

INTERNATIONAL
STANDARD

ISO/IEEE
11073-
10201

Second edition
2020-04

Health informatics — Device interoperability —

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

Informatique de santé — Interopérabilité des dispositifs —

*Partie 10201: Communication entre dispositifs médicaux sur le site
des soins — Modèle d'informations du domaine*



IEEE

Reference number
ISO/IEEE 11073-10201:2020(E)



Documents

Document Sharing Hub

© IEEE 2019

Authorized licensed use limited to: Tsinghua University. Downloaded on February 24, 2026 at 08:21:34 UTC from IEEE Xplore. Restrictions apply.



COPYRIGHT PROTECTED DOCUMENT

© IEEE 2019

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from IEEE at the address below.

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

Email: stds.ipr@ieee.org
Website: www.ieee.org

Published in Switzerland

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted (see www.iso.org/directives).

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

ISO/IEEE 11073-10201 was prepared by the IEEE 11073 Standards Committee of the IEEE Engineering in Medicine and Biology Society (as IEEE Std 11073-10201-2018) and drafted in accordance with its editorial rules. It was adopted, under the “fast-track procedure” defined in the Partner Standards Development Organization cooperation agreement between ISO and IEEE, by Technical Committee ISO/TC 215, *Health informatics*.

This second edition cancels and replaces the first edition (ISO/IEEE 11073-10201:2004), which has been technically revised.

A list of all parts in the ISO/IEEE 11073 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

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

Keywords: abstract syntax, alarm, alert, ASN.1, DIM, domain information model, IEEE 11073-10201™, information model, medical device communications, medical information bus, MIB, object-oriented, patient, POC, point-of-care, remote control

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

Copyright © 2019 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 10 June 2019. Printed in the United States of America.

IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics Engineers, Incorporated.

OMG and UML are registered trademarks of Object Management Group, Inc. in the United States and/or other countries.

PDF: ISBN 978-1-5044-5433-9 STD23484
Print: ISBN 978-1-5044-5434-6 STDPD23484

IEEE prohibits discrimination, harassment, and bullying.

For more information, visit <http://www.ieee.org/web/aboutus/whatis/policies/p9-26.html>.

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

Important Notices and Disclaimers Concerning IEEE Standards Documents

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

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

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

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

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

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

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

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

Translations

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

Official statements

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

Comments on standards

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

Comments on standards should be submitted to the following address:

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

Laws and regulations

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

Copyrights

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

Photocopies

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

Updating of IEEE Standards documents

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

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

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit IEEE Xplore at <http://ieeexplore.ieee.org/> or contact IEEE at the address listed previously. For more information about the IEEE-SA or IEEE's standards development process, visit the IEEE-SA Website at <http://standards.ieee.org>.

Errata

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

Patents

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

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

Participants

At the time this IEEE standard was completed, the Point-of-Care Devices Working Group had the following membership:

John Rhoads, *Chair*
Michael Faughn, *Subgroup Chair*

Bjoern Anderson
 Malcolm Clarke
 Todd Cooper
 Chris Courville
 Kenneth Fuchs
 John Garguilo
 Frank Golatowski

David Gregorczyk
 Kai Hassing
 John Hatcliff
 Stefan Karl
 Martin Kasparick
 Koichiro Matsumoto
 Joerg-Uwe Meyer
 Stephan Poehlsen

Tracy Rausch
 Stefan Schlichting
 Paul Schluter
 Masato Tanaka
 Eugene Vasserman
 Stan Wiley
 Jan Wittenber

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

Bjoern Andersen
 Michael Bayer
 Keith Chow
 Malcolm Clarke
 Michael Faughn
 David Fuschi
 David Gregorczyk
 Randall Groves

Kai Hassing
 Werner Hoelzl
 Noriyuki Ikeuchi
 Atsushi Ito
 Stefan Karl
 Piotr Karocki
 Martin Kasparick
 H. Moll
 John Rhoads

Stefan Schlichting
 Paul Schluter
 Walter Struppler
 Ganesh Subramanian
 Thomas Tullia
 Jan Wittenber
 Oren Yuen
 Daidi Zhong

When the IEEE-SA Standards Board approved this standard on 5 December 2018, it had the following membership:

Jean-Philippe Faure, *Chair*
Gary Hoffman, *Vice Chair*
John D. Kulick, *Past Chair*
Konstantinos Karachalios, *Secretary*

Ted Burse
 Guido R. Hiertz
 Christel Hunter
 Joseph L. Koepfinger*
 Thomas Koshy
 Hung Ling
 Dong Liu

Xiaohui Liu
 Kevin Lu
 Daleep Mohla
 Andrew Myles
 Paul Nikolich
 Ronald C. Petersen
 Annette D. Reilly

Robby Robson
 Dorothy Stanley
 Mehmet Ulema
 Phil Wennblom
 Philip Winston
 Howard Wolfman
 Jingyi Zhou

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 11073-10201-2018, Health informatics—Point-of-care medical device communication—Part 10201: Domain Information Model.

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

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

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

“Plug-and-play” means that when a device or system is connected to another device or system, detection, configuration, and the initiation of communication all occur automatically and without any other human interaction.

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

Contents

1. Scope	9
2. Normative references.....	9
3. Definitions, acronyms, and abbreviations	10
3.1 Definitions	10
3.2 Abbreviations and acronyms	13
4. General requirements.....	14
5. Domain information model (DIM)	15
5.1 General	15
5.2 Package diagram—Overview	18
5.3 Model for the Medical Package	19
5.4 Model for the Alert Package	23
5.5 Model for the System Package	25
5.6 Model for the Control Package.....	27
5.7 Model for the ExtendedServices Package.....	30
5.8 Model for the Communication Package	33
5.9 Model for the Archival Package	35
5.10 Model for the Patient Package	37
5.11 DIM—Dynamic model	37
6. DIM class definitions	42
6.1 Overview	42
6.2 Top class.....	51
6.3 Medical package.....	52
6.4 Alert package.....	89
6.5 System package	95
6.6 Control package.....	115
6.7 ExtendedServices package.....	129
6.8 Communication package	142
6.9 Archival package	149
6.10 Patient package.....	156
7. Service model for communicating systems	159
7.1 General	159
7.2 Communicating systems.....	159
7.3 General service model overview.....	160
7.4 General object management services definition	162
8. MDIB nomenclature.....	167
9. Conformance model	168
9.1 Applicability	168
9.2 Conformance specification	168
9.3 ICSs	169
Annex A (informative) Bibliography	175

Health informatics—Point-of-care medical device communication

Part 10201: Domain Information Model

1. Scope

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

2. Normative references

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

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

ISO/IEEE 11073-10101, Health informatics — Point-of-care medical device communication — Part 10101: Nomenclature.²

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

OMG[®] Unified Modeling Language[®] (OMG UML[®]) 2.5.1.³

¹ ISO/IEC documents are available from the International Organization for Standardization (<http://www.iso.org/>), the International Electrotechnical Commission (<http://www.iec.ch>), and the American National Standards Institute (<http://www.ansi.org/>).

