5.9.1 Attributes

The Battery object class defines the attributes in Table 7.45.

Table 7.45—Battery object 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-Spec- ification	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 milliAmpere- hours).	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

Table 7.45—Battery object class attributes *(continued)*

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
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 object class defines in Table 7.46 the attribute groups or extensions to inherited attribute groups.

Table 7.46—Battery object class attribute groups

Attribute group	Attribute group ID	Group elements
Battery Attribute Group	MDC_ATTR_GRP_BATT	from Battery: (all)

The following 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
}
```

7.5.9.2 Behavior

The Battery object does not define any special methods.

7.5.9.3 Notifications

The Battery object does not generate any special notifications.

7.5.10 Clock object

Object:	Clock
Description:	“The Clock object provides additional date/time capability and status information beyond the information provided by the basic MDS object’s time-related attributes. The Clock object does not imply any specific hardware or software support.”
Derived From:	Top
Name Binding:	Handle
Registered As:	MDC_MOC_CLOCK

7.5.10.1 Attributes

The Clock object class defines the attributes in Table 7.47.

Table 7.47—Clock object 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 ASCII-based 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	HighRes-RelativeTime	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	Absolute-RelativeTimeSync	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

Table 7.47—Clock object class attributes *(continued)*

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Daylight-Savings-Transition	MDC_ATTR_TIME_DAYLIGHT_SAVINGS_TRANS	Daylight-SavingsTransition	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
Next-Leap-Seconds	MDC_ATTR_NEXT_LEAP_SECOND	LeapSeconds-Transition	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 object class defines in Table 7.48 the attribute groups or extensions to inherited attribute groups.

Table 7.48—Clock object class attribute groups

Attribute group	Attribute group ID	Group elements
Clock Attribute Group	MDC_ATTR_GRP_CLOCK	<u>from Clock:</u> (all)

The following 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           SEQUENCE OF TimeProtocolId
                                          -- List of external time protocols supported (e.g., SNTP)
}
```

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 (Real Time Sample Array objects)
    time-capab-rtsa-time-sync-high-precision(8),  -- Real Time Sample Array 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
--
TimeProtocolId ::= OID-Type                       -- from the infrastructure nomenclature partition

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

--
-- 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
}
```

NOTES

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.

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 either using zulu (or Z) format or  the offset between local
-- and GMT/UTC time; specifying the time offset shall be used if known
-- (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
--
AbsoluteTimeISO ::= OCTET STRING            -- ASCII text string that adheres to ISO 8601 format

--
-- 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 Table 7.47)
                                                -- NOTE--May be same as previous value
}
```

7.5.10.2 Behavior

The Clock object defines the methods in Table 7.49.

Table 7.49—Clock object methods

Action	Mode	Action ID	Action parameter	Action result
Set-Time	Confirmed	MDC_ACT_SET_TIME	SetTimeInvoke	None
Set-Time-Zone	Confirmed	MDC_ACT_SET_TIME_ZONE	SetTimeZoneInvoke	None

Table 7.49—Clock object methods *(continued)*

Action	Mode	Action ID	Action parameter	Action result
Set-Leap-Seconds	Confirmed	MDC_ACT_SET_LEAP_SECONDS	SetLeapSeconds-Invoke	None
Set-Time-ISO	Confirmed	MDC_ACT_SET_TIME_ISO	AbsoluteTimeISO	None

NOTE—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 data types 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
}
```

7.5.10.3 Notifications

The Clock object defines the events in Table 7.50.

Table 7.50—Clock object events

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

The following data types 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
}
```

7.6 Objects in the Control Package

The definitions of objects in the Control Package are given in 7.6.1 through 7.6.9.

7.6.1 SCO

Object:	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."
Derived From:	VMO
Name Binding:	Handle (VMO inherited)
Registered As:	MDC_MOC_CNTRL_SCO

7.6.1.1 Attributes

The SCO class defines the attributes in Table 7.51.

Table 7.51—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	ScoActivity-Indicator	Can be set by remote system to give feedback that system is under remote control.	O
Lock-State	MDC_ATTR_STAT_LOCK	Administrative-State	If locked, no operation can be invoked.	M
Invoke-Cookie	MDC_ATTR_ID_INVOK_COOKIE	INT-U32	Transaction ID assigned by invoke command.	M

The SCO class defines in Table 7.52 the attribute groups or extensions to inherited attribute groups.

