INTERNATIONAL STANDARD

ISO/IEEE 11073-10471

First edition
2010-05-01

Health informatics — Personal health device communication —

Part 10471: Device specialization — Independant living activity hub

Informatique de santé — Communication entre dispositifs de santé personnels —

Partie 10471: Spécialisation des dispositifs — Concentrateur d'activité vivante indépendante

Reference number
ISO/IEEE 11073-10471:2010(E)

Contents

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.

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.

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

Attention is called to the possibility that implementation of this standard may require the use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. ISO/IEEE is not responsible for identifying essential patents or patent claims for which a license may be required, for conducting inquiries into the legal validity or scope of patents or patent claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance or a Patent Statement and Licensing Declaration Form, 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 ISO or the IEEE Standards Association.

ISO/IEEE 11073-10471 was prepared by the 11073 Committee of the Engineering in Medicine and Biology Society of the IEEE (as IEEE Std 11073-10471-2008). It was adopted by Technical Committee ISO/TC 215, *Health informatics*, in parallel with its approval by the ISO member bodies, under the “fast-track procedure” defined in the Partner Standards Development Organization cooperation agreement between ISO and IEEE. Both parties are responsible for the maintenance of this document.

ISO/IEEE 11073 consists of the following parts, under the general title *Health informatics — Personal health device communication (text in parentheses gives a variant of subtitle)*:

— *Part 10101: (Point-of-care medical device communication) Nomenclature*

— *Part 10201: Domain information model*

— *Part 10404: Device specialization — Pulse oximeter*

— *Part 10407: Device specialization — Blood pressure monitor*

— *Part 10408: (Point-of-care medical device communication) Device specialization — Thermometer*

— *Part 10415: (Point-of-care medical device communication) Device specialization — Weighing scale*

— *Part 10417: Device specialization — Glucose meter*

— *Part 10471: (Point-of-care medical device communication) Device specialization — Independant living activity hub*

— *Part 20101: (Point-of-care medical device communication) Application profiles — Base standard*

— *Part 20601: (Point-of-care medical device communication) Application profile — Optimized exchange protocol*

— *Part 30200: (Point-of-care medical device communication) Transport profile — Cable connected*

— *Part 30300: (Point-of-care medical device communication) Transport profile — Infrared wireless*

Introduction

ISO/IEEE 11073 standards enable communication between medical devices and external computer systems. This document uses the optimized framework created in IEEE Std 11073-20601[a] and describes a specific, interoperable communication approach for independent living activity hubs. These standards align with, and draw upon, the existing clinically focused standards to provide support for communication of data from clinical or personal health devices.

[a] For information on references, see Clause 2.

Health informatics — Personal health device communication —

Part 10471: Device specialization — Independant living activity hub

IMPORTANT NOTICE: This standard is not intended to assure safety, security, health, or environmental protection in all circumstances. Implementers of the standard are responsible for determining appropriate safety, security, environmental, and health practices or regulatory requirements.

This IEEE document is made available for use subject to important notices and legal disclaimers. These notices and disclaimers appear in all publications containing this document and may be found under the heading "Important Notice" or "Important Notices and Disclaimers Concerning IEEE Documents." They can also be obtained on request from IEEE or viewed at http://standards.ieee.org/IPR/disclaimers.html.

1. Overview

1.1 Scope

Within the context of the ISO/IEEE 11073 family of standards for device communication, this standard establishes a normative definition of the communication between independent living activity hubs and managers (e.g., cell phones, personal computers, personal health appliances, and set top boxes) in a manner that enables plug-and-play interoperability. It leverages appropriate portions of existing standards including ISO/IEEE 11073 terminology and information models. It specifies the use of specific term codes, formats, and behaviors in telehealth environments restricting ambiguity in base frameworks in favor of interoperability. This standard defines a common core of communication functionality for independent living activity hubs. In this context, independent living activity hubs are defined as devices that communicate with simple situation monitors (binary sensors), normalize information received from the simple environmental monitors, and provide this normalized information to one or more managers. This information can be examined (for example) to determine when a person's activities/behaviors have deviated significantly from what is normal for them such that relevant parties can be notified. Independent living activity hubs will normalize information from the following simple situation monitors (binary sensors) for the initial release of the proposed standard: fall sensor, motion sensor, door sensor, bed/chair occupancy sensor, light switch sensor, smoke sensor, (ambient) temperature threshold sensor, personal emergency response system (PERS), and enuresis sensor (bed-wetting).

1.2 Purpose

This standard addresses a need for an openly defined, independent standard for controlling information exchange to and from personal health devices and managers (e.g., cell phones, personal computers, personal health appliances, and set top boxes). Interoperability is the key to growing the potential market for these devices and to enabling people to be better informed participants in the management of their health.

1.3 Context

See IEEE Std 11073-20601™ for an overview of the environment within which this standard is written.

This document, IEEE Std 11073-10471, defines the device specialization for the independent living activity hub, being a specific agent type, and it provides a description of the device concepts, its capabilities, and its implementation according to this standard.

This standard is based on IEEE Std 11073-20601, which in turn draws information from both ISO/IEEE 11073-10201:2004 [B3][1] and ISO/IEEE 11073-20101:2004 [B4]. The medical device encoding rules (MDERs) used within this standard are fully described in IEEE Std 11073-20601.

This standard reproduces relevant portions of the nomenclature found in ISO/IEEE 11073-10101:2004 [B2] and adds new nomenclature codes for the purposes of this standard. Between this standard and IEEE Std 11073-20601 all required nomenclature codes for implementation are documented.

NOTE— In this standard, IEEE Std 11073-104zz is used to refer to the collection of device specialization standards that utilize IEEE Std 11073-20601, where zz can be any number from 01 to 99, inclusive.[2]

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so that 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.

IEEE Std 11073-20601™-2008, Health informatics—Personal health device communication—Part 20601: Application profile—Optimized Exchange Protocol.[3,4]

See Annex A for all informative material referenced by this standard.

3. Definitions, acronyms, and abbreviations

For the purposes of this standard, the following terms and definitions apply. *The Authoritative Dictionary of IEEE Standards* [B1] should be referenced for terms not defined in this clause.

[1]The numbers in brackets correspond to those of the bibliography in Annex A.
[2]Notes in text, tables, and figures are given for information only and do not contain requirements needed to implement the standard.
[3] The IEEE standards or products referred to in this clause are trademarks of the Institute of Electrical and Electronics Engineers, Inc.
[4] IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org/).

3.1 Definitions

3.1 agent: A node that collects and transmits personal health data to an associated manager.

3.2 class: In object-oriented modeling, it describes the attributes, methods, and events that objects instantiated from the class utilize.

3.3 compute engine: *See:* **manager.**

3.4 device: A term used to refer to a physical apparatus implementing either an agent or a manager role.

3.5 handle: An unsigned 16-bit number that is locally unique and identifies one of the object instances within an agent.

3.6 manager: A node receiving data from one or more agent systems. Some examples of managers include a cellular phone, health appliance, set top box, or a computer system.

3.7 obj-handle: *See*: **handle.**

3.8 object: In object-oriented modeling, a particular instantiation of a class. The instantiation realizes attributes, methods, and events from the class.

3.9 personal health device: A device used in personal health applications.

3.10 personal telehealth device: *See*: **personal health device.**

3.11 sensor: An apparatus that measures physical properties. These comprise the primary inputs to an independent living activity hub agent.

3.2 Acronyms and abbreviations

APDU	application protocol data unit
ASN.1	Abstract Syntax Notation One
DIM	domain information model
EUI-64	extended unique identifier (64 bits)
ICS	implementation conformance statement
MDC	medical device communication
MDER	medical device encoding rules
MDS	medical device system
MOC	managed object class
PDU	protocol data unit
PERS	personal emergency response system
PHD	personal health device
RT-SA	real-time sample array
VMO	virtual medical object
VMS	virtual medical system

4. Introduction to ISO/IEEE 11073 personal health devices

4.1 General

This standard and the remainder of the series of ISO/IEEE 11073 personal health device (PHD) standards fit in the larger context of the ISO/IEEE 11073 series of standards. The full suite of standards enables agents to interconnect and interoperate with managers and with computerized health-care information systems. See IEEE Std 11073-20601 for a description of the guiding principles for this series of ISO/IEEE 11073 personal health device standards.

IEEE Std 11073-20601 supports the modeling and implementation of an extensive set of personal health devices. This standard defines aspects of the independent living activity hub device. It describes all aspects necessary to implement the application layer services and data exchange protocol between an ISO/IEEE 11073 PHD independent living activity hub agent and a manager. This standard defines a subset of the objects and functionality contained in IEEE Std 11073-20601 and extends and adds definitions where appropriate. All new definitions are given in Annex B in abstract syntax notation one (ASN.1) [B5]. Nomenclature codes referenced in this standard, which are not defined in IEEE Std 11073-20601, are normatively defined in Annex C.

4.2 Introduction to IEEE Std 11073-20601 modeling constructs

4.2.1 General

The ISO/IEEE 11073 series of standards, and in particular IEEE Std 11073-20601, is based on an object-oriented systems management paradigm. The overall system model is divided into three principal components: the domain information model (DIM), the service model, and the communication model. See IEEE Std 11073-20601 for a detailed description of the modeling constructs.

4.2.2 Domain information model

The DIM is a hierarchical model that describes an agent as a set of objects. These objects and their attributes represent the elements that control behavior and report on the status of the agent and data that an agent can communicate to a manager. Communication between the agent and the manager is defined by the application protocol in IEEE Std 11073-20601.

4.2.3 Service model

The service model defines the conceptual mechanisms for the data exchange services. Such services are mapped to messages that are exchanged between the agent and the manager. Protocol messages within the ISO/IEEE 11073 series of standards are defined in ASN.1. The messages defined in IEEE Std 11073-20601 can coexist with messages defined in other standard application profiles defined in the ISO/IEEE 11073 series of standards.

4.2.4 Communication model

In general, the communication model supports the topology of one or more agents communicating over logical point-to-point connections to a single manager. For each logical point-to-point connection, the dynamic system behavior is defined by a connection state machine as specified in IEEE Std 11073-20601.

4.2.5 Implementing the models

An agent implementing this standard shall implement all mandatory elements of the information, service, and communication models as well as all conditional elements where the condition is met. The agent should implement the recommended elements, and it may implement any combination of the optional elements. A manager implementing this standard shall utilize at least one of the mandatory, conditional, recommended, or optional elements. In this context, “utilize” means to use the element as part of the primary function of the manager device. For example, a manager whose primary function is to display data would need to display a piece of data in the element in order to utilize it.

5. Independent living activity hub device concepts and modalities

5.1 General

This clause presents the general concepts of independent living activity hub devices. In the context of personal health devices in this family of standards, an independent living activity hub is a device that aggregates activity data sensor events from multiple sensor data sources, all of which are used in the support of the independent living of one or more occupants. The occupants' environment may vary greatly and encompass a varying mixture of sensors; therefore, the activity data sensor events reported by any particular agent have a corresponding variance.