² ISO/IEEE documents are available from the International Organization for Standardization (<http://www.iso.org/>) and the Institute of Electrical and Electronics Engineers (<http://standards.ieee.org/>).

³ The OMG UML standard can be freely downloaded at <https://www.omg.org/spec/UML/>

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary Online* should be referenced for terms not defined in this clause.⁴

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

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

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

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

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

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

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

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

attribute: The definition of a property of an object.

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

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

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

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

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

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

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

⁴ The *IEEE Standards Dictionary Online* is available at <http://dictionary.ieee.org/>. An IEEE Account is required for access to the dictionary, and one can be created at no charge on the dictionary sign-in page.

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

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

NOTE—There may be several different types of data loggers; clinical, technical, forensic, alarm condition, user logs.⁵

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

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

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

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

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

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

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

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

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

Example: A single ECG waveform snippet.

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

information service element: Instances in the medical data information base.

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

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

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

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

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

⁵ Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

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

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

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

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

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

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

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

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

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

object instance: *See:* object.

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

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

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

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

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

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

scanner: An observer and “summarizer” of attribute values.

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

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

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

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

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

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

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

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

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

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

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

3.2 Abbreviations and acronyms

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

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
POC	point of care
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
UDI	Unique Device Identification
UML	unified modeling language
UTC	coordinated universal time
VMD	virtual medical device
VMO	virtual medical object
VMS	virtual medical system

4. General requirements

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

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

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

The top-level user requirements are defined in ISO/IEEE 11073-20101,⁶ 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 MDC, including measurement information, contextual data, device control methods, and other relevant aspects.
- Provide detailed specification of the classes defined in the object-oriented model, including their attributes and methods.
- Define a service model for communicating medical devices that provides access to the object instances, their attributes, and their methods.

⁶ Information on references can be found in Clause 2.

- Use the nomenclature defined in ISO/IEEE 11073-10101 to identify all data elements in the model.
- Be usable for the definition of data communication protocols as well as for the definition of file storage formats.
- Define conformance requirements.
- Be extensible and expandable to incorporate future needs in the defined modeling framework.

5. Domain information model (DIM)

5.1 General

5.1.1 Modeling concept

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

The information model and the service model for communicating systems defined and used in this standard are conceptually based on the International Organization for Standardization (ISO)/open systems interconnection (OSI) system management model (ISO/IEC 7498-1 [B16], ISO/IEC 7498-2 [B17], ISO/IEC 7498-3 [B18], ISO/IEC 7498-4 [B19]).⁷ Classes defined in the information model are considered to model managed (here, medical) objects. For the most part, the objects are directly available to management (i.e., access) services provided by the common medical device information service element (CMDISE) as defined in this standard.

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

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

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

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

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.

⁷ The numbers in brackets correspond to those of the bibliography in Annex A.

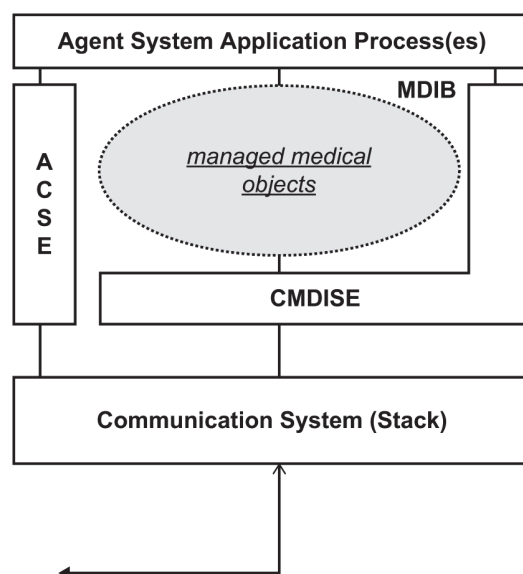


Figure 1—MDIB in communicating systems

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

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

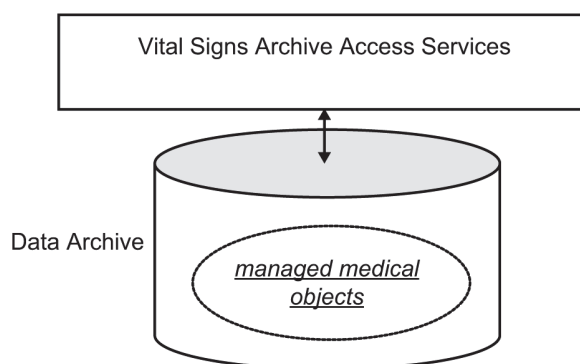


Figure 2—Managed medical objects in a vital signs archive

5.1.2 Scope of the DIM

5.1.2.1 General

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

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

5.1.2.2 Communicating systems

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

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

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

5.1.2.3 Archived vital signs

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

5.1.3 Approach

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

For the class definitions, a textual approach is followed. Attributes are defined in attribute definition tables. Attribute data types are defined using Abstract Syntax Notation One (ASN.1) as defined by ISO/IEC 8824-1 and ISO/IEC 8824-2 [B23]. The behavior of and notifications generated by objects are also defined in definition tables. These definitions directly relate to the service model specified in Clause 7.

5.1.4 Extension of the model

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

In some cases, it may be possible to use the concept of external object relations. Most classes defined in this standard provide an attribute group (e.g., the Relationship Attribute Group) that can be used to supply information about related objects that are not defined in the DIM. Supplying such information can be done by specifying a relation to an external object and assigning attributes to this relation (see 6.1.3.20).

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

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

5.2 Package diagram—Overview

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

The package diagram depicted in Figure 3 contains only a small subset of all classifiers defined in the DIM. Not all relations are shown between the shown classes. Refer to the detailed UML class diagrams for each package for more information.