Table 7.52—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-Help-Text-String, Sco-Capability
VMO Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_VMO_ DYN	from VMO: Label-String from SCO: Activity-Indicator, Vmo-Reference
SCO Transaction Group	MDC_ATTR_GRP_SCO_ TRANSACTION	from SCO: Lock-State, Invoke-Cookie

The following 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
}
```

7.6.1.2 Behavior

In addition to the SET service, which can be used to modify the Activity-Indicator attribute, the SCO defines the methods in Table 7.53.

Table 7.53—SCO methods

Action	Mode	Action ID	Action parameter	Action result
Operation-Invoke	Confirmed	MDC_ACT_SCO_OP_ INVOKE	OperationInvoke	Operation- InvokeResult
Get-Ctxt-Help	Confirmed	MDC_ACT_GET_CTXT _HELP	CtxtHelpRequest	CtxtHelpResult

The following data types 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
     }
```

7.6.1.3 Notifications

The SCO defines the events in Table 7.54.

Table 7.54—SCO events

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

The following data types 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                     INT-U16 {
          op-err-unspec(0),
          checksum-error(1),
          sco-lock-violation(2),
          unknown-operation(3),
          invalid-value(4),
          invalid-mod-type(5)
          },
      failed-operation-list        SEQUENCE OF InstNumber
}
```

7.6.2 Operation object

Object:	Operation
Description:	“The Operation object is the abstract base class for classes that represent remote controllable items.”
Derived From:	Top
Name Binding:	Instance-Number (not directly accessible by object management services; unique within a single SCO instance)
Registered As:	MDC_MOC_CNTRL_OP

7.6.2.1 Attributes

The Operation object class defines the attributes in Table 7.55.

Table 7.55—Operation object class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Instance-Number	MDC_ATTR_ID_ INSTNO	InstNumber	Unique within SCO for operation identification.	M

Table 7.55—Operation object class attributes *(continued)*

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
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 object class defines in Table 7.56 the attribute groups or extensions to inherited attribute groups.

Table 7.56—Operation object 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-Texts
Operation Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_ DYN_CTXT	from Operation: Operational-State, Vmo-Reference

The following 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
      }
```

7.6.2.2 Behavior

The Operation object does not define any special methods.

7.6.2.3 Notifications

The Operation object does not generate any special notifications.

7.6.3 Select Item Operation object

Object:	Select Item Operation
Description:	“The Select Item Operation object allows selection of one item out of a given list. The list can have different types.”
Derived From:	Operation
Name Binding:	Instance-Number (not directly accessible by object management services)
Registered As:	MDC_MOC_CNTRL_OP_SEL_IT

7.6.3.1 Attributes

The Select Item Operation object class defines the attributes in Table 7.57.

Table 7.57—Select Item Operation object 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 Select Item Operation object class defines in Table 7.58 the attribute groups or extensions to inherited attribute groups.

Table 7.58—Select Item Operation object 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-Texts from Select Item Operation: Nom-Partition
Operation Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_DYN_CTXT	from Operation: Operational-State, Vmo-Reference from Select Item Operation: Selected-Item-Index, Select-List

The following type definitions apply:

```
--
-- Select-List attribute defines valid selections
--
SelectList ::= CHOICE {
    oid-list                [1] SEQUENCE OF OID-Type,
    value-list              [3] SEQUENCE OF FLOAT-Type,
    value-u-list            [4] SEQUENCE OF SelectUValueEntry,
    string-list             [5] SEQUENCE OF OCTET STRING
}

--
-- Value with a unit/dimension code
--
SelectUValueEntry ::= SEQUENCE {
    value                   FLOAT-Type,
    m-units                 OID-Type        -- from dimensions nomenclature partition
}
```

7.6.3.2 Behavior

The Select Item Operation object does not define any special methods.

7.6.3.3 Notifications

The Select Item Operation object does not generate any special notifications.