5.2 Concepts

While there are many data generating sensors, they have a number of properties in common that influence the design of this standard. Note that these are only generalities employed in the design and there may be instances where an activity data generating sensor exceeds these properties.

- Price: usually they are very inexpensive sensors.
- Power: typically they are very low power sensors.
- Communication: typically they use a very inexpensive communication technique and are quite often wireless.
- Frequency: typically they communicate very infrequently or only when an event occurs.
- Quantity: there may be a wide range of sensors and many instances of any one sensor type.

It is the responsibility of the independent living activity hub agent to manage all these sensors. Management of the sensors is likely to be within the proprietary purview of the agent both because there is no acceptable existing industry standard and the desire to integrate existing legacy solutions with the IEEE Std 11073-20601 framework. Therefore, this responsibility is outside the scope of this standard. Only the communication between the independent living activity hub agent and the manager are covered by this standard.

Additionally, due to the many different types of sensors that may be employed in any particular installation, there is a range of functionality that the corresponding independent living activity hub agent presents to the manager. On the one hand, a fully functional independent living activity hub agent could represent a significant number of sensors and have a complex conversation with the associated manager. On the other hand, a subset of this protocol could be employed by just one sensor such that it would appear to a manager as an independent living activity hub agent with a single sensor.

5.3 Collected data

5.3.1 General

This clause provides an overview of the kinds of sensors and activity data that could be collected. This is not to imply that all independent living activity hub agents would necessarily report values for all of these sensors. Furthermore, this standard is not concerned with the form of the data nor the communication between any actual sensor and the independent living activity hub; rather only the activity data sensor events derived as a result of that data are considered part of this standard. See Clause 6 for the normative definition of this derived data.

5.3.2 Fall sensor

This sensor is used to notify the monitoring system that a fall sensor event has taken place. This could take the form of a sensor of the type that detects a person's fall and automatically generates the event.

5.3.3 PERS sensor

This sensor is used to notify the monitoring system that a personal emergency sensor event has taken place. This would typically take the form of a button that the person presses to indicate some sort of perceived emergency ("panic button").

5.3.4 Environmental sensors

These sensors generate a sensor event whenever they sense an environmental aspect that is beyond a preset threshold. Examples include smoke sensors, carbon monoxide sensors, water sensors, and natural gas/LP gas sensors.

5.3.5 Motion sensor

This sensor generates a sensor event whenever it has sensed movement within its range above a preset level. This type of sensor is typically employed in two manners: general motion and intruder detection.

In the case of general motion, the detection of the motion causes an immediate generation of a sensor event and subsequent action. This may be used for tracking the activity level of the occupant to discern whether behavior patterns have altered.

In the case of intruder detection, the motion sensor event could be used to trigger intruder detection actions.

Optionally, in the case of the primary entrances to the building, for example, the subsequent action to the sensor event may be delayed for some period before taken. This is to allow for an authorized person to enter the premises and to disable the sensor event before the action is taken (e.g., in the case where a triggered sensor event will generate an intruder alert). Should the proper disabling not take place within the expected delay time period, the normal subsequent action would take place. There may also be a sensor event for the case when the motion sensor detects someone attempting to tamper with its function.

5.3.6 Property exit sensor

This sensor generates a sensor event whenever it has sensed the exit of an occupant from the premises. This is commonly employed when there is an occupant with some cognitive issues who would encounter difficulties in an unfamiliar environment. There may also be a sensor event for the case where the premises' exit is left open.

5.3.7 Enuresis sensor

This sensor generates a sensor event when it detects occurrences of involuntary urination or bed-wetting. This sensor could be utilized in a range of settings such as a bed, chair, or any similar structure.

5.3.8 Contact closure sensor

This sensor issues a sensor event whenever a contact is opened or closed. This sensor reports the state of the contact after a transition, either from closed to open or from open to closed. Only a single sensor event is sent for each transition. Examples of where this sensor would be deployed are passageway doors, cupboard doors, drawers, windows, and pressure mats.

5.3.9 Usage sensor

This sensor issues a sensor event to denote the start of use (into a bed/chair, for example) or the end of use (out of a bed/chair, for example). It also issues a sensor event for an anticipated usage not occurring by an expected preset time (expected use start violation) as well as a sensor event for the usage continuing beyond an expected preset time (expected use stop violation). Additionally, there would be a sensor event generated if during an expected usage time, the usage is discontinued for longer than a preset period of time (intermittent absence violation). An example would be that during a normal sleep period, a person got out of the bed and did not return in an expected period of time.

5.3.10 Switch use sensor

This sensor issues a sensor event for a switch changing states either to the used state (ON) or to the unused state (OFF). Examples of this are light switches, fan switches, and other similar switches that control electrical apparatus.

5.3.11 Simple medication dispenser

The dispenser is a container that contains doses of one or more medications. The medications are to be taken in predetermined doses at predetermined times. A sensor event is generated for a presented dosage being taken from the dispenser (dosage taken) and/or for a dose not being taken after a predetermined amount of time (dosage missed).

NOTE—The sensor description presented here is of a conceptual model to aid in understanding. Be aware that the same derived sensor events could be generated through many other means (such as a user interaction with some screen interface). This standard is only concerned with the generated sensor events.

5.3.12 Temperature sensor

This sensor monitors the temperature in an environment. It issues sensor events based on the sensor value being outside of a preset temperature limit. The sensor events could be that the ambient temperature has risen above a certain level (high threshold) or dropped below a certain level (low threshold), or that the rate-of-change is faster than a predetermined expected rate (rate-of-change). This can be used for detecting conditions such as the temperature of a dwelling being dangerously high/low or that stove elements have been left on after cooking has been completed.

6. Independent living activity hub domain information model

6.1 Overview

This clause describes the domain information model of the independent living activity hub.

6.2 Class extensions

In this standard, no class extensions are defined with respect to IEEE Std 11073-20601.

6.3 Object instance diagram

The object instance diagram of the independent living activity hub domain information model, defined for the purposes of this standard, is shown in Figure 1.

The generic DIM of the independent living activity hub that is presented in Figure 1 defines all possible data objects. However, it would be expected that most independent living activity hubs would implement only a restricted subset of the data objects. An independent living activity hub shall implement at least one sensor instance.

The objects of the DIM, as shown in Figure 1, are described in 6.4 to 6.12. This includes the medical device system (MDS) object (see 6.5), the numeric objects (see 6.6), the real-time sample array (RT-SA) objects (see 6.7), the enumeration objects (see 6.8), the PM-store objects (see 6.9), and the scanner objects (see 6.10). See 6.12 for rules for extending the independent living activity hub information model beyond elements as described in this standard. Each clause that describes an object of the independent living activity hub contains the following information:

— The nomenclature code used to identify the class of the object. One example of where this code is used is the configuration event, where the object class is reported for each object. This allows the manager to determine whether the class of the object being specified is a numeric, real-time sample array, enumeration, scanner, or PM-store class.

— The attributes of the object. Each object has attributes that represent and convey information on the activity data generating sensor and its data sources. Each object has a Handle attribute that identifies the object instance within an agent. Attribute values are accessed and modified using communication services such as GET and SET. Attributes types are defined using ASN.1. The ASN.1 definitions for new attribute types specific to this standard are in Annex B, and the ASN.1 definitions for existing attribute types referenced in this standard are in IEEE Std 11073-20601.

— The methods available on the object.

— The potential events generated by the object. Data are sent to the manager using events.

— The available services such as getting or setting attributes.

The attributes for each class are defined in tables that specify the name of the attribute, its value, and its qualifier. The qualifiers mean M — Attribute is Mandatory, C — Attribute is Conditional and depends on the condition stated in the Remark or Value column (if IEEE Std 11073-20601 is referenced, then it contains the conditions), R — Attribute is Recommended, NR — Attribute is Not Recommended, and O — Attribute is Optional. Mandatory attributes shall be implemented by the agent. Conditional attributes shall be implemented if the condition applies and may be implemented otherwise. Recommended attributes should be implemented by the agent. Not recommended attributes should not be implemented by the agent. Optional attributes may be implemented by the agent.

The attributes can be either static, meaning that they shall remain unchanged after the configuration is agreed on, or dynamic, meaning that the attribute may change at some point after configuration.

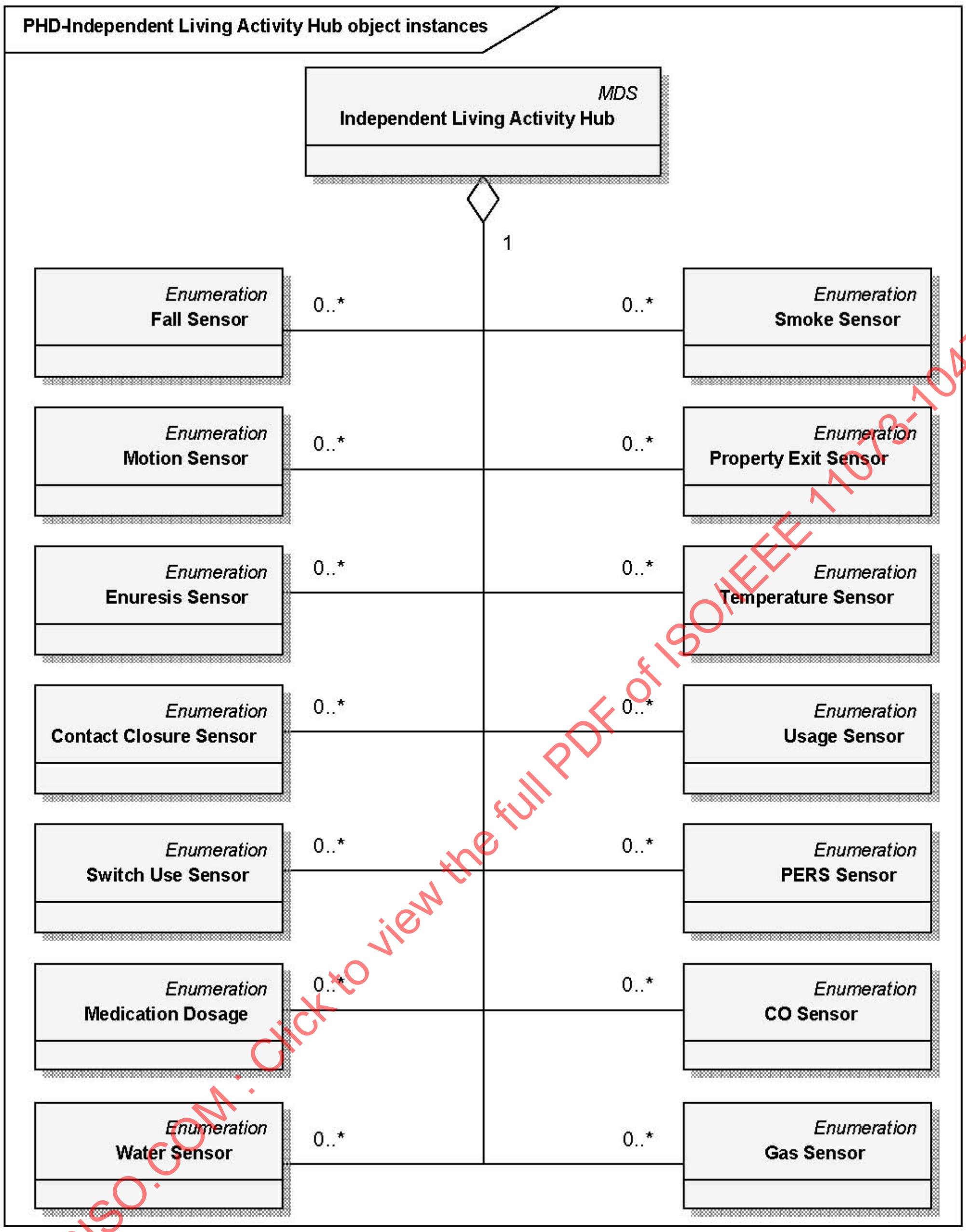

Figure 1—Independent living activity hub—object instances

6.4 Types of configuration

6.4.1 General

As specified in IEEE Std 11073-20601, there are two styles of configuration available. Subclauses 6.4.2 and 6.4.3 briefly introduce standard and extended configurations.