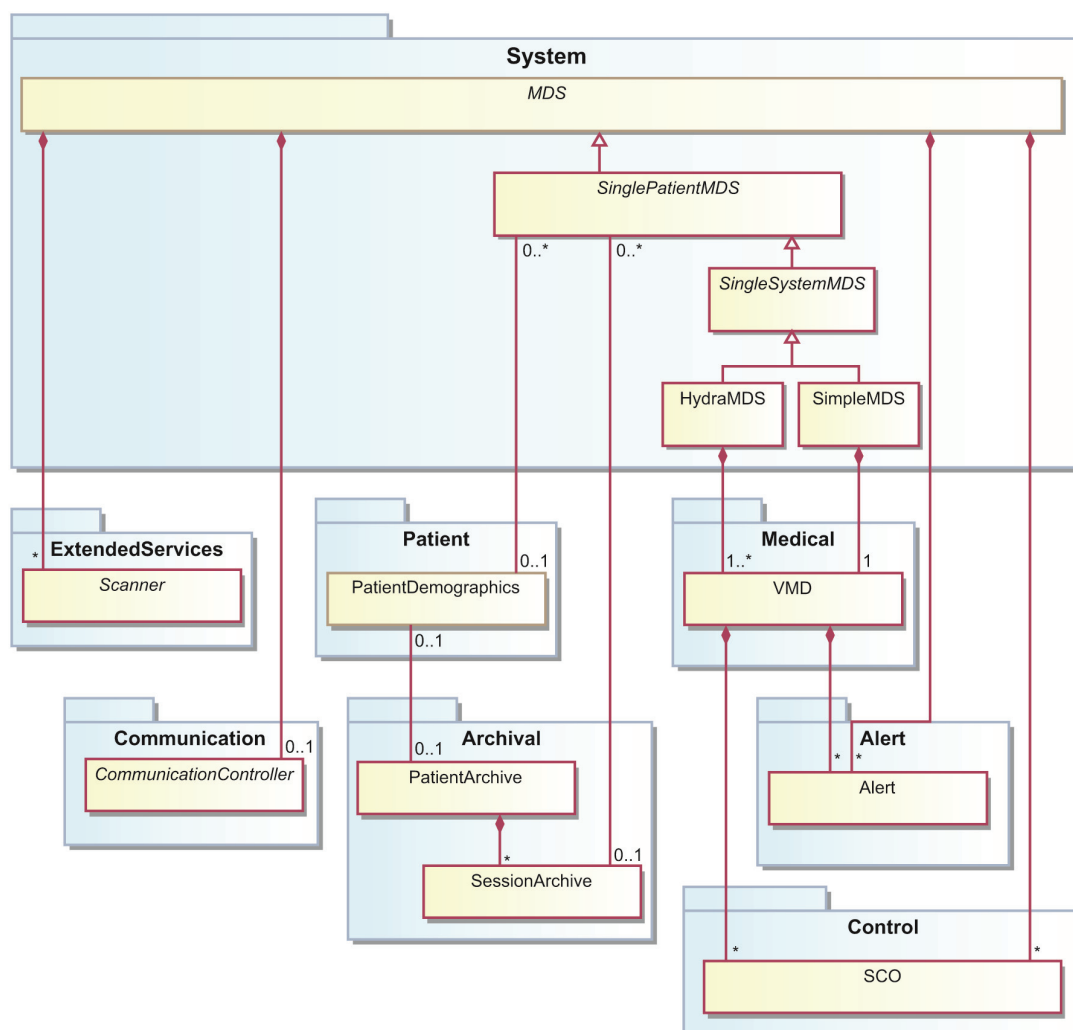


Figure 3—Packages of the DIM

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

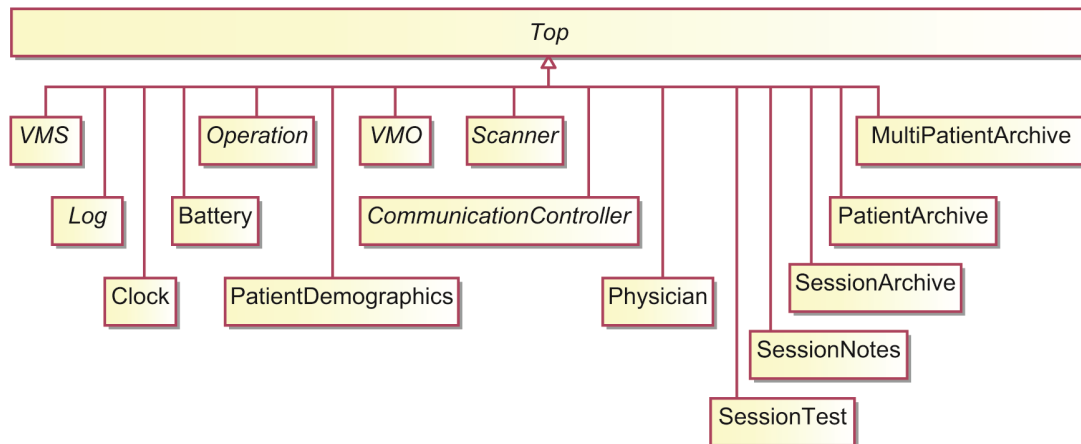


Figure 4—Top class inheritance

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

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

5.3 Model for the Medical Package

5.3.1 General

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

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

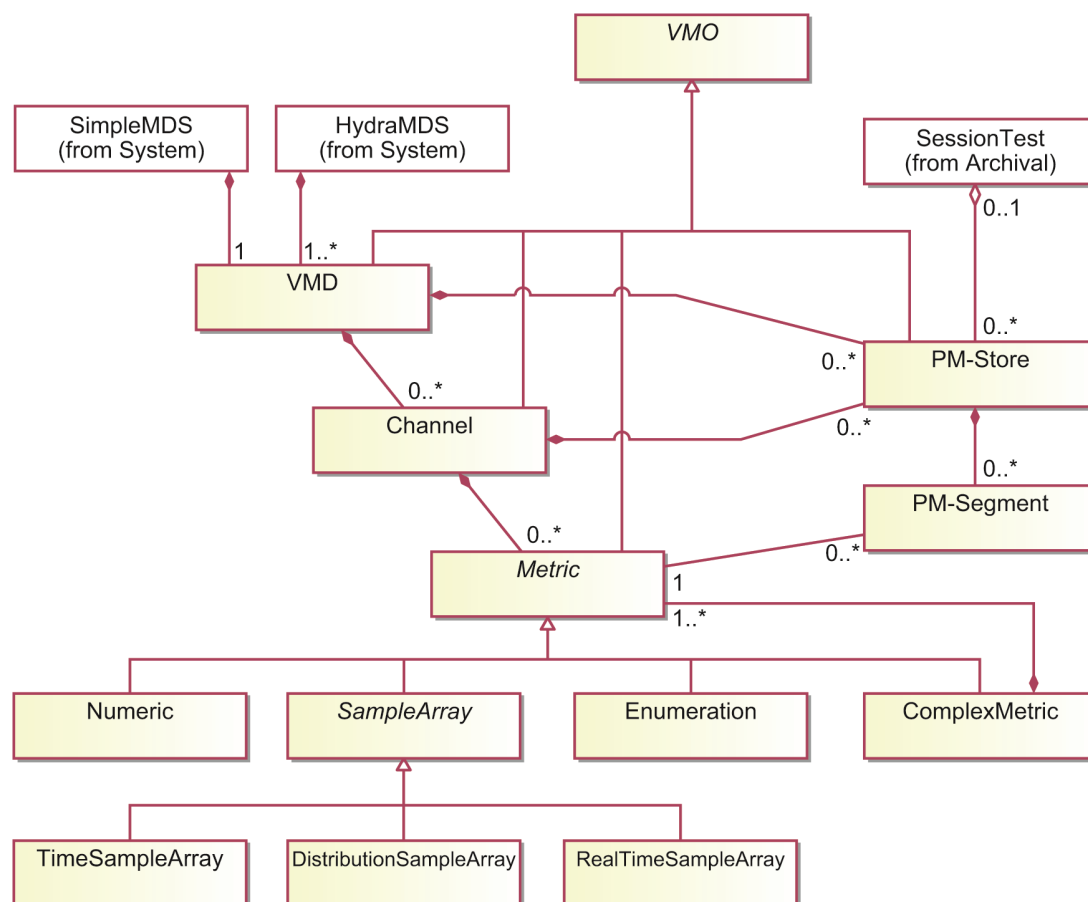


Figure 5—Medical Package model

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

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

5.3.2 VMO (virtual medical object) class

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

The VMO class is abstract and cannot be instantiated.