7.6.4 Set Value Operation object

Object:	Set Value Operation
Description:	"The Set Value Operation object allows the system to adjust a value within a given range with a given resolution."
Derived From:	Operation
Name Binding:	Instance-Number (not directly accessible by object management services)
Registered As:	MDC_MOC_CNTRL_OP_SEL_VAL

7.6.4.1 Attributes

The Set Value Operation object class defines the attributes in Table 7.59.

Table 7.59—Set Value Operation object 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 Set Value Operation object class defines in Table 7.60 the attribute groups or extensions to inherited attribute groups.

Table 7.60—Set Value Operation object 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-Texts
Operation Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_DYN_CTXT	from Operation: Operational-State, Vmo-Reference from Set Value Operation: Current-Value, Set-Value-Range, Unit-Code, Step-Width

The following 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
}
```

7.6.4.2 Behavior

The Set Value Operation object does not define any special methods.

7.6.4.3 Notifications

The Set Value Operation object does not generate any special notifications.

7.6.5 Set String Operation object

Object:	Set String Operation
Description:	“The Set String Operation object is used to set the contents of a string type virtual attribute.”
Derived From:	Operation
Name Binding:	Instance-Number (not directly accessible by object management services)
Registered As:	MDC_MOC_CNTRL_OP_SET_STRING

7.6.5.1 Attributes

The Set String Operation object class defines the attributes in Table 7.61.

Table 7.61—Set String Operation object 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 Set String Operation object class defines in Table 7.62 the attribute groups or extensions to inherited attribute groups.

Table 7.62—Set String Operation object 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-Texts
Operation Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_DYN_CTXT	from Operation: Operational-State, Vmo-Reference from Set String Operation: Current-String, Set-String-Spec

The following 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
}
```

7.6.5.2 Behavior

The Set String Operation object does not define any special methods.

7.6.5.3 Notifications

The Set String Operation object does not generate any special notifications.

7.6.6 Toggle Flag Operation object

Object:	Toggle Flag Operation
Description:	“The Toggle Flag Operation object allows a switch to be toggled (with two states, e.g., on/off).”
Derived From:	Operation
Name Binding:	Instance-Number (not directly accessible by object management services)
Registered As:	MDC_MOC_CNTRL_OP_TOG

7.6.6.1 Attributes

The Toggle Flag Operation object class defines the attributes in Table 7.63.

Table 7.63—Toggle Flag Operation object 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	ToggleLabel-Strings		M

The Toggle Flag Operation object class defines in Table 7.64 the attribute groups or extensions to inherited attribute groups.

Table 7.64—Toggle Flag Operation object 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-Texts
Operation Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_ DYN_CTXT	from Operation: Operational-State, Vmo-Reference from Toggle Flag Operation: Toggle-State, Toggle-Label-Strings

The following 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
    }
```

7.6.6.2 Behavior

The Toggle Flag Operation object does not define any special methods.

7.6.6.3 Notifications

The Toggle Flag Operation object does not generate any special notifications.

7.6.7 Activate Operation object

Object:	Activate Operation
Description:	"The Activate Operation object allows a defined activity to be started (e.g., a zero pressure)."
Derived From:	Operation
Name Binding:	Instance-Number (not directly accessible by object management services)
Registered As:	MDC_MOC_CNTRL_OP_ACTIV

7.6.7.1 Attributes

The Activate Operation object class does not define any additional attributes.

This object class defines in Table 7.65 the attribute groups or extensions to inherited attribute groups.

Table 7.65—Activate Operation object 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-Texts
Operation Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_ DYN_CTXT	from Operation: Operational-State, Vmo-Reference

No additional type definitions are needed.

7.6.7.2 Behavior

The Activate Operation object does not define any special methods.

7.6.7.3 Notifications

The Activate Operation object does not generate any special notifications.

7.6.8 Limit Alert Operation object

Object:	Limit Alert Operation
Description:	"The Limit Alert Operation object allows the limits of a limit alarm detector to be adjusted and the limit alarm to be switched on or off."
Derived From:	Operation
Name Binding:	Instance-Number (not directly accessible by object management services)
Registered As:	MDC_MOC_CNTRL_OP_LIM

7.6.8.1 Attributes

The Limit Alert Operation object class defines the attributes in Table 7.66.

Table 7.66—Limit Alert Operation object 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	AlOpTextString	Individual text for upper and lower limit.	O
Set-Value-Range	MDC_ATTR_VAL_RANGE	OpSetValueRange	Allowed range for limits.	M
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 Limit Alert Operation object class defines in Table 7.67 the attribute groups or extensions to inherited attribute groups.

Table 7.67—Limit Alert Operation object 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-Texts from Limit Alert Operation: Alert-Op-Capability, Alert-Op-Text-String
Operation Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_DYN_CTXT	from Operation: Operational-State, Vmo-Reference from Limit Alert Operation: Alert-Op-State, Current-Limits, Set-Value-Range, Unit-Code, Metric-Id

The following 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
}
```