6.4.2 Standard configuration

This standard does not define any standard configurations since the set of sensors for each agent configuration is likely to vary significantly for each deployment scenario. Therefore, all configurations shall be specified as extended configurations.

6.4.3 Extended configuration

In extended configurations, the agent's configuration is not predefined in a standard. The agent determines which objects, attributes, and values that it wants to use in a configuration and assigns a configuration identifier. When the agent associates with a manager, it negotiates an acceptable configuration. Typically, the manager does not recognize the agent's configuration on the first connection, so the manager responds that the agent must send its configuration information as a configuration event report. If, however, the manager already understands the configuration, either because it was preloaded in some way or the agent had previously associated with the manager, then the manager responds that the configuration is known and no further configuration information needs to be sent.

6.5 Medical device system object

6.5.1 MDS object attributes

Table 1 summarizes the attributes of the independent living activity hub MDS object. The nomenclature code to identify the MDS class is MDC_MOC_VMS_MDS_SIMP.

Table 1—MDS object attributes

Attribute name	Value	Qual.
Handle	0	M
System-Type	See IEEE Std 11073-20601.	C
System-Model	{"Manufacturer","Model"}.	M
System-Id	EUI-64.	M
Dev-Configuration-Id	Extended configs: 0x4000–0x7FFF.	M
Attribute-Value-Map	See IEEE Std 11073-20601.	C
Production-Specification	See IEEE Std 11073-20601.	O
Mds-Time-Info	See IEEE Std 11073-20601.	C
Date-and-Time	See IEEE Std 11073-20601.	M
Relative-Time	See IEEE Std 11073-20601.	O
HiRes-Relative-Time	See IEEE Std 11073-20601.	O
Date-and-Time-Adjustment	See IEEE Std 11073-20601.	C
Power-Status	*onBattery* or *onMains.*	R
Battery-Level	See IEEE Std 11073-20601.	R
Remaining-Battery-Time	See IEEE Std 11073-20601.	R
Reg-Cert-Data-List	See IEEE Std 11073-20601.	O
System-Type-Spec-List	{MDC_DEV_SPEC_PROFILE_AI_ACTIVITY_HUB,1}.	M
Confirm-Timeout	See IEEE Std 11073-20601.	O

NOTE—See IEEE Std 11073-20601 for information on whether an attribute is static or dynamic.

In the response to a Get MDS object command, only implemented attributes and their corresponding values are returned.

See IEEE Std 11073-20601 for descriptive explanations of the individual attributes as well as for information on attribute ID and attribute type.

The Dev-Configuration-Id attribute holds a locally unique 16-bit identifier that identifies the device configuration. For an independent living activity hub agent with extended configuration, this identifier is chosen in the range of extended-config-start to extended-config-end (see IEEE Std 11073-20601) as shown in Table 1.

The agent sends the Dev-Configuration-Id during the Associating state (see 8.3) to identify its configuration for the duration of the association. If the manager already holds the configuration information relating to the Dev-Configuration-Id, it recognizes the Dev-Configuration-Id. Then the Configuring state (see 8.4) is skipped; the agent and manager then enter the Operating state. If the manager does not recognize the Dev-Configuration-Id, the agent and manager enter the Configuring state.

If an agent implements multiple IEEE 11073-104zz specializations, System-Type-Spec-List is a list of type/version pairs, each referencing the respective device standard and version of that standard.

6.5.2 MDS object methods

Table 2 defines the methods (actions) of the MDS object. These methods are invoked using the Action service. In Table 2, the Subservice type name column defines the name of the method; the Mode column defines whether the method is invoked as an unconfirmed action (i.e., roiv-cmip-action from IEEE Std 11073-20601) or a confirmed action (i.e., roiv-cmip-confirmed-action); the Subservice type (action-type) column defines the nomenclature code used in the action-type field of an action request and response (see IEEE Std 11073-20601); the Parameters (action-info-args) column defines the associated ASN.1 data structure (see IEEE Std 11073-20601 for ASN.1 definitions) to use in the action message for the action-info-args field of the request; and the Results (action-info-args) column defines the structure to use in the action-info-args of the response.

Table 2—MDS object methods

Service	**Subservice type name**	**Mode**	**Subservice type (action-type)**	**Parameters (action-info-args)**	**Results (action-info-args)**
ACTION	Set-Time	Confirmed	MDC_ACT_SET_TIME	SetTimeInvoke	—

Set-Time
This method allows the manager to set a real-time clock in the agent with the absolute time. The agent indicates whether the Set-Time command is valid using the mds-time-capab-set-clock bit in the Mds-Time-Info attribute (see IEEE Std 11073-20601). Agents with an internal real-time clock (RTC) shall indicate this capability by also setting the mds-time-capab-real-time-clock bit in the Mds-Time-Info attribute. For agents that contain internal clocks, if the agent is wall powered or has access to a constant nondiminishing supply of power then this support shall be implemented. Otherwise this support may be implemented. This method only affects the time contained in the agent. There is no requirement for the agent to alter the time of the associated sensors in any way.

Agents following only this device specialization and no others shall send event reports (see 6.5.3) using agent-initiated measurement data transmission. Agents following this device specialization as well as others shall send event reports in the appropriate fashion. During the association procedure (see 8.3), DataReqModeCapab shall be set to the appropriate value for the event report style. As a result, the manager shall assume the independent living activity hub agent does not support any of the MDS-Data-Request features (see IEEE Std 11073-20601 for additional information). Thus, implementation of the MDS-Data-Request method/action is not required in this standard and is not shown in Table 2.

6.5.3 MDS object events

Table 3 defines the events sent by the independent living activity hub MDS object:

Table 3—Independent living activity hub MDS object events

Service	**Subservice type name**	**Mode**	**Subservice type (event-type)**	**Parameters (event-info)**	**Results (event-reply-info)**
EVENT REPORT	MDS-Configuration-Event	Confirmed	MDC_NOTI_CONFIG	ConfigReport	ConfigReportRsp
	MDS-Dynamic-Data-Update-Var	Confirmed	MDC_NOTI_SCAN_REPORT_VAR	ScanReportInfoVar	—
	MDS-Dynamic-Data-Update-Fixed	Confirmed	MDC_NOTI_SCAN_REPORT_FIXED	ScanReportInfoFixed	—
	MDS-Dynamic-Data-Update-MP-Var	Confirmed	MDC_NOTI_SCAN_REPORT_MP_VAR	ScanReportInfoMPVar	—
	MDS-Dynamic-Data-Update-MP-Fixed	Confirmed	MDC_NOTI_SCAN_REPORT_MP_FIXED	ScanReportInfoMPFixed	—

— **MDS-Configuration-Event:**
This event is sent by the independent living activity hub agent during the configuring procedure if the manager does not already know the independent living activity hub agent's configuration from past associations or because the manager has not been implemented to recognize the configuration according to the independent living activity hub device standard. The event provides static information about the supported measurement capabilities of the independent living activity hub agent.

— **MDS-Dynamic-Data-Update-Var:**
This event provides dynamic measurement data from the independent living activity hub agent for the independent living activity data objects. These data are reported using a generic attribute list variable format. The event is sent as an unsolicited message by the agent (i.e., an agent-initiated measurement data transmission). See 8.5.3 for more information on unsolicited event reporting.

— **MDS-Dynamic-Data-Update-Fixed:**
This event provides dynamic measurement data from the independent living activity hub agent for the independent living activity data objects. These data are reported in the fixed format defined by the Attribute-Value-Map attribute of the object(s). The event is sent as an unsolicited message by the agent (i.e., an agent-initiated measurement data transmission). See 8.5.3 for more information on unsolicited event reporting.

— **MDS-Dynamic-Data-Update-MP-Var:**
This is the same as MDS-Dynamic-Data-Update-Var but allows inclusion of data from multiple people.

— **MDS-Dynamic-Data-Update-MP-Fixed:**
This is the same as MDS-Dynamic-Data-Update-Fixed but allows inclusion of data from multiple people.

NOTE—IEEE Std 11073-20601 requires that managers support all of the MDS object events listed above.

6.5.4 Other MDS services

6.5.4.1 GET service

An independent living activity hub agent shall support the GET service to retrieve the values of all implemented attributes of the MDS object. The GET service can be invoked as soon as the independent living activity hub agent receives the Association Response and moves to the Associated state, including the Operating and Configuring substates.

The GET request for all attributes shall be supported. An attribute-id-list parameter may be supported.

The manager may request the MDS object attributes of the independent living activity hub agent; in which case the manager shall send the “Remote Operation Invoke | Get” message (see roiv-cmip-get in IEEE Std 11073-20601) with the reserved MDS handle value of 0. The independent living activity hub agent shall report its MDS object attributes to the manager using the “Remote Operation Response | Get” message (see rors-cmip-get in IEEE Std 11073-20601). See Table 4 for a summary of the GET service including some message fields.

Table 4—Independent living activity hub MDS object GET service

Service	Subservice type name	Mode	Subservice type	Parameters	Results
GET	<na>	<implied confirmed>	<na>	GetArgumentSimple=(obj-handle=0), attribute-id-list *<optional>*	GetResultSimple=(obj-handle=0), attribute-list

See 8.5.2 for details on the procedure for getting the MDS object attributes.

6.5.4.2 SET service

The independent living activity hub standard does not require an implementation to support the MDS object SET service.

6.6 Numeric objects

Numeric objects are not required by this standard.

6.7 Real-time sample array objects

Real-time sample array objects are not required by this standard.