5.3.3 VMD (virtual medical device) class

The VMD class models medical-related subsystems (e.g., hardware or even pure software) of medical devices. The characteristics of a subsystem (e.g., modes, versions) are captured by instances of this class. At the same time, a VMD object is a container for objects representing measurement and status information.

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

5.3.4 Channel class

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

Example: 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.

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

5.3.5 Metric class

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

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

The Metric class is abstract and cannot be instantiated.

5.3.6 Numeric class

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

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

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

5.3.7 SampleArray class

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

The SampleArray class is abstract and cannot be instantiated.

5.3.8 RealTimeSampleArray class

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

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

5.3.9 TimeSampleArray class

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

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

5.3.10 DistributionSampleArray class

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

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

5.3.11 Enumeration class

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

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

5.3.12 ComplexMetric class

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

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

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

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

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

5.3.14 PM-Segment class

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

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

5.4 Model for the Alert Package

5.4.1 General

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

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

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

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

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

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

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

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

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

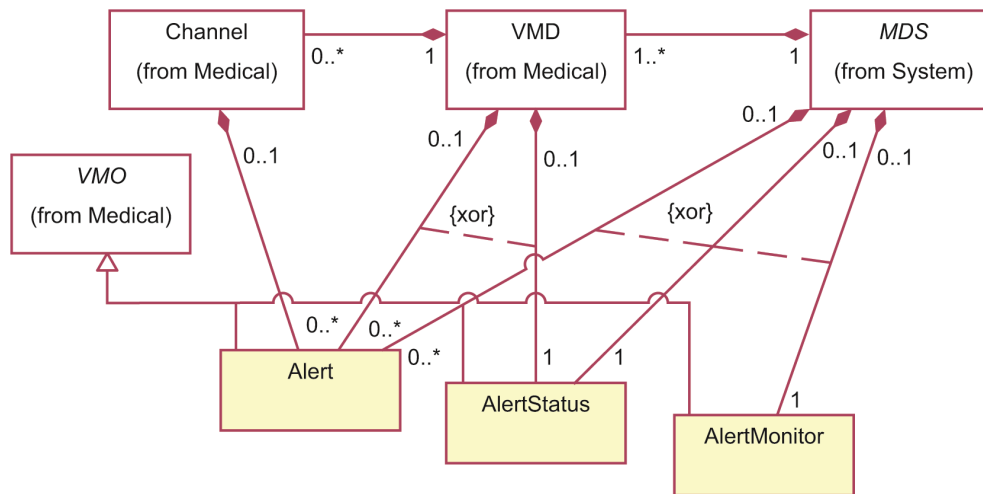


Figure 6—Alert Package model

5.4.2 Alert class

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

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

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

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

5.4.3 AlertStatus class

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

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

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

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

5.4.4 AlertMonitor class

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

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

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

5.5 Model for the System Package

5.5.1 General

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

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

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

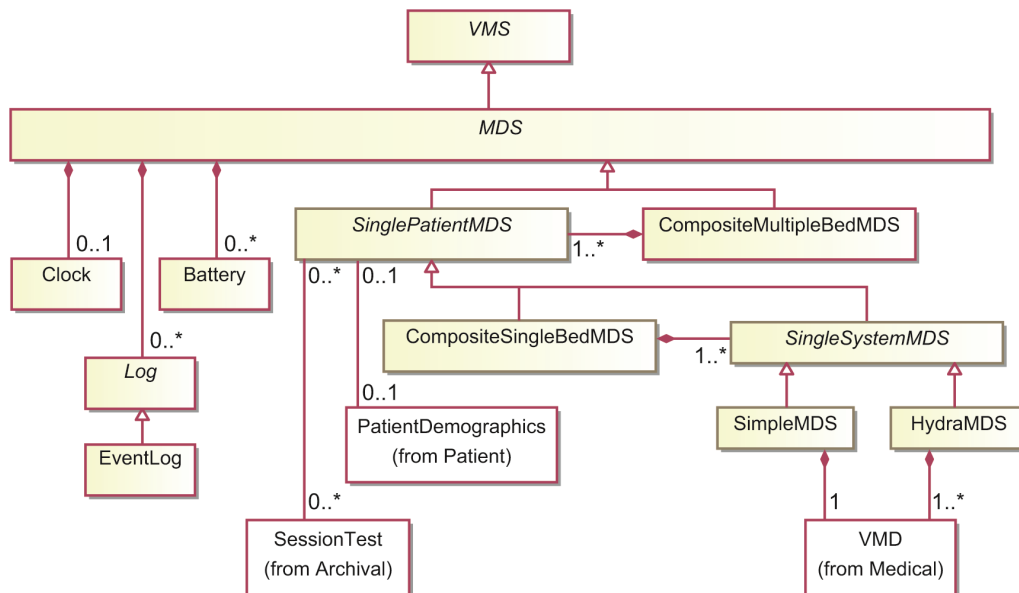


Figure 7—System Package model

5.5.2 VMS (virtual medical system) class

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

The VMS class is abstract and cannot be instantiated.

5.5.3 MDS class

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

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

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

The MDS class is abstract and cannot be instantiated.

5.5.4 CompositeMultipleBedMDS class

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

5.5.5 SinglePatientMDS class

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

The SinglePatientMDS class is abstract and cannot be instantiated.

5.5.6 CompositeSingleBedMDS class

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

5.5.7 SingleSystemMDS class

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

The SingleSystemMDS class is abstract and cannot be instantiated.

5.5.8 SimpleMDS class

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

5.5.9 HydraMDS class

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

5.5.10 Log class

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

The Log class is abstract and cannot be instantiated.