7.6.8.2 Behavior

The Limit Alert Operation object does not define any special methods.

7.6.8.3 Notifications

The Limit Alert Operation object does not generate any special notifications.

7.6.9 Set Range Operation object

Object:	Set Range Operation
Description:	“The Set Range Operation object allows the system to adjust low and high values (i.e., a value range) within defined boundaries.”
Derived From:	Operation
Name Binding:	Instance-Number (not directly accessible by object management services)
Registered As:	MDC_MOC_CNTRL_OP_SET_RANGE

7.6.9.1 Attributes

The Set Range Operation object class defines the attributes in Table 7.68.

Table 7.68—Set Range Operation object 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

Table 7.68—Set Range Operation object class attributes *(continued)*

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
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 Set Range Operation object class defines in Table 7.69 the attribute groups or extensions to inherited attribute groups.

Table 7.69—Set Range Operation object 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-Texts from Set Range Operation: Range-Op-Text
Operation Dynamic Context Group (extensible attribute group)	MDC_ATTR_GRP_OP_ DYN_CTXT	from Operation: Operational-State, Vmo-Reference from Set Value Operation: Current-Range, Set-Value-Range, Unit-Code, Step-Width

The following 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
}
```

7.6.9.2 Behavior

The Set Range Operation object does not define any special methods.

7.6.9.3 Notifications

The Set Range Operation object does not generate any special notifications.

7.7 Objects in the Extended Services Package

The definitions of objects in the Extended Services Package are given in 7.7.1 through 7.7.9.

7.7.1 Scanner object

Object:	Scanner
Description:	“A Scanner object is an observer and ‘summarizer’ of object 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.”
Derived From:	Top
Name Binding:	Handle
Registered As:	MDC_MOC_SCAN

7.7.1.1 Attributes

The Scanner object class defines the attributes in Table 7.70.

Table 7.70—Scanner object class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Handle	MDC_ATTR_ID_ HANDLE	HANDLE	Scanners are identified by handles.	M
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 object class defines in Table 7.71 the attribute groups or extensions to inherited attribute groups.

Table 7.71—Scanner object class attribute groups

Attribute group	Attribute group ID	Group elements
Scanner Attribute Group (extensible attribute group)	MDC_ATTR_GRP_SCAN	from Scanner: (all)

The attributes require no new type definitions.

7.7.1.2 Behavior

The Scanner object does not define any special methods.

Derived scanner specializations use the following common data types:

--

```
-- 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
}
```

7.7.1.3 Notifications

Events are defined in derived scanner specializations.

However, most scanner specializations 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 is grouped together
--
ScanReportInfo ::= SEQUENCE {
    scan-report-no            INT-U16,        -- counter for detection of missing events
    glb-scan-info             SEQUENCE OF SingleCtxtScan
}

SingleCtxtScan::= SEQUENCE {
    context-id                MdsContext,
    scan-info                 SEQUENCE OF ObservationScan
}

ObservationScan ::= SEQUENCE {
    obj-handle                HANDLE,
    attributes                AttributeList
}
```

7.7.2 CfgScanner object

Object:	CfgScanner
Description:	"The CfgScanner object has a special attribute (i.e., the ScanList attribute) that is used to configure which object attributes are scanned. The 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."
Derived From:	Scanner
Name Binding:	Handle
Registered As:	MDC_MOC_SCAN_CFG

7.7.2.1 Attributes

The CfgScanner object class defines the attributes in Table 7.72.