6.8 Enumeration objects

6.8.1 General

The independent living activity hub requires one activity data object for each supported sensor instance. Objects instantiated from the enumeration class are used to model the activity data. The nomenclature code to identify the enumeration class is MDC_MOC_VMO_METRIC_ENUM. The attributes of an activity data object are shown in Table 5. The attribute structure shown in Table 5 is common to all sensor event types. Later clauses define the precise definitions for each sensor event type.

Table 5—Activity data enumeration object attributes

Attribute name	Extended configuration	
	Value	**Qual.**
Handle	See IEEE Std 11073-20601.	M
Type	Any of the MDC_PART_PHD_AI | MDC_AI_TYPE_* ID's. See Annex C.	M
Supplemental-Types	The value that denotes the sensor location. This location is a 16-bit value constructed where the high 10 bits are one of the MDC_AI_LOCATION family of constants as denoted in Annex C, and the lower 6 bits are the unique instance of that location. For example, the high 10 bits could be the value for bedroom, and the lower 6 bits could be the value for bedroom 3 (0 being the first bedroom). All these values are from the Partition ID: MDC_PART_PHD_AI.	M
Metric-Spec-Small	mss-avail-intermittent | mss-avail-stored-data | mss-upd-aperiodic | mss-msmt-aperiodic | mss-acc-agent-initiated.	M
Metric-Structure-Small	See IEEE Std 11073-20601.	NR
Measurement-Status	See IEEE Std 11073-20601.	NR
Metric-Id	See IEEE Std 11073-20601.	NR
Metric-Id-List	See IEEE Std 11073-20601.	NR
Metric-Id-Partition	See IEEE Std 11073-20601.	NR
Unit-Code	See IEEE Std 11073-20601.	NR
Attribute-Value-Map	See IEEE Std 11073-20601.	C
Source-Handle-Reference	See IEEE Std 11073-20601.	NR
Label-String	See IEEE Std 11073-20601.	O
Unit-LabelString	See IEEE Std 11073-20601.	O
Absolute-Time-Stamp	See IEEE Std 11073-20601.	R
Relative-Time-Stamp	See IEEE Std 11073-20601.	O
HiRes-Time-Stamp	See IEEE Std 11073-20601.	O
Measure-Active-Period	See IEEE Std 11073-20601.	NR
Enum-Observed-Value-Simple-OID	See IEEE Std 11073-20601.	NR
Enum-Observed-Value-Simple-Bit-Str	This attribute contains the data for this sensor measurement. This attribute consists of two parts: 1) The specific sensor event properties flags are represented by bit(0) through bit(15). 2) The generic sensor health flags are represented by bit(16) through bit(31). The field is the logical "or" of these two values.	M

Attribute name	Extended configuration	
	Value	**Qual.**
	Any valid setting of the specific sensor event flags can be combined with any valid setting of the generic sensor health flags. *Specific sensor event properties flags:* The interpretation of this is specific to the instance type and is enumerated in the following specific sensor clauses. *Generic sensor health properties flags:* See Annex B for ASN.1 definitions. (multiple of the following flags may be set) auto-presence-received(16) (For sensors that have “heartbeat” operational status: indicates that the “heartbeat” has been seen and is ok. This flag shall be reset if Auto-Presence-Failed is set.) auto-presence-failed(17) (For sensors that have “heartbeat” operational status: indicates that the “heartbeat” has not been seen as expected. This flag shall be reset if Auto-Presence-Received is set.) low-battery(18) (Indicates the sensor is in the low battery condition. This determination is unique to the sensor.) fault(19) (Indicates that the sensor is in a fault condition and needs attention. This determination is unique to the sensor.) end-of-life(20) (Indicates that the sensor has reached end of life and needs replacement. This indication is unique to the sensor.) This attribute is of type BITS-32.	M
Enum-Observed-Value-Basic-Bit-Str	See IEEE Std 11073-20601.	NR
Enum-Observed-Value-Simple-Str	See IEEE Std 11073-20601.	NR
Enum-Observed-Value	See IEEE Std 11073-20601.	NR
Enum-Observed-Value-Partition	See IEEE Std 11073-20601.	NR

Subclauses 6.8.2 to 6.8.12 describe the possible uses of the activity data enumeration object. Each use is an instance of the activity data enumeration class with a particular type value. The interpretation of associated values is dependent on the type value. The description of each activity data enumeration object defines all the possible states, and where appropriate, its behavior. The respective tables define the activity data sensor events generated by the agent in response to a change in sensor state.

Unless otherwise defined, a sensor shall always generate a sensor event whenever a sensor changes from the no condition detected (nonactivated) state to another (activated) state. A sensor shall always generate a sensor event should the sensor state subsequently change to another (activated) state. A sensor may generate a sensor event whenever the sensor returns to the no condition detected (nonactivated) state. The no condition event is a generic sensor event that has no implication as to any sensor state. This is merely a value that can be used to delimit the ending of a previous event and/or be used as the specific sensor event value when desiring to issue a generic event value.

The activity data enumeration object does not support any methods, events, or other services.

See IEEE Std 11073-20601 for descriptive explanations on the individual attributes as well as information on attribute ID and attribute type.

6.8.2 Activity data enumeration object—fall sensor

Table 6 is the sensor event specification of the fall sensor (see 5.3.2). It is fundamentally activated or nonactivated. A fall detected sensor event shall be sent whenever a sensor determines a fall has occurred. A no condition detected sensor event may be sent if the sensor can determine such a status.

This object is an instance of the enumeration class.

Table 6—Activity data enumeration object—fall sensor

Attribute name	**Extended configuration**	
	Value	**Qual.**
Handle	See IEEE Std 11073-20601.	M
Type	MDC_AI_TYPE_SENSOR_FALL.	M
Supplemental-Types	The unique value that denotes the location. See Table 5 for definition.	M
Absolute-Time-Stamp	See IEEE Std 11073-20601.	M
Enum-Observed-Value-Simple-Bit-Str	This attribute consists of two parts: 1) The specific sensor event properties flags are represented by bit(0) through bit(15). 2) The generic sensor health flags are represented by bit(16) through bit(31). The field is the logical "or" of these two values. Any valid setting of the specific sensor event flags can be combined with any valid setting of the generic sensor health flags. *Specific sensor event properties flags:* See Annex B for ASN.1 definitions of the FallSensorFlags. (Exactly one flag at most shall be set. In the case of no condition detected, then all specific sensor event flags shall be reset.) fall-detected(0) *Generic sensor health properties flags:* See Table 5 for definition.	M

6.8.3 Activity data enumeration object—PERS sensor

Table 7 is the sensor event specification of the PERS sensor (see 5.3.3). A button-activated sensor event shall be sent whenever the button is pressed. A no condition detected sensor event may be sent when the button is released. A button-activated sensor event is always treated and acted on as an urgent event regardless of whether a no condition detected sensor event is subsequently generated. Should the PERS sensor be mobile, there is no requirement that its location be updated or altered.

This object is an instance of the enumeration class.

NOTE—The above discussion of a button usage to trigger the event is example only. The agent implementation may use any mechanism it desires to accomplish this function.

Table 7—Activity data enumeration object—PERS sensor

Attribute name	**Extended configuration**	
	Value	**Qual.**
Handle	See IEEE Std 11073-20601.	M
Type	MDC_AI_TYPE_SENSOR_PERS.	M
Absolute-Time-Stamp	See IEEE Std 11073-20601.	M
Supplemental-Types	The unique value that denotes the location. See Table 5 for definition.	M
Enum-Observed-Value-Simple-Bit-Str	This attribute consists of two parts: 1) The specific sensor event properties flags are represented by bit(0) through bit(15). 2) The generic sensor health flags are represented by bit(16) through bit(31). The field is the logical "or" of these two values. Any valid setting of the specific sensor event flags can be combined with any valid setting of the generic sensor health flags. *Specific sensor event properties flags:* See Annex B for ASN.1 definitions. (Exactly one flag at most shall be set. In the case of no condition detected, then all specific sensor event flags shall be reset.) pers-activated(0) *Generic sensor health properties flags:* See Table 5 for definition.	M

6.8.4 Activity data—environmental sensor

Table 8 is the sensor event specification of the smoke sensor, carbon monoxide sensor, water sensor, and gas sensor (see 5.3.4). It is fundamentally activated or nonactivated. A condition detected event shall be sent whenever a sensor determines the condition has occurred. A no condition detected sensor event may be sent if the sensor can determine such a status.

This object is an instance of the enumeration class.

Table 8—Activity data enumeration object—environmental sensor

Attribute name	Extended configuration	
	Value	**Qual.**
Handle	See IEEE Std 11073-20601.	M
Type	MDC_AI_TYPE_SENSOR_SMOKE or MDC_AI_TYPE_SENSOR_CO or MDC_AI_TYPE_SENSOR_WATER or MDC_AI_TYPE_SENSOR_GAS.	M
Supplemental-Types	The unique value that denotes the location. See Table 5 for definition.	M
Absolute-Time-Stamp	See IEEE Std 11073-20601.	M
Enum-Observed-Value-Simple-Bit-Str	This attribute consists of two parts: 1) The specific sensor event properties flags are represented by bit(0) through bit(15). 2) The generic sensor health flags are represented by bit(16) through bit(31). The field is the logical “or” of these two values. Any valid setting of the specific sensor event flags can be combined with any valid setting of the generic sensor health flags. *Specific sensor event properties flags:* See Annex B for ASN.1 definitions. (Exactly one flag at most shall be set. In the case of no condition detected then all specific sensor event flags shall be reset.) condition-detected(0) *Generic sensor health properties flags:* See Table 5 for definition.	M

6.8.5 Activity data enumeration object—motion sensor

Table 9 is the sensor event specification of the motion sensor (see 5.3.5). It is fundamentally activated immediate, activated delayed, nonactivated, or tampered. A motion detected event shall be sent whenever a sensor determines that motion has occurred. A motion detected delayed, tamper detected, and no condition detected sensor events may be sent if the sensor can determine such a status.

This object is an instance of the enumeration class.

Table 9—Activity data enumeration object—motion sensor

Attribute name	Extended configuration	
	Value	**Qual.**
Handle	See IEEE Std 11073-20601.	M
Type	MDC_AI_TYPE_SENSOR_MOTION.	M
Supplemental-Types	The unique value that denotes the location. See Table 5 for definition.	M
Absolute-Time-Stamp	See IEEE Std 11073-20601.	M
Enum-Observed-Value-Simple-Bit-Str	This attribute consists of two parts: 1) The specific sensor event properties flags are represented by bit(0) through bit(15). 2) The generic sensor health flags are represented by bit(16) through bit(31). The field is the logical "or" of these two values. Any valid setting of the specific sensor event flags can be combined with any valid setting of the generic sensor health flags. *Specific sensor event properties flags:* See Annex B for ASN.1 definitions. (Exactly one motion flag at most shall be set; the tamper flag may be additionally set at any time. In the case of no condition detected, then all specific sensor event flags shall be reset.) motion-detected(0) motion-detected-delayed(1) tamper-detected(2) *Generic sensor health properties flags:* See Table 5 for definition.	M

6.8.6 Activity data enumeration object—property exit sensor

Table 10 is the sensor event specification of the property exit sensor (see 5.3.6). It is fundamentally activated or nonactivated and may detect if the door of the premises has been left open. An occupant exit detected event shall be sent whenever a sensor determines an occupant exiting event has occurred. An exit door left open and no condition detected sensor event may be sent if the sensor can determine such a status. An example use of this sensor would be in situations where a dwelling had occupants that suffered from Alzheimer's. It could notify when an occupant had strayed from the premises and whether the exit door had been left open.