5.5.11 EventLog class

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

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

5.5.12 Battery class

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

5.5.13 Clock class

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

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

5.6 Model for the Control Package

5.6.1 General

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

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

- A system that allows remote control is able to explicitly register which attributes or features can be accessed or modified by a remote system.
- For attributes that can be remotely modified, a list of possible legal attribute values is provided to the controlling system.
- It is not mandatory that a remote-controllable item correspond to an attribute of a medical object.
- Dependence of a controllable item on internal system states is modeled.
- A simple locking transaction scheme allows the handling of transient states during remote control.

At least two different uses of remote control are considered:

- Automatic control may be done by some processes running on the controlling device. Such a process has to be able to discover automatically how it can modify or access the controllable items to provide its function.
- It is also possible to use remote control to present some form of control interface to a human operator. For this use, descriptions of functions, and possibly help information, need to be provided.

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

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

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

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

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

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

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

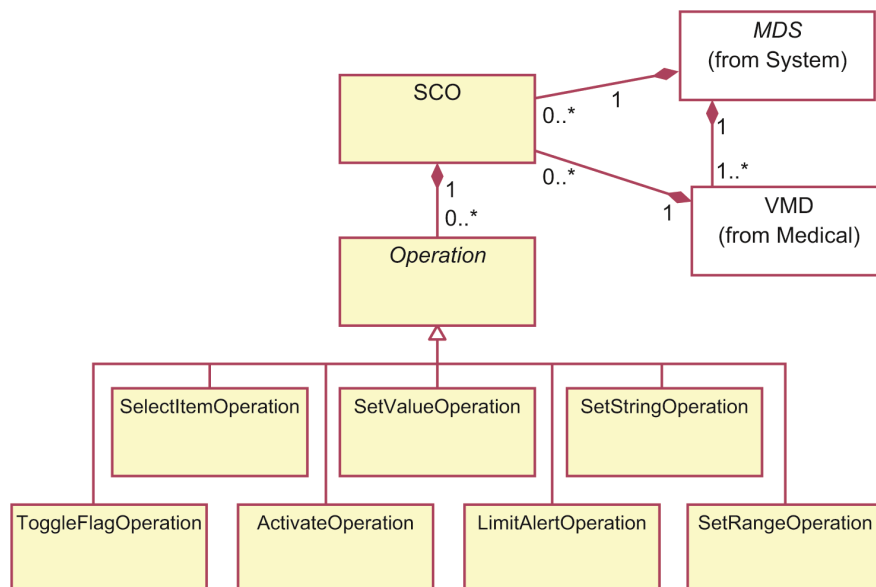


Figure 8—Control Package model

5.6.2 SCO class

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

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

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

5.6.3 Operation class

The Operation class is the abstract base class for classes that model remote-controllable items. Each Operation object allows the system to modify some specific item (i.e., a virtual attribute) in a specific way

defined by the Operation object. Operation objects are not directly accessible by services defined in the service model in Clause 7. All controls shall be routed through the SCO object (i.e., the parent) to allow a simple form of transaction processing.

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

The Operation class is abstract and cannot be instantiated.

5.6.4 SelectItemOperation class

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

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

5.6.5 SetValueOperation class

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

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

5.6.6 SetStringOperation class

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

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

5.6.7 ToggleFlagOperation class

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

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

5.6.8 ActivateOperation class

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

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

5.6.9 LimitAlertOperation class

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

5.6.10 SetRangeOperation class

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

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

5.7 Model for the ExtendedServices Package

5.7.1 General

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

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

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

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

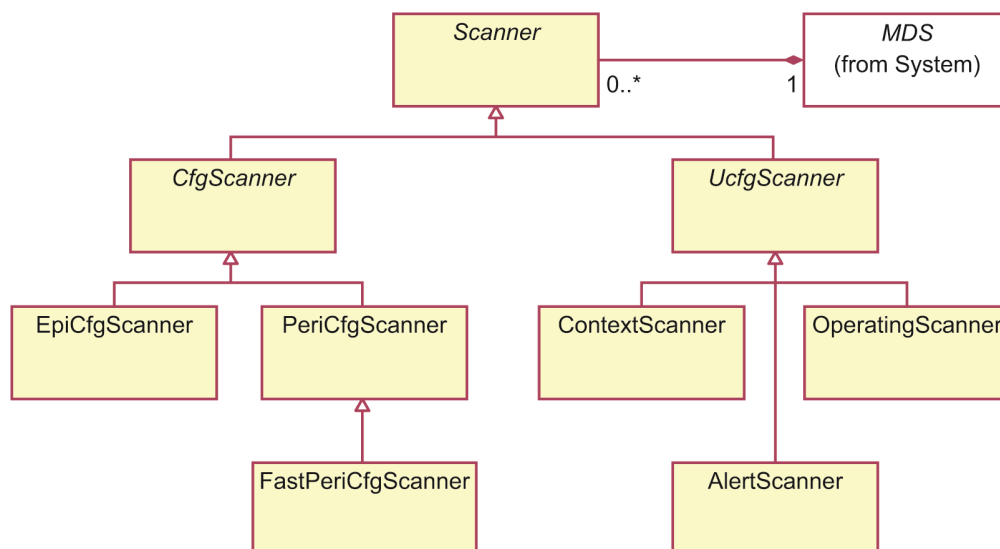


Figure 9—ExtendedServices Package model

5.7.2 Scanner class

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

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

The Scanner class is abstract and cannot be instantiated.

5.7.3 CfgScanner (configurable scanner) class

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

A CfgScanner object may support different granularity for scanning:

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

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

The CfgScanner class is abstract and cannot be instantiated.

5.7.4 EpiCfgScanner (episodic configurable scanner) class

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

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

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

5.7.5 PeriCfgScanner (periodic configurable scanner) class

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

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

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

5.7.6 FastPeriCfgScanner (fast periodic configurable scanner) class

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

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

5.7.7 UcfgScanner (unconfigurable scanner) class

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

The UcfgScanner class is abstract and cannot be instantiated.