Table 7.72—CfgScanner object 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 object class defines in Table 7.73 the attribute groups or extensions to inherited attribute groups.

Table 7.73—CfgScanner object class attribute groups

Attribute group	Attribute group ID	Group elements
Scanner Attribute Group (extensible attribute group)	MDC_ATTR_GRP_SCAN	(all)

The following type definitions apply:

```
--
-- Scan-List attribute determines which object attributes are observed
--
-- 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
}
```

7.7.2.2 Behavior

The CfgScanner object does not define any special methods.

7.7.2.3 Notifications

Events are defined in derived scanner specializations.

7.7.3 EpiCfgScanner object

Object:	EpiCfgScanner
Description:	"The 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."
Derived From:	CfgScanner
Name Binding:	Handle
Registered As:	MDC_MOC_SCAN_CFG_EPI

7.7.3.1 Attributes

The EpiCfgScanner object class does not define attributes other than the attributes inherited from the Cfg-Scanner object.

The EpiCfgScanner object class uses the Scanner Attribute Group that is inherited from the CfgScanner object.

7.7.3.2 Behavior

The EpiCfgScanner object defines the methods in Table 7.74.

Table 7.74—EpiCfgScanner object methods

Action	Mode	Action ID	Action parameter	Action result
Refresh-Episodic-Data	Confirmed	MDC_ACT_REFR_EPI_DATA	RefreshObjList	none

The Refresh-Episodic-Data method triggers a refresh of all scanned attributes.

7.7.3.3 Notifications

The EpiCfgScanner object sends the notifications in Table 7.75.

Table 7.75—EpiCfgScanner object notifications

Event	Mode	Event ID	Event parameter	Event result
Unbuf-Scan-Report	Confirmed/ Unconfirmed	MDC_NOTI_UNBUF_ SCAN_RPT	ScanReportInfo	—

NOTES

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.

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).

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.

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.

7.7.4 PeriCfgScanner object

Object:	PeriCfgScanner
Description:	"The PeriCfgScanner object is responsible for scanning attributes and attribute groups of objects and for reporting these attributes in periodic event reports."
Derived From:	CfgScanner
Name Binding:	Handle
Registered As:	MDC_MOC_SCAN_CFG_PERI

7.7.4.1 Attributes

The PeriCfgScanner object class defines the attributes in Table 7.76.

Table 7.76—PeriCfgScanner object 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 object class defines in Table 7.77 the attribute groups or extensions to inherited attribute groups.

Table 7.77—PeriCfgScanner object class attribute groups

Attribute group	Attribute group ID	Group elements
Scanner Attribute Group (extensible attribute group)	MDC_ATTR_GRP_SCAN	(all)

The attributes require the following new type definitions:

```
--
-- 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
}
```

7.7.4.2 Behavior

The PeriCfgScanner object does not define any special methods.

7.7.4.3 Notifications

The PeriCfgScanner object sends the notifications in Table 7.78.

Table 7.78—PeriCfgScanner object notifications

Event	Mode	Event ID	Event parameter	Event result
Buf-Scan-Report	Confirmed/ Unconfirmed	MDC_NOTI_BUF_ SCAN_RPT	ScanReportInfo	—

7.7.5 FastPeriCfgScanner object

Object:	FastPeriCfgScanner
Description:	"The FastPeriCfgScanner object is a specialized object class for scanning the observed value attribute of the Real Time Sample Array object. This special Scanner object is further optimized for low-latency reporting and efficient communication bandwidth utilization, which is required to access real-time waveform data."
Derived From:	PeriCfgScanner
Name Binding:	Handle
Registered As:	MDC_MOC_SCAN_CFG_PERI_FAST

7.7.5.1 Attributes

The FastPeriCfgScanner object class does not define attributes other than the attributes inherited from the PeriCfgScanner object.

The FastPeriCfgScanner object class uses the Scanner Attribute Group that is inherited from the PeriCfg-Scanner object.

7.7.5.2 Behavior

The FastPeriCfgScanner object does not define any special methods.

7.7.5.3 Notifications

The FastPeriCfgScanner object sends the notifications in Table 7.79.