This object is an instance of the enumeration class.

Table 10 —Activity data enumeration object – property exit sensor

Attribute name	Extended configuration	
	Value	Qual.
Handle	See IEEE Std 11073-20601	M
Type	MDC_AI_TYPE_SENSOR_PROPEXIT.	M
Supplemental-Types	The unique value that denotes the location. See Table 5 for definition.	M
Absolute-Time-Stamp	See IEEE Std 11073-20601	M
Enum-Observed-Value-Simple-Bit-Str	This attribute consists of two parts: 1) The specific sensor event properties flags are represented by bit(0) through bit(15). 2) The generic sensor health flags are represented by bit(16) through bit(31). The field is the logical "or" of these two values. Any valid setting of the specific sensor event flags can be combined with any valid setting of the generic sensor health flags. *Specific sensor event properties flags:* See Annex B for ASN.1 definitions. (Exactly one flag at most shall be set. In the case of no condition detected, then all specific sensor event flags shall be reset.) occupant-exit-property(0) (one or more people have departed via exit door) exit-door-left-open(1) (exit door remains open) *Generic sensor health properties flags:* See Table 5 for definition.	M

6.8.7 Activity data enumeration object—enuresis sensor

Table 11 is the sensor event specification of the bed-wetting (enuresis) sensor (see 5.3.7). It is fundamentally activated or nonactivated. An enuresis detected event shall be sent whenever a sensor determines the condition has occurred. A no condition detected sensor event may be sent if the sensor can determine such a status.

This object is an instance of the enumeration class.

Table 11 —Activity data enumeration object—enuresis sensor

Attribute name	Extended configuration	
	Value	Qual.
Handle	See IEEE Std 11073-20601	M
Type	MDC_AI_TYPE_SENSOR_ENURESIS.	M
Supplemental-Types	The unique value that denotes the location. See Table 5 for definition.	M
Absolute-Time-Stamp	See IEEE Std 11073-20601.	M
Enum-Observed-Value-Simple-Bit-Str	This attribute consists of two parts: 1) The specific sensor event properties flags are represented by bit(0) through bit(15). 2) The generic sensor health flags are represented by bit(16) through bit(31). The field is the logical “or” of these two values. Any valid setting of the specific sensor event flags can be combined with any valid setting of the generic sensor health flags. *Specific sensor event properties flags:* See Annex B for ASN.1 definitions. (Exactly one flag at most shall be set. In the case of no condition detected, then all specific sensor event flags shall be reset.) enuresis-detected(0) *Generic sensor health properties flags:* See Table 5 for definition.	M

6.8.8 Activity data enumeration object—contact closure sensor

Table 12 is the sensor events of the contact closure sensor (see 5.3.8). It is fundamentally activated or nonactivated. A closure closed event and closure opened event shall be sent whenever a sensor determines the condition has occurred. A no condition detected sensor event may be sent if the sensor can determine such a status.

This object is an instance of the enumeration class.

Table 12 —Activity data enumeration object—contact closure sensor

Attribute name	**Extended configuration**	
	Value	**Qual.**
Handle	See IEEE Std 11073-20601.	M
Type	MDC_AI_TYPE_SENSOR_CONTACTCLOSURE.	M
Supplemental-Types	The unique value that denotes the location. See Table 5 for definition.	M
Absolute-Time-Stamp	See IEEE Std 11073-20601.	M
Enum-Observed-Value-Simple-Bit-Str	This attribute consists of two parts: 1) The specific sensor event properties flags are represented by bit(0) through bit(15). 2) The generic sensor health flags are represented by bit(16) through bit(31). The field is the logical “or” of these two values. Any valid setting of the specific sensor event flags can be combined with any valid setting of the generic sensor health flags. *Specific sensor event properties flags:* See Annex B for ASN.1 definitions. (Exactly one flag at most shall be set. In the case of no condition detected, then all specific sensor event flags shall be reset.) contact-opened(0) contact-closed(1) *Generic sensor health properties flags:* See Table 5 for definition.	M

6.8.9 Activity data enumeration object—usage sensor

Table 13 is the sensor event specification of the usage sensor (see 5.3.9). An example of this sensor would be a bed/chair usage sensor. It is fundamentally activated or nonactivated. It may generate further events based on violation of timing constraints based on usage or absence. A usage started event and usage ended event shall be sent whenever a sensor determines the condition has occurred. An expected use start violation, expected use stop violation, absence violation, and no condition detected sensor events may be sent if the sensor can determine such a status.

This object is an instance of the enumeration class.

Table 13 —Activity data enumeration object—usage sensor

Attribute name	Extended configuration	
	Value	**Qual.**
Handle	See IEEE Std 11073-20601.	M
Type	MDC_AI_TYPE_SENSOR_USAGE.	M
Supplemental-Types	The unique value that denotes the location. See Table 5 for definition.	M
Absolute-Time-Stamp	See IEEE Std 11073-20601.	M
Enum-Observed-Value-Simple-Bit-Str	This attribute consists of two parts: 1) The specific sensor event properties flags are represented by bit(0) through bit(15). 2) The generic sensor health flags are represented by bit(16) through bit(31). The field is the logical "or" of these two values. Any valid setting of the specific sensor event flags can be combined with any valid setting of the generic sensor health flags. *Specific sensor event properties flags:* See Annex B for ASN.1 definitions. (Exactly one flag at most shall be set. In the case of no condition detected, then all specific sensor event flags shall be reset.) usage-started(0) (bed/chair in) usage-ended(1) (bed/chair out) expected-use-start-violation(2) (expected usage not started) expected-use-stop-violation(3) (usage continued beyond expected usage end) absence-violation(4) (absent for too long a period during expected usage) *Generic sensor health properties flags:* See Table 5 for definition.	M

6.8.10 Activity data enumeration object—switch sensor

Table 14 is the sensor event specification of the switch use sensor (see 5.3.10). It is fundamentally activated or nonactivated. A switch ON and switch OFF event shall be sent whenever a sensor determines the condition has occurred. A no condition detected sensor event may be sent if the sensor can determine such a status.

This object is an instance of the enumeration class.

Table 14—Activity data enumeration object—switch sensor

Attribute name	Extended configuration	
	Value	Qual.
Handle	See IEEE Std 11073-20601.	M
Type	MDC_AI_TYPE_SENSOR_SWITCH.	M
Supplemental-Types	The unique value that denotes the location. See Table 5 for definition.	M
Absolute-Time-Stamp	See IEEE Std 11073-20601.	M
Enum-Observed-Value-Simple-Bit-Str	This attribute consists of two parts: 1) The specific sensor event properties flags are represented by bit(0) through bit(15). 2) The generic sensor health flags are represented by bit(16) through bit(31). The field is the logical "or" of these two values. Any valid setting of the specific sensor event flags can be combined with any valid setting of the generic sensor health flags. *Specific sensor event properties flags:* See Annex B for ASN.1 definitions. (Exactly one flag at most shall be set. In the case of no condition detected, then all specific sensor event flags shall be reset.) switch-on(0) switch-off(1) *Generic sensor health properties flags:* See Table 5 for definition.	M

6.8.11 Activity data enumeration object—medication dosage sensor

Table 15 is the sensor event specification of the medication dispenser (see 5.3.11). A dosage taken event shall be sent whenever a sensor determines the condition has occurred. A dosage missed and no condition detected sensor event may be sent if the sensor can determine such a status.

This object is an instance of the enumeration class.

Table 15—Activity data enumeration object—medication dosage sensor

Attribute name	Extended configuration	
	Value	**Qual.**
Handle	See IEEE Std 11073-20601.	M
Type	MDC_AI_TYPE_SENSOR_DOSAGE.	M
Supplemental-Types	The unique value that denotes the location. See Table 5 for definition.	M
Absolute-Time-Stamp	See IEEE Std 11073-20601.	M
Enum-Observed-Value-Simple-Bit-Str	This attribute consists of two parts: 1) The specific sensor event properties flags are represented by bit(0) through bit(15). 2) The generic sensor health flags are represented by bit(16) through bit(31). The field is the logical "or" of these two values. Any valid setting of the specific sensor event flags can be combined with any valid setting of the generic sensor health flags. *Specific sensor event properties flags:* See Annex B for ASN.1 definitions. (Exactly one flag at most shall be set. In the case of no condition detected then all specific sensor event flags shall be reset.) dosage-taken(0) dosage-missed(1) *Generic sensor health properties flags:* See Table 5 for definition.	M

6.8.12 Activity data enumeration object—temperature sensor

Table 16 is the sensor event specification of the temperature sensor (see 5.3.12). A high temperature detected and low temperature detected event shall be sent whenever a sensor determines the condition has occurred. A rate of change too fast and no condition detected sensor event may be sent if the sensor can determine such a status.

This object is an instance of the enumeration class.

Table 16—Activity data enumeration object—temperature sensor

Attribute name	Extended configuration	
	Value	**Qual.**
Handle	See IEEE Std 11073-20601.	M
Type	MDC_AI_TYPE_SENSOR_TEMP.	M
Supplemental-Types	The unique value that denotes the location. See Table 5 for definition.	M
Absolute-Time-Stamp	See IEEE Std 11073-20601.	M
Enum-Observed-Value-Simple-Bit-Str	This attribute consists of two parts: 1) The specific sensor event properties flags are represented by bit(0) through bit(15). 2) The generic sensor health flags are represented by bit(16) through bit(31). The field is the logical "or" of these two values. Any valid setting of the specific sensor event flags can be combined with any valid setting of the generic sensor health flags. *Specific sensor event properties flags:* See Annex B for ASN.1 definitions. (Exactly one temperature flag or no condition detected flag at most shall be set; the rate_of_change_too_fast flag may be set at any time. In the case of no condition detected, then all specific sensor event flags shall be reset.) high-temperature-detected(0) low-temperature-detected(1) rate-of-change-too-fast(2) *Generic sensor health properties flags:* See Table 5 for definition.	M

6.9 PM-store objects

PM-store objects are not required by this standard.

6.10 Scanner objects

Scanner objects are not required by this standard.

6.11 Class extension objects

In this standard, no class extension objects are defined with respect to IEEE Std 11073-20601.

6.12 Independent living activity hub information model extensibility rules

The independent living activity hub domain information model of this standard may be extended by including vendor-specific metrics and attributes as required. For example, a vendor might include a location attribute value (i.e., MDC_AI_LOCATION_BEDROOM) that is not included in the standard location attribute value table. Any object or attribute extensions implemented should follow the guidelines of this standard as closely as possible.