5.7.8 ContextScanner class

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

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

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

5.7.9 AlertScanner class

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

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

5.7.10 OperatingScanner class

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

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

5.8 Model for the Communication Package

5.8.1 General

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

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

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

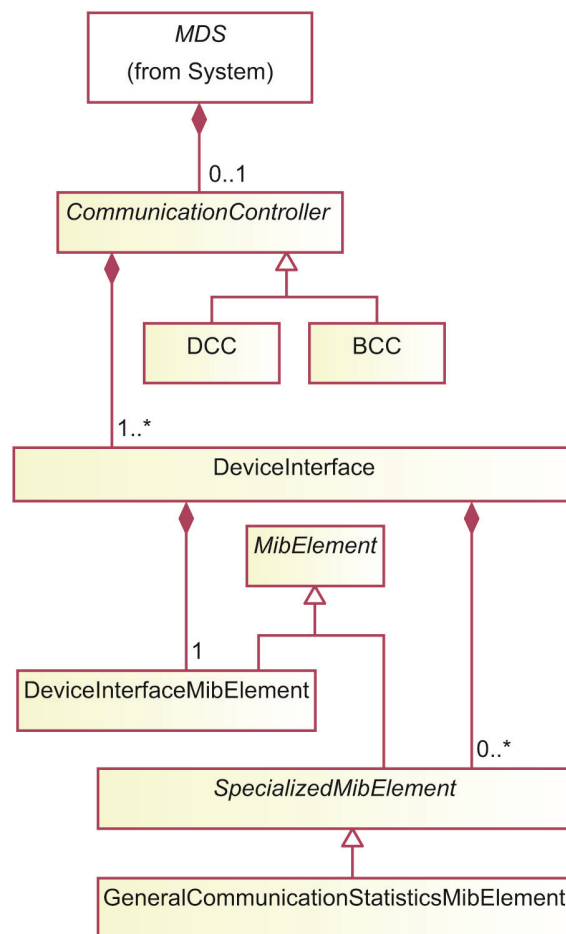


Figure 10—Communication Package model

5.8.2 CommunicationController class

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

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

The CommunicationController class is abstract and cannot be instantiated. Two instantiable subclasses are defined: BCC and DCC. A medical device MDIB contains either no CommunicationController object or one BCC object or one DCC object (depending on its role).

5.8.3 DCC (device communication controller) class

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

DCC objects shall contain one or more DeviceInterface objects.

5.8.4 BCC (bedside communication controller) class

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

BCC objects shall contain one or more DeviceInterface objects.

5.8.5 DeviceInterface class

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

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

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

5.8.6 MibElement class

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

The MibElement class is abstract and cannot be instantiated

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

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

5.8.7 DeviceInterfaceMibElement class

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

5.8.8 SpecializedMibElement class

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

5.8.9 GeneralCommunicationStatisticsMibElement class

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

5.9 Model for the Archival Package

5.9.1 General

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

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

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

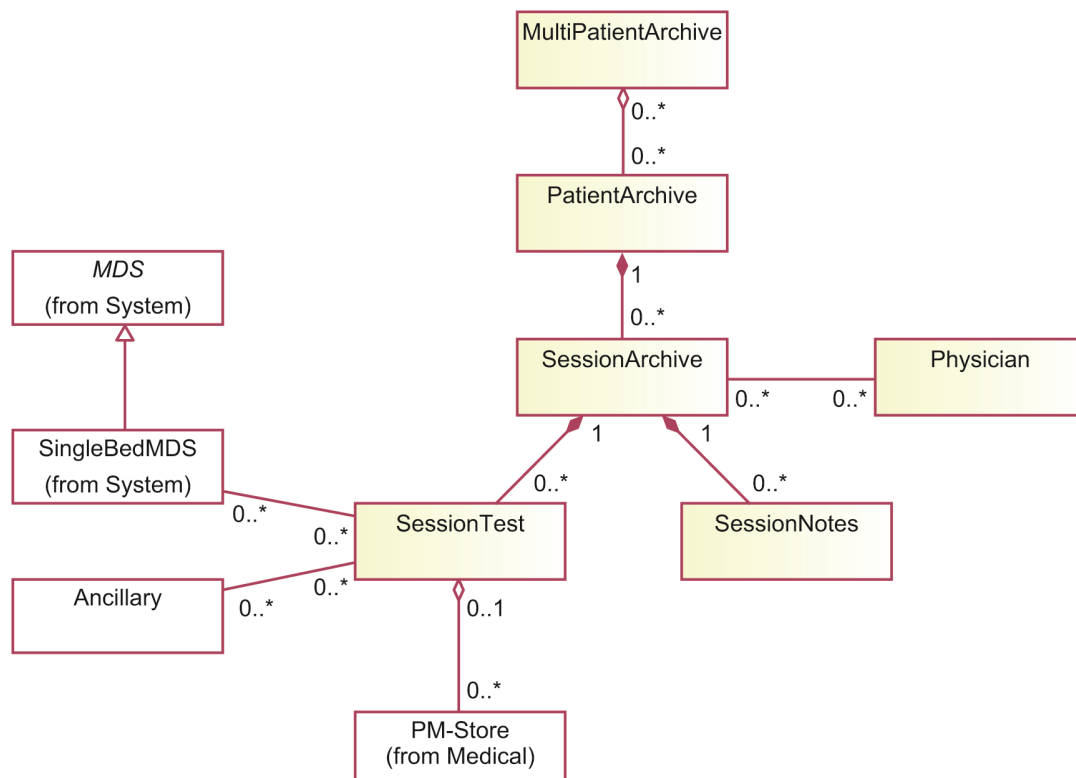


Figure 11 —Archival Package model

5.9.2 MultipatientArchive class

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

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

5.9.3 PatientArchive class

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

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

5.9.4 SessionArchive class

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

5.9.5 PhysicianArchive class

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

5.9.6 SessionTest class

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

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

5.9.7 SessionNotes class

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

5.9.8 Ancillary class

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

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

5.10 Model for the Patient Package

5.10.1 General

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

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

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

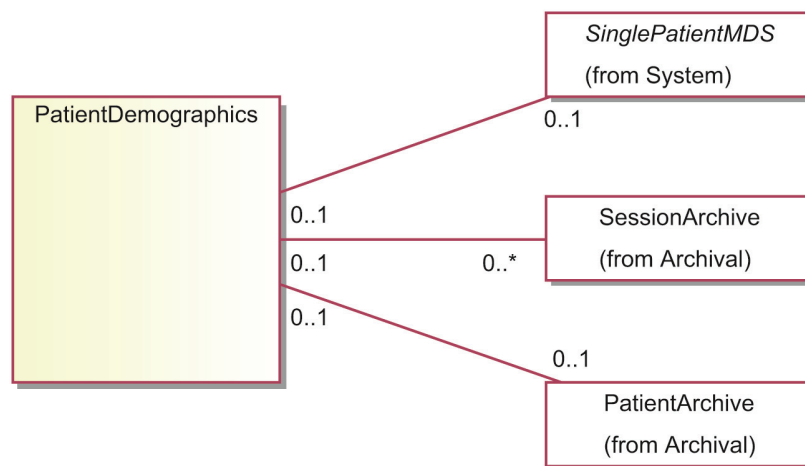


Figure 12—Patient Package model

5.10.2 PatientDemographics class

Instances of the PatientDemographics class store patient census data.

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

5.11 DIM—Dynamic model

5.11.1 General

Subclause 5.11 defines global dynamic system behavior.