Table 7.79—FastPeriCfgScanner object notifications

Event	Mode	Event ID	Event parameter	Event result
Fast-Buf-Scan-Report	Confirmed/ Unconfirmed	MDC_NOTI_FAST_ BUF_SCAN_RPT	FastScanReportInfo	—

The following type definitions apply:

```
--
-- Event report contains the observed values of scanned Real Time Sample Array objects
--
FastScanReportInfo ::= SEQUENCE {
      scan-report-no                  INT-U16,
      glb-scan-info                   SEQUENCE OF SingleCtxtFastScan
}

SingleCtxtFastScan ::= SEQUENCE {
      context-id                      MdsContext,
      scan-info                       SEQUENCE OF RtsaObservationScan
}

RtsaObservationScan ::= SEQUENCE {
      handle                          HANDLE,
      observation                     SaObsValue
}
```

The FastPeriCfgScanner object is a dedicated scanner for Real Time Sample Array 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, Real Time Sample Array objects may support the Average-Reporting-Delay and Sample-Time-Sync attributes. The support for this method is signalled by the presence of the Time-Support::time-capability-time-capab-rtsa-time-sync-high-precision 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.

7.7.6 UcfgScanner object

Object:	UcfgScanner
Description:	"The UcfgScanner object scans a predefined set of managed medical objects that cannot be modified. In other words, the UcfgScanner object typically is a reporting object that is specialized for one specific purpose. It has the following properties: a) Scanner event reports are typically used in confirmed mode because the data they contain are not stateless. b) The list of scanned objects/attributes is fixed (i.e., cannot be configured). The UcfgScanner object is an abstract class; it cannot be instantiated."

Derived From:	Scanner
Name Binding:	Handle
Registered As:	MDC_MOC_SCAN_UCFG

7.7.6.1 Attributes

The UcfgScanner object class defines the attributes in Table 7.80.

Table 7.80—UcfgScanner object 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 object class defines in Table 7.81 the attribute groups or extensions to inherited attribute groups.

Table 7.81—UcfgScanner object class attribute groups

Attribute group	Attribute group ID	Group elements
Scanner Attribute Group (extensible attribute group)	MDC_ATTR_GRP_SCAN	(all)

7.7.6.2 Behavior

The UcfgScanner object does not define any special methods.

7.7.6.3 Notifications

Events are defined in derived scanner specializations.

7.7.7 Context Scanner object

Object:	Context Scanner
Description:	“The Context Scanner object is responsible for observing device configuration changes. After instantiation, the Context Scanner object is responsible for announcing the object instances in the device’s MDIB. The scanner provides the object instance containment hierarchy and static object attribute values. In case of dynamic configuration changes, the Context Scanner object sends notifications about new object instances or deleted object instances.”
Derived From:	UcfgScanner
Name Binding:	Handle
Registered As:	MDC_MOC_SCAN_UCFG_CTXT

7.7.7.1 Attributes

The Context Scanner object class defines the attributes in Table 7.82.

Table 7.82—Context Scanner object class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Context-Mode	MDC_ATTR_SCAN_ CTXT_MODE	ContextMode	Default is dynamic.	M

The Context Scanner object class uses the Scanner Attribute Group that is defined by the Scanner object.

The attributes require the following new type definitions:

```
--
-- 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)
}
```

7.7.7.2 Behavior

The Context Scanner object defines the methods in Table 7.83.

Table 7.83—Context Scanner object 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.

7.7.7.3 Notifications

The Context Scanner object defines the events in Table 7.84.

Table 7.84—Context Scanner object 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 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                SEQUENCE OF CreateEntry
}