All configurations are required to be extended configurations defined by the vendor since this standard does not define any standard configurations.

Vendor nomenclature extensions can be made in the preserved area for extension. In the aging independently partition (MDC_PART_PHD_AI), this is the space 0xF000 to 0xFFFF.

7. Independent living activity hub service model

7.1 General

The service model defines the conceptual mechanisms for data-exchange services. These services are mapped to messages that are exchanged between the agent and the manager. Protocol messages within the ISO/IEEE 11073 series of standards are defined in ASN.1. See IEEE Std 11073-20601 for a detailed description of the personal health device service model. Sublcauses 7.2 through 7.3 define the specifics of object access and event reporting services for an independent living activity hub agent according to this standard.

7.2 Object access services

The object access services of IEEE Std 11073-20601 are used to access the objects defined in the domain information model of the independent living activity hub.

The following generic object access services are supported by an independent living activity hub according to this standard:

— GET service: used by the manager to retrieve the implemented attribute values of the agent MDS object. The list of independent living activity hub MDS object attributes is given in 6.5.1.

— SET service: used by the manager to set the values of the agent object attributes. There are no settable attributes defined for an independent living activity hub agent according to this standard.

— Event report service: used by the agent to send configuration reports and measurement data to the manger. The list of event reports for the independent living activity hub agent standard is given in 6.5.3.

— Action service: used by the manager to invoke actions (or methods) supported by the agent. An example is Set-Time action, which is used to set a real-time clock with the absolute time at the agent.

Table 17 summarizes the object access services described in this standard.

Table 17 —Independent living activity hub object access services

Service	Subservice type name	Mode	Subservice type	Parameters	Result	Remark
GET	<na>	<implied Confirmed>	<na>	GetArgumentSimple=(obj-handle=0), attribute-id-list<optional>	GetResultSimple=(obj-handle=0), attribute-list	Allows the manager to retrieve the value of an attribute of an object in the agent.
EVENT REPORT	MDS-Configuration-Event	Confirmed	MDC_NOTI_CONFIG	ConfigReport	ConfigReportRsp	Configuration Report to inform manager of the configuration of the agent.
	MDS-Scan-Report-Var	Confirmed	MDC_NOTI_SCAN_REPORT_VAR	ScanReportInfoVar	—	Data Report to provide dynamic data to manager for some or all of the agent's objects in variable format.
	MDS-Scan-Report-Fixed	Confirmed	MDC_NOTI_SCAN_REPORT_FIXED	ScanReportInfoFixed	—	Data Report to provide dynamic data to manager for some or all of the agent's objects in fixed format.
	MDS-Scan-Report-MP-Var	Confirmed	MDC_NOTI_SCAN_REPORT_MP_VAR	ScanReportInfoMPVar	—	This is the same as MDS-Dynamic-Data-Update-Var but allows inclusion of data from multiple people.
	MDS-Scan-Report-MP-Fixed	Confirmed	MDC_NOTI_SCAN_REPORT_MP_FIXED	ScanReportInfoMPFixed	—	This is the same as MDS-Dynamic-Data-Update-Fixed but allows inclusion of data from multiple people.
ACTION	Set-Time	Confirmed	MDC_ACT_SET_TIME	SetTimeInvoke	—	Manager method to invoke the agent to set time to requested value.

7.3 Object access event report services

The event report service (see Table 17) is used by the agent to report its information (e.g., measurements). Event reports in this standard are a property of the MDS object only. The event reports used in this standard are defined in IEEE Std 11073-20601.

The following conditions apply for an independent living activity hub agent according to this standard:

- Event reports shall be used in confirmed mode.
- Agent-initiated mode shall be supported for measurement data transmission.

An independent living activity hub agent, which is designed to operate in an environment where data may be collected from multiple people, may use one of the multiple-person event report styles to transmit all the data from each person in a single event report. If this functionality is not required, the agent may use the single-person event report styles, which have reduced overhead.

A manager shall support both single-person and multiple-person event reports. An independent living activity hub agent may support either one or both single-person and multiple-person event reports. The formats for single- and multiple-person event reports are described in IEEE Std 11073-20601.

8. Independent living activity hub communication model

8.1 Overview

This clause describes the general communication model and procedures of the independent living activity hub agent as defined in IEEE Std 11073-20601. Therefore, the respective parts of IEEE Std 11073-20601 are not reproduced; rather the specific choices and restrictions with respect to optional elements (e.g., objects, attributes, and actions) and specific extensions (e.g., nomenclature terms) are specified.

For an illustrative overview of the various message transactions during a typical measurement session, see the sequence diagram for the example use case in Annex D and the corresponding protocol data unit (PDU) examples in Annex E.

8.2 Communication characteristics

In this section, limits on the size of an application protocol data unit (APDU) to be transmitted or received by an independent living activity hub agent are defined. Small limits allow for simple implementations in terms of low cost and complexity.

For an independent living activity hub agent implementing no other device specialization except this standard, the maximum size of an APDU sent shall be not larger than N_{tx}. For this standard, it is $N_{tx} = 5120$ octets. This allows for 100 sensors using the objects in this standard. An agent according to this definition shall be capable of receiving an APDU up to the size of at least N_{rx}. For this standard, it is $N_{rx} = 224$ octets.

For an independent living activity hub agent implementing multiple functions according to multiple device specializations, the maximum size of an APDU sent shall not be larger than $N_{tx,i}$, if this agent implements another device specialization i with $N_{tx,i} > N_{tx}$. Otherwise, the maximum size of an APDU sent shall not be larger than N_{tx}. An agent according to this definition shall be capable of receiving an APDU up to the size of at least $N_{rx,i}$ octets, if this agent implements another device specialization i with $N_{rx,i} > N_{tx}$. Otherwise, the agent shall be capable of receiving an APDU up to the size of at least N_{rx}.

In case the APDU size limit does not allow for the inclusion of a certain amount of multiple pending measurements at the agent, they shall be sent using multiple event reports. See 8.5.3 for the maximum number of measurements allowed for inclusion in a single event report.

8.3 Association procedure

8.3.1 General

Unless otherwise stated, the association procedure for an independent living activity hub agent and manager according to this standard shall be pursued as specified in IEEE Std 11073-20601.

8.3.2 Agent procedure—association request

In the association request sent by the agent to the manager:

— The version of the association procedure used by the agent shall be set to assoc-version1 (i.e., *assoc-version* = 0x80000000).

— The DataProtoList structure element of the data protocol identifier shall be set to data-proto-id-20601 (i.e., *data-proto-id* = 0x5079).

— The *data-proto-info* field shall contain a PhdAssociationInformation structure that shall contain the following parameter values:

1) The version of the data exchange protocol shall be set to protocol-version1 (i.e., *protocol-version* = 0x80000000).

2) At least the MDER shall be supported (i.e., *encoding-rules* = 0x8000).

3) The version of the nomenclature used shall be set to nom-version1 (i.e., *nomenclature-version* = 0x80000000).

4) The field *functional-units* may have the test association bits set but shall not have any other bits set.

5) The field *system-type* shall be set to sys-type-agent (i.e., *system-type* = 0x00800000).

6) The *system-id* field shall be set to the value of the System-Id attribute of the MDS object of the agent. The manager may use this field to determine the identity of the independent living activity hub with which it is associating and, optionally, to implement a simple access restriction policy.

7) The *dev-config-id* field shall be set to the value of the Dev-Configuration-Id attribute of the MDS object of the agent. The independent living activity hub agent does not support standard configurations. Only extended configuration IDs shall be specified here. The extended configurations range from 0x4000–0x7FFF.

8) If the agent supports only the independent living activity hub specialization, then the field indicating the data request modes (*data-req-mode-capab*) supported by the agent shall be set to the *data-req-supp-init-agent.*

9) If the agent supports only the independent living activity hub standard, then *data-req-init-manager-count* shall be set to zero, and *data-req-init-agent-count* shall be set to 1.

8.3.3 Manager procedure—association response

In the association response message sent by the manager:

— The *result* field shall be set to an appropriate response from those defined in IEEE Std 11073-20601. For example, if all other conditions of the association protocol are satisfied, accepted is returned when the manager recognizes the *dev-config-id* of the agent and accepted-unknown-config otherwise.

— In the DataProtoList structure element, the data protocol identifier shall be set to data-proto-id-20601 (i.e., *data-proto-id* = 0x5079).

— The *data-proto-info* field shall be filled in with a PhdAssociationInformation structure that shall contain the following parameter values:

1) The version of the data exchange protocol shall be set to protocol-version1 (i.e., *protocol-version* = 0x80000000).
2) The manager shall respond with a single selected encoding rule that is supported by both agent and manager. The manager shall support at least the MDER .
3) The version of the nomenclature used shall be set to nom-version1 (i.e., *nomenclature-version* = 0x80000000).
4) The field *functional-units* shall have all bits reset except for those relating to a test association.
5) The field *system-type* shall be set to sys-type-manager (i.e., *system-type* = 0x80000000).
6) The *system-id* field shall contain the unique system ID of the manager device, which shall be a valid EUI-64 type identifier.
7) The field *dev-config-id* shall be manager-config-response (0).
8) The field *data-req-mode-capab* shall be 0.
9) The fields *data-req-init-*-count* shall be 0.

8.4 Configuring procedure

8.4.1 General

The agent enters the Configuring state if it receives an association response of accepted-unknown-config. In this case, the configuration procedure as specified in IEEE Std 11073-20601 shall be followed. Subclause 8.4.2 specifies the configuration notification and response messages for an independent living activity hub agent with an example extended configuration ID 0x4000. Normally, a manager would not already know the extended configuration unless it had previously associated with this agent. For the purposes of this example, it does not.

8.4.2 Independent living activity hub—extended configuration

8.4.2.1 Agent procedure

The agent performs the configuration procedure using a “Remote Operation Invoke | Confirmed Event Report” message with an MDC_NOTI_CONFIG event to send its configuration to the manager (see IEEE Std 11073-20601). The ConfigReport structure is used for the *event-info* field (see Table 3). An example configuration notification message is shown in E.3.2.2 for an independent living activity hub agent with an extended configuration ID 0x4000.

8.4.2.2 Manager procedure

The manager shall respond to a configuration notification message using a “Remote Operation Response | Confirmed Event Report” data message with an MDC_NOTI_CONFIG event using the ConfigReportRsp structure for the *event-info* field (see Table 3). An example configuration notification response message corresponding to the configuration notification request message in 8.4.2.1 can be seen in E.3.2.3.

8.5 Operating procedure

8.5.1 General

Measurement data and status information are communicated from the independent living activity hub agent during the Operating state. If not stated otherwise, the operating procedure for an independent living activity hub agent of this standard shall be as specified in IEEE Std 11073-20601.

8.5.2 GET an independent living activity hub's MDS attributes

See Table 4 for a summary of the GET service.