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

5.11.2 MDS communication finite state machine (FSM)

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

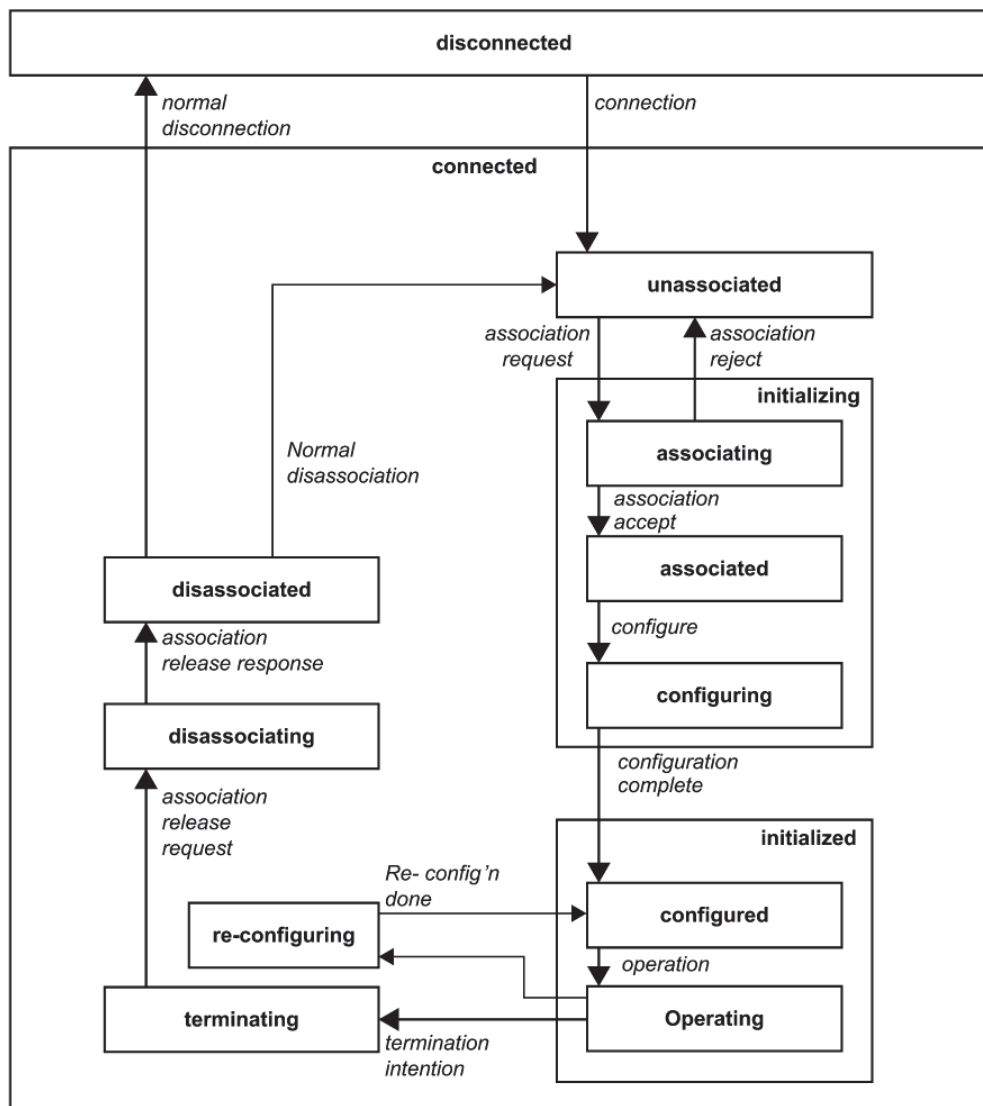


Figure 13—MDS FSM

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.