--
-- A single new entry for one parent object, necessary to construct hierarchy in MDIB
--
CreateEntry ::= SEQUENCE {
     superior-object                 ManagedObjectId,
     created-object                  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                     SEQUENCE OF ManagedObjectId
}
```

7.7.8 Alert Scanner object

Object:	Alert Scanner
Description:	"The Alert Scanner object is responsible for observing the alert-related attribute groups of objects 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). The Alert Scanner object sends event reports periodically so that timeout conditions can be checked."
Derived From:	UcfgScanner
Name Binding:	Handle
Registered As:	MDC_MOC_SCAN_UCFG_ALSTAT

7.7.8.1 Attributes

The Alert Scanner object class defines the attributes in Table 7.85.

Table 7.85—Alert Scanner object class attributes

Attribute name	Attribute ID	Attribute type	Remark	Qualifier
Reporting-Interval	MDC_ATTR_SCAN_REP_PD	RelativeTime	Period of reports.	M

The Alert Scanner object class uses the Scanner Attribute Group that is defined by the Scanner object.

The attributes require no new type definitions.

7.7.8.2 Behavior

The Alert Scanner object does not define any special methods.

7.7.8.3 Notifications

The Alert Scanner object defines the events in Table 7.86.

Table 7.86—Alert Scanner object events

Event	Mode	Event ID	Event parameter	Event result
Alert-Scan-Report	Confirmed/ Unconfirmed	MDC_NOTI_AL_STAT_ SCAN_RPT	ScanReportInfo	—

7.7.9 Operating Scanner object

Object: Operating Scanner

Description: "The Operating Scanner object 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 object attributes together with attributes of the SCO Transaction Group (see 7.6.1.1)
- — Provides a refresh mechanism for Operation object attributes."

Derived From: UcfgScanner

Name Binding: Handle

Registered As: MDC_MOC_SCAN_UCFG_OP

7.7.9.1 Attributes

The Operating Scanner object class does not define attributes other than the attributes inherited from the UcfgScanner object.

The Operating Scanner object class uses the Scanner Attribute Group that is defined by the Scanner object.

7.7.9.2 Behavior

The Operating Scanner object defines the methods in Table 7.87.

Table 7.87—Operating Scanner object 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 object attributes.

7.7.9.3 Notifications

The Operating Scanner object defines the events in Table 7.88.

Table 7.88—Operating Scanner object 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 type definitions apply:

```
--
-- Support data types
--
OpElemAttr ::= SEQUENCE {
     op-class-id                  OID-Type,
     op-instance-no               InstNumber,
     attributes                   AttributeList
}

OpElemAttrList ::= SEQUENCE OF OpElemAttr

OpElem ::= SEQUENCE {
     op-class-id                  OID-Type,
     op-instance-no               InstNumber
}

--
-- Create and delete operations
--
OpCreateInfo ::= SEQUENCE {
     scan-report-no               INT-U16,
     scan-info                    SEQUENCE OF OpCreateEntry
}

OpCreateEntry ::= SEQUENCE {
     sco-glb-handle               GLB-HANDLE,
     created-op-list              OpElemAttrList
}

OpDeleteInfo ::= SEQUENCE {
     scan-report-no               INT-U16,
     deleted-op-list              SEQUENCE OF OpDeleteEntry
}

OpDeleteEntry ::= SEQUENCE {
     sco-glb-handle               GLB-HANDLE,
     deleted-op-list              SEQUENCE OF OpElem
}
```

```
--
-- Report of Operation object attributes (from multiple contexts, if necessary)
--
OpAttributeInfo ::= SEQUENCE {
      scan-report-no                    INT-U16,
      glb-scan-info                     SEQUENCE OF SingleCtxtOperScan
}

SingleCtxtOperScan ::= SEQUENCE {
      context-id                        MdsContext,
      scan-info                         SEQUENCE OF OpAttributeScan
}

--
-- The scanned information contains SCO transaction attributes and Operation object attributes
--
OpAttributeScan ::= SEQUENCE {
      sco-handle                        HANDLE,
      invoke-cookie                     INT-U32,
      lock-state                        AdministrativeState,
      op-elem-updt-list                 OpElemAttrList
}
```

7.8 Objects in the Communication Package

The definitions of objects in the Communication Package are given in 7.8.1 through 7.8.7.

7.8.1 Communication Controller object

Object:	Communication Controller
Description:	"The Communication Controller object 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."
Derived From:	Top
Name Binding:	Handle
Registered As:	MDC_MOC_CC (from object-oriented nomenclature partition)

7.8.1.1 Attributes

The Communication Controller object class defines the attributes in Table 7.89.

Table 7.89—Communication Controller object 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 Communication Controller 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