If the manager leaves the attribute-id-list field in the roiv-cmip-get service message empty, the independent living activity hub agent shall respond with a rors-cmip-get service message, in which the attribute-list contains a list with the values of all implemented attributes of the MDS object.

If the manager requests specific MDS object attributes, indicated by the elements in attribute-id-list, and the agent supports this capability, then the independent living activity hub agent shall respond with a rors-cmip-get service message in which the attribute-list contains a list of the values of the requested attributes of the MDS object that are implemented. It is not required for an independent living activity hub agent to support this capability. If this capability is not implemented, the independent living activity hub agent shall respond with a "Remote Operation Error Result" service message (see IEEE Std 11073-20601) with the error-value field set to no-such-action (9).

8.5.3 Measurement data transmission

See Table 3 for a summary of the event report services available for measurement data transfer of activity data sensor events.

Measurement data transfer for an independent living activity hub agent of this standard shall always be initiated by the independent living activity hub (see agent-initiated measurement data transmission in IEEE Std 11073-20601). To limit the amount of data being transported within an APDU, the independent living activity hub agent shall not include more than 25 temporarily stored measurements in a single event report. If more than 25 pending measurements are available for transmission, they shall be sent using multiple event reports. If multiple measurements are available, up to 25 measurements should be transmitted within a single event report. Alternatively, they may be transmitted using a single event report for each measurement. However, the former strategy is recommended to reduce overall message size and power consumption.

An independent living activity hub agent always has an extended configuration and may use either fixed or variable format data update messages for transmitting measurement data.

8.6 Time synchronization

Time synchronization between an independent living activity hub agent and a manager may be used to coordinate the clocks used when reporting physiological events. Note that the mechanism for synchronizing an agent to a manager is outside the scope of this standard. If time synchronization is used, then this shall be reported in the Mds-Time-Info attribute of the MDS object.

9. Test associations

An independent living activity hub may implement a wide range of behaviors in a test association that enable a manufacturer to test features of a product in a comprehensive manner. It is also possible for an independent living activity hub not to support test associations at all. This clause defines a simple behavior that simulates the generation of a measurement in the context of an extended device configuration.

9.1 Behavior with standard configuration

As the independent living activity hub does not support a standard configuration, this standard does not define a test association that uses a standard configuration.

9.2 Behavior with extended configurations

To facilitate automated standardized test processes, an independent living activity hub that presents the extended configuration ID of 0x7FFF and enters into a test association should be able to simulate the arrival of measurement data from the device sensors. It should not be necessary for an operator to stimulate the sensors for the measurement data to be generated.

After the agent enters the Operating state, it simulates the reception of a sensor event from the sensor representing a temperature sensor event of high_temperature_detected. To the extent possible, this measurement is seen only by those components of the agent that understand the test association. When the event is propagated into an activity data enumeration object, the test-data bit of the measurement-status attribute shall be set if the measurement-status attribute is supported. An agent is not required to use the measurement-status attribute if it would not normally do so outside of a test association.

The agent should send the events reports for all simulated measures within 30 s of entering the operating state. The test association is terminated in a manner consistent with the agent's normal behavior for terminating an association.

10. Conformance

10.1 Applicability

This standard shall be used in conjunction with IEEE Std 11073-20601.

An implementation or a system can conform to the following elements of this standard:

- Domain information model class hierarchy and object definitions (object attributes, notifications, methods, and data type definitions)
- Nomenclature code values
- Protocol and service models
- Communication service model (association and configuration)

10.2 Conformance specification

This standard offers levels of conformance with respect to strict adherence to the standard device and the use of extensions for:

- Information model of a specific device
- Use of attributes, value ranges, and access methods

A vendor shall specify the level of conformance for an implementation based on this standard and provide details of the way in which the definitions of this standard and any extensions are applied.

Specifications shall be provided in the form of a set of implementation conformance statements (ICSs) as detailed in 10.4.

This standard is used in conjunction with IEEE Std 11073-20601; the ICSs should be created for this standard first. The ICS created for IEEE Std 11073-20601 may refer to the ICS for this standard where applicable.

10.3 Levels of conformance

10.3.1 General

This standard defines the following levels of conformance:

10.3.2 Conformance level 1: Base conformance

The application uses elements of the information, service, and communication models (object hierarchy, actions, event reports, and data type definitions) and the nomenclature scheme defined in IEEE Std 11073-20601 and IEEE 11073-104zz documents. All mandatory features defined in the object definition tables and in the ICS tables are implemented. Furthermore, any conditional, recommended, or optional features that are implemented shall follow the requirements in IEEE Std 11073-20601 and IEEE 11073-104zz documents.

10.3.3 Conformance level 2: Extended nomenclature (ASN.1 and/or ISO/IEEE 11073-10101)

Conformance level 2 meets conformance level 1 but also uses or adds extensions in at least one of the information, service, communication, or nomenclature models. These extensions shall conform to nomenclature codes from ASN.1 and/or within the ISO/IEEE 11073-10101 framework (0xF000 – 0xFFFF). These extensions should be defined in ICS tables pointing toward their reference.

10.4 ICSs

10.4.1 General format

The ICSs are provided as an overall conformance statement document that comprises a set of tables in the form given by the templates in the following clauses.

Each ICS table has the following columns:

Index	Feature	Reference	Req/Status	Support	Comment

The table column headings have the following meaning:

— Index: an identifier (e.g., a tag) of a specific feature.

— Feature: briefly describes the characteristic for which a conformance statement is being made.

— Reference: to the clause/paragraph within this document or an external source for the definition of the feature (may be empty).

— Req/Status: specifies the conformance requirement (e.g., mandatory or recommended)—in some cases, this standard does not specify conformance requirements but requests the status of a particular feature be provided.

- Support: specifies the presence or absence of a feature and any description of the characteristics of the feature in the implementation. This column is to be filled out by the implementer.
- Comment: contains any additional information on the feature. This column is to be filled out by the implementer.

Subclauses 10.4.2 to 10.4.6 specify the format of the specific ICS tables.

10.4.2 General ICS

The general ICS specifies the versions/revisions that are supported by the implementation and high-level system behavior.

Table 18 shows general ICSs.

Table 18 —IEEE 11073-10471 general ICSs' table

Index[a]	Feature	Reference	Req./Status	Support	Comment
GEN 11073-10471-1	Implementation Description	—	Identification of the device/application. Description of functionality.		
GEN 11073-10471-2	Standards followed and their revisions	(standard documents)	(set of existing revisions)	(set of supported revision)	
GEN 11073-10471-3	Nomenclature document used and revision	(standard documents)	(set of existing revisions)	(set of supported revisions)	
GEN 11073-10471-4	Conformance Adherence - Level 1 -	See 10.3.2	Base conformance declaration that device meets the following IEEE 11073-10471 conformance requirements: a) All mandatory requirements shall be implemented. b) If implemented, conditional, recommended, and optional requirements shall conform to standard.	Yes/No (No is not expected as No implies that the implementation is non-conformant)	
GEN 11073-10471-5	Conformance Adherence - Level 2 -	See 10.3.3	In addition to GEN 11073-10471-3, if the device implements extensions and/or additions, they shall conform to nomenclature codes from ASN.1 and/or 10101 framework. These extensions should also be defined in ICS tables pointing toward their reference.	Yes/No	
GEN 11073-10471-6	Object Containment Tree	See 6.3	Provide Object Containment Diagram showing relations between object instances used by the application. A conforming implementation uses only object relations as defined in the DIM.		
GEN 11073-10471-7	Nomenclature document used and revision	(standard documents)	(set of existing revisions)	(set of supported revision)	

Index[a]	Feature	Reference	Req./Status	Support	Comment
GEN 11073-10471-8	Data Structure Encoding	—	—	description of encoding method(s) for ASN.1 data structures	
GEN 11073-10471-9	Use of Private Objects	—	Does the implementation use objects that are not defined in the DIM?	Yes/No [If yes: explain in Table 19]	
GEN 11073-10471-10	Use of Private Nomenclature Extensions	—	Does the implementation use private extensions to the nomenclature (i.e., 0xF000-0xFFFF codes from ISO/IEEE 11073-10101)? Private nomenclature extensions are *only* allowed if the standard nomenclature does not include the specific terms required by the application.	Yes/No [If yes: explain in the Table 22]	
GEN 11073-10471-11	11073-20601 Conformance		Provide the conformance report required by IEEE Std 11073-20601.		

[a]The prefix GEN11073-10471- is used for the index in the general ICSs table.

10.4.3 DIM MOC implementation conformance statement

The DIM MOC ICS defines which objects are implemented. Information on each object shall be provided as a separate row in the template of Table 19.

Table 19—Template for DIM MOC ICS table

Index	Feature	Reference	Req./Status	Support	Comment
MOC-n	Object description	Reference to the clause in the standard or other location where the object is defined.	Implemented	Specify restrictions (e.g., maximum number of supported instances)	

The n in the Index column should be the object handle for implementations that have predefined objects. Otherwise, the Index column shall simply be a unique number (1..m).

All private objects shall be specified and include either a reference to the definition for the object or, where no publicly available reference is available, the definition of the object should be appended to the conformance statement.

The Support column should indicate any restrictions for the object implementation.

An object containment diagram (class instance diagram) should be provided as part of the DIM MOC ICS.

10.4.4 MOC attribute ICS

For each supported object as defined in the DIM MOC ICS, a MOC attribute ICS has to be provided that defines which attributes are used/supported by the implementation, including any inherited attributes. Table 20 is a template only.

Table 20 —Template for MOC attribute ICS table

Index	Feature	Reference	Req./Status	Support	Comment
ATTR-n-x	Attribute Name. Extended attributes shall include the attribute ID also.	Fill in the reference to the ASN.1 structure if the attribute is not defined in this standard.	M = Mandatory / C = Conditional / R = Recommended / O = Optional (as per definition in Attribute Definition Tables)	Implemented? Yes/No Static/Dynamic Specify restrictions (e.g., value ranges). Describe how attribute is accessed (e.g., Get, Set, sent in config event report or sent in a data event report). Describe any specific restrictions.	

All private attributes shall be specified and include reference to the definition for the attribute. Where no publicly available reference is available, the definition of the attribute should be appended to the conformance statement.

The Support column shall specify whether the attribute is implemented; for extension attributes, whether the attribute value is static or dynamic; any value ranges; restrictions on attribute access or availability; and any other information.

The n in the Index column refers to the ID of the managed object for which the table is supplied (i.e., the index of the managed object as specified in the MOC ICS). There is one separate table for each supported managed object.

The x in the Index column is a unique serial number (1..m).

NOTE—The attribute definition tables in the standard define a minimum mandatory set of attributes for each object.

10.4.5 MOC notification implementation conformance statement

The MOC notification ICS specifies all implemented notifications (typically in the form of the event report service) that are emitted by the agent. Table 21 provides a template for use. One table has to be provided for each object that supports special object notifications. One row of the table shall be used for each notification.