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

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

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

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

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

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

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

5.11.3 Communicating systems—Startup object interaction diagram

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

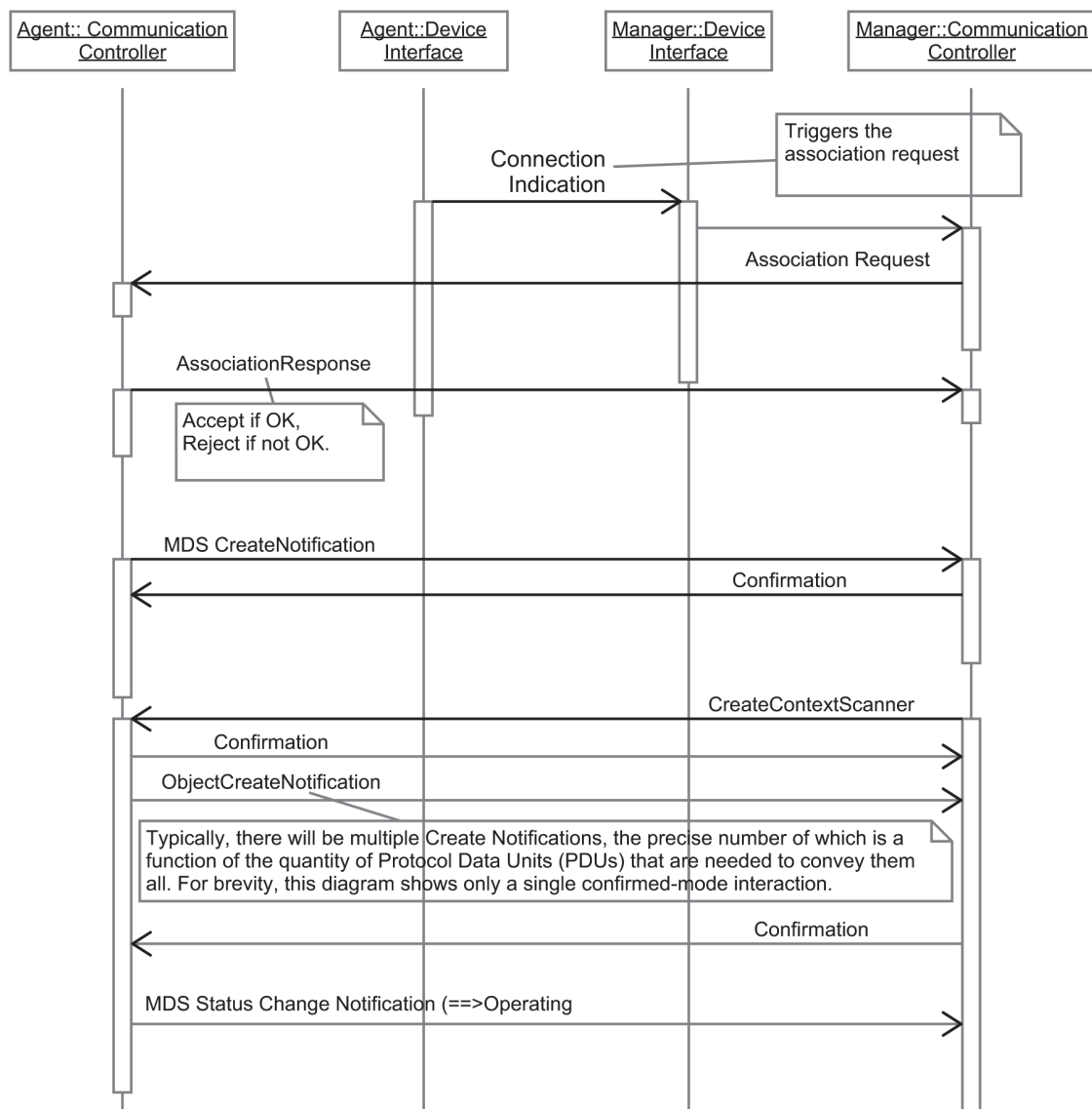


Figure 14—Startup after connection

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

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

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

5.11.4 Communication Package—MibElement data access

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

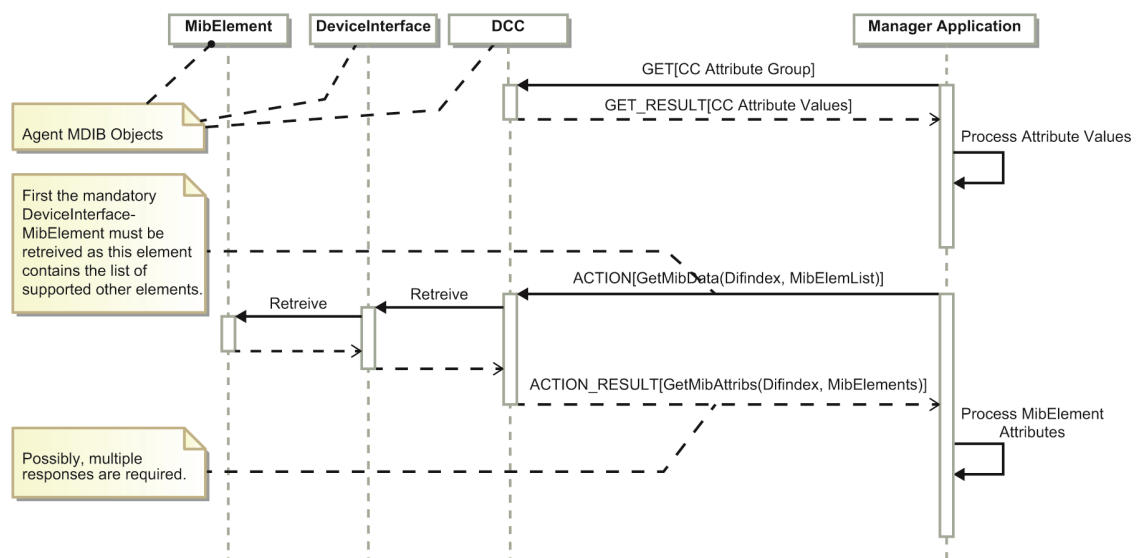


Figure 15—MibElement data access

The diagram assumes the following:

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

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

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

5.11.5 Dynamic object relations

5.11.5.1 General

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

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

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

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

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

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

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

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

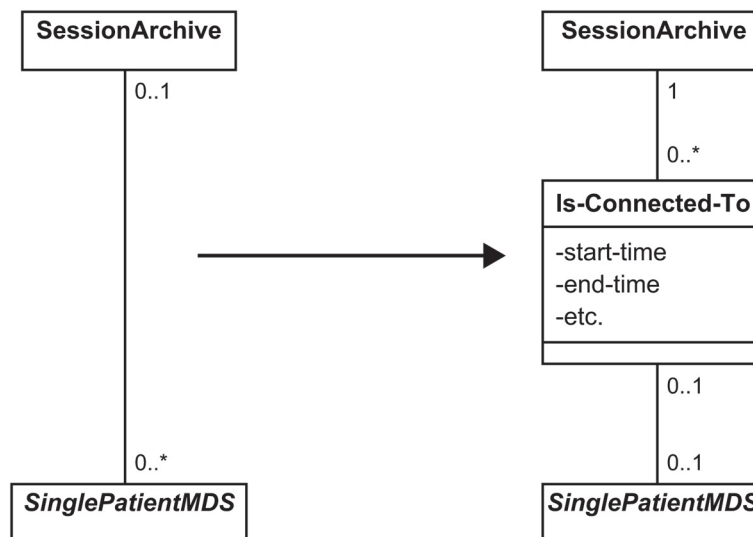


Figure 16—Example of a relationship represented by an object

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

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

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

5.11.5.2 Dynamic object relations in communicating systems

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

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

5.11.5.3 Dynamic object relations in archival systems

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

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

6. DIM class definitions

6.1 Overview

6.1.1 General

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

6.1.2 Notation

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

The class is defined in a subclause as follows:

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

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

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

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

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

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

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

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

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

6.1.3 Common data types

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

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

6.1.3.1 Integer and bit string data types

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

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

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

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

--
-- 16-bit signed integer
```

```
--
INT-I16 ::= INTEGER (-32768..32767)

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

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

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

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

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

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

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

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

6.1.3.2 Identification data type

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

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

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

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

```
--
-- Private OID
```

```
--
PrivateOid ::= INT-U16
```

6.1.3.3 Handle data type

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

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

6.1.3.4 Instance number data type

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

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

6.1.3.5 Global object identification

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

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

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

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

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

Example: A medical device may interface with further medical devices (i.e., sub-devices). In the MDIB, this device may model these sub-devices as individual MDS objects with their own naming context. In this way, name space collisions (e.g., duplicate handle values, duplicate nomenclature codes) can be avoided without reassigning handle values. A manager system needs to interpret the

MDS context IDs together with handle values to uniquely identify object instances within this composite MDIB. The context IDs are assigned when the MDIB is created by ContextScanner object create notifications.

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

6.1.3.6 Type ID data type

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

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

--
-- The following nomenclature partitions exist
--
NomPartition ::= INT-U16 {
    nom-part-unspec(0),
    nom-part-obj(1),           -- object-oriented partition
    nom-part-metric(2),       -- metric [supervisory control and data acquisition (SCADA)] partition
    nom-part-alert(3),        -- alerts/events partition
    nom-part-dim(4),          -- dimensions partition
    nom-part-vattr(5),        -- virtual attribute partition for Operation objects
    nom-part-pgrp(6),         -- parameter group ID partition
    nom-part-sites(7),        -- measurement and body site locations
    nom-part-infrastruct(8),   -- infrastructure elements partition
    nom-part-feff(9),         -- file exchange format partition
    nom-part-ecg-extn(10),    -- ECG extensions partition
    nom-part-icdo(11),        -- IDCO partiion
    nom-part-phddm(128),      -- phd disease management partition
    nom-part-hf(129),         -- health and fitness partition
    nom-part-ageind(130),     -- aging independently partition
    nom-part-returncodes(255), -- return codes patition
    nom-part-ext-nom(256),    -- IDs of other nomenclatures and dictionaries
    nom-part-settings(258),   -- settings partition
    nom-part-priv(1024)       -- private partition
}
```

6.1.3.7 Attribute value assertion data type

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

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