Table 21 —Template for MOC notification ICS table

Index	Feature	Reference	Req./Status	Support	Comment
NOTI-n-x	Notification Name and Notification ID	Reference to the clause in the standard or other location where the event is defined.		The Support column shall specify how the notification is sent and any restrictions.	

The n in the Index column refers to the ID of the managed object for which the table is supplied (i.e., the index of the managed object as specified in the POC ICS). There is one separate table for each managed object that supports specific object notifications (i.e., events).

The x in the Index column is a unique serial number (1..m).

All private notifications shall be specified and include reference to the definition for the notification. Where no publicly available reference is available, the definition of the notification should be appended to the conformance statement.

10.4.6 MOC nomenclature conformance statement

The MOC nomenclature ICS specifies all nonstandard nomenclature codes that are utilized by the agent. Table 22 provides a template for use. One row of the table is to be used for each nomenclature element.

Table 22—Template for MOC nomenclature ICS table

Index	Feature	Reference	Req./Status	Support	Comment
NOME-n	Nomenclature Name and Nomenclature value	Reference to the clause in the standard or other location where the nomenclature is defined or used.		Describe how the nomenclature is used. Describe any specific restrictions.	

The n in the Index column is a unique serial number (1..m).

Annex A

(informative)

Bibliography

[B1] IEEE 100™, *The Authoritative Dictionary of IEEE Standards Terms*, Seventh Edition. New York, Institute of Electrical Engineers, Inc.[5,6]

[B2] ISO/IEEE 11073-10101:2004, Health informatics—Point-of-care medical device communication—Part 10101: Nomenclature.[7]

[B3] ISO/IEEE 11073-10201:2004, Health informatics—Point-of-care medical device communication—Part 10201: Domain information model.

[B4] ISO/IEEE 11073-20101:2004, Health informatics—Point-of-care medical device communication—Part 20101: Application profile—Base standard.

[B5] ITU-T Rec. X.680-2002, Information technology—Abstract Syntax Notation One (ASN.1): Specification of basic notation.[8]

[5]IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org/).
[6]The IEEE standards or products referred to in this clause are trademarks of the Institute of Electrical and Electronics Engineers, Inc.
[7]ISO/IEEE publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse (http://www.iso.ch/). ISO/IEEE publications are also available in the United States from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org/).
[8]ITU publications are available from the International Telecommunications Union, Place des Nations, 1211 Geneva 20, Switzerland (http://www.itu.in/).

Annex B

(normative)

Any additional ASN.1 definitions

```
-- All unassigned "SensorHealthFlags" bit values are reserved for future expansion and
-- shall be reset.
-- Zero or more flags may be set.
SensorHealthFlags ::= BITS-32 {    -- this field is used in the activity data events
                                   -- to report sensor health
            auto-presence-received(16),  -- For sensors that have "heartbeat" operational status:
                                         -- indicates that the "heartbeat" has been seen and is ok.
                                         -- This flag may not be set if auto-presence-failed is set.
            auto-presence-failed(17),     -- For sensors that have "heartbeat" operational status:
                                         -- indicates that the "heartbeat" has not been seen as expected.
                                         -- This flag may not be set if auto-presence-received is set.
            low-battery(18),   -- Indicates the sensor is in the low battery condition.
                               -- This determination is unique to the sensor.
            fault(19),         -- Indicates that the sensor is in a fault condition and needs attention.
                                -- This determination is unique to the sensor.
            end-of-life(20)    -- Indicates that the sensor has reached end of life.
                               -- This indication is unique to the sensor.
}

 -- All unassigned "FallSensorFlags" bit values are reserved for future expansion and
-- shall be reset.
-- Only one flag can be set at a time.
FallSensorFlags::= BITS-32 {       -- this field is used in the fall sensor activity data events
            fall-detected(0)   -- indicates that a fall has been detected
}

 -- All unassigned "PersSensorFlags" bit values are reserved for future expansion and
-- shall be reset .
-- Only one flag can be set at a time.
PersSensorFlags::= BITS-32 {       -- this field is used in the PERS sensor activity data events
            pers-activated(0)  -- indicates that a PERS event has been detected
}

 -- All unassigned "EnvironmentalSensorFlags" bit values are reserved for future expansion and
-- shall be reset.
-- Only one flag can be set at a time.

EnvironmentalSensorFlags::= BITS-32 {-- this field is used in the environmental sensor activity data events
            condition-detected(0)  -- indicates that an environmental event has been detected
}

 -- All unassigned "MotionSensorFlags" bit values are reserved for future expansion and
-- shall be reset.
-- Only one motion flag can be set at a time.
-- The tamper flag may be additionally set at any time.
MotionSensorFlags::= BITS-32 {    -- this field is used in the motion sensor activity data events
            motion-detected(0),               -- indicates that a motion event has been detected
```

```
                motion-detected-delayed(1),     -- indicates that a motion with delay event has been
                                                -- detected
                tamper-detected(2)              -- indicates that a tamper event has been detected
}

 -- All unassigned "PropertyExitSensorFlags" bit values are reserved for future expansion and
-- shall be reset.
-- Only one flag can be set at a time.
PropertyExitSensorFlags::= BITS-32 {-- this field is used in the property exit sensor activity data events
                occupant-exit-property(0),  -- indicates that an occupant exit event has been detected
                exit-door-left-open(1)  -- indicates that an exit door left open event has been detected
}

 -- All unassigned "EnuresisSensorFlags" bit values are reserved for future expansion and
-- shall be reset.
-- Only one flag can be set at a time.
EnuresisSensorFlags::= BITS-32 {-- this field is used in the enuresis sensor activity data events
                enuresis-detected(0)  -- indicates that an enuresis event has been detected
}

 -- All unassigned "ContactClosureSensorFlags" bit values are reserved for future expansion and
-- shall be reset.
-- Only one flag can be set at a time.
ContactClosureSensorFlags::= BITS-32 {-- this field is used in the door use sensor activity data events
                contact-opened(0),  -- indicates that a door open event has been detected
                contact-closed(1)  -- indicates that a door close event has been detected
}

 -- All unassigned "UsageSensorFlags" bit values are reserved for future expansion and
-- shall be reset.
-- Only one flag can be set at a time.
UsageSensorFlags::= BITS-32 {-- this field is used in the usage sensor (bed/chair) activity data events
                usage-started(0),  -- indicates that a usage started event has been detected
                usage-ended(1),  -- indicates that a usage ended event has been detected
                expected-use-start-violation(2),  -- indicates that an expected use start violation event
                                  -- has been detected
                expected-use-stop-violation(3),  -- indicates that an expected use stop violation event
                                  -- has been detected
                absence-violation(4)  -- indicates that an absence violation event has been detected
}

 -- All unassigned "SwitchSensorFlags" bit values are reserved for future expansion and
-- shall be reset.
-- Only one flag can be set at a time.
SwitchSensorFlags::= BITS-32 {    -- this field is used in the switch sensor activity data events
                switch-on(0),  -- indicates that a switch on event has been detected
                switch-off(1)  -- indicates that a switch off event has been detected
}

 -- All unassigned "MedDosageSensorFlags" bit values are reserved for future expansion and
-- shall be reset.
-- Only one flag can be set at a time.
MedDosageSensorFlags::= BITS-32 {-- this field is used in the dosage sensor activity data events
```

```
                          dosage-taken(0),   -- indicates the med dosage was taken
                          dosage-missed(1)  -- indicates the med dosage was not taken when expected
}

 -- All unassigned "TemperatureSensorFlags" bit values are reserved for future expansion and
-- shall be reset.
-- Only one flag can be set at a time.
TemperatureSensorFlags::= BITS-32 {-- this field is used in the temperature sensor activity data events
                          high-temperature-detected(0), -- indicates that a high temperature event has been detected
                          low-temperature-detected(1),  -- indicates that a low temperature event has been detected
                          rate-of-change-too-fast(2)  -- indicates that a rate of change too fast event
                                                     -- has been detected
}
```

Annex C

(normative)

Allocation of identifiers

This annex contains the nomenclature codes used in this document and not found in IEEE Std 11073-20601. For those not contained in this annex, the normative definition is found in IEEE Std 11073-20601.

The format used here follows that of ISO/IEEE 11073-10101.

```
/*****************************************************************************************
* All of the following are from NomPartition (MDC_PART_PHD_AI)                           *
*****************************************************************************************/
#define MDC_AI_TYPE_SENSOR_FALL                    1    /*                             */
#define MDC_AI_TYPE_SENSOR_PERS                    2    /*                             */
#define MDC_AI_TYPE_SENSOR_SMOKE                   3    /*                             */
#define MDC_AI_TYPE_SENSOR_CO                      4    /*                             */
#define MDC_AI_TYPE_SENSOR_WATER                   5   /*                              */
#define MDC_AI_TYPE_SENSOR_GAS                     6   /*                              */
#define MDC_AI_TYPE_SENSOR_MOTION                  7    /*                             */
#define MDC_AI_TYPE_SENSOR_PROPEXIT                8    /*                             */
#define MDC_AI_TYPE_SENSOR_ENURESIS                9    /*                             */
#define MDC_AI_TYPE_SENSOR_CONTACTCLOSURE          10    /*                            */
#define MDC_AI_TYPE_SENSOR_USAGE                   11   /*                             */
#define MDC_AI_TYPE_SENSOR_SWITCH                  12   /*                             */
#define MDC_AI_TYPE_SENSOR_DOSAGE                  13   /*                             */
#define MDC_AI_TYPE_SENSOR_TEMP                    14    /*                            */
/* The range 15-1024 is reserved for future expansion                                  */
/*****************************************************************************************
* All of the following are from NomPartition – Partition: MDC_PART_PHD_AI
*
*****************************************************************************************/
/*****************************************************************************************
* Locations are encoded within 16 bits as…..                                             *
* upper 10 bits are the location type (for example bedroom - MDC_AI_LOCATION_BEDROOM)    *
* lower 6 bits are the location type instance identifier (for example bedroom 0, bedroom 1, etc.)  *
* therefore, location viewed as a 16 bit entity would have the types assigned in blocks of  64   *
* The location type instance identifier allows for both a single dwelling with multiple rooms of that  *
* type, as well as a multiple family dwelling that may contain that type room in each subunit    *
* ****************************************************************************************/
/* general                                                                               */
#define MDC_AI_LOCATION_START                 1024  /*                                   */
#define MDC_AI_LOCATION_UNKNOWN               1024 /*                                    */
#define MDC_AI_LOCATION_UNSPECIFIED           1088  /*                                   */
#define MDC_AI_LOCATION_RESIDENT              1152  /*                                   */
#define MDC_AI_LOCATION_LOCALUNIT             1216  /*                                   */
/* The range 1217-3071 is reserved for future expansion                                  */
/* rooms                                                                                 */
#define MDC_AI_LOCATION_BEDROOM               3072  /*                                   */
#define MDC_AI_LOCATION_BEDROOMMASTER         3136  /*                                   */
#define MDC_AI_LOCATION_TOILET                3200  /*                                   */
```