INTERNATIONAL STANDARD

ISO/IEC 30128

First edition
2014-11-15

Information technology — Sensor networks — Generic Sensor Network Application Interface

Technologies de l'information — Réseaux de capteurs — Interface générique pour des applications de réseaux de capteurs

Reference number
ISO/IEC 30128:2014(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

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

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

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword — Supplementary information.

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*.

Introduction

Sensor network is a key technology to enable building context aware smart environments for human beings, monitoring health status, cross-reality services, etc. But there are many different sensor network implementations and they are not interoperable. In general, sensor networks are developed according to the sensor network applications' requirements (in which ways sensor network applications use sensor networks) within sensor network hardware constraints.

When it comes to sensor network applications' requirements, they include transport-level requirements, sensor networks' hardware point of requirements, applications' operational requirements, etc. Of these requirements, applications' operational requirements affect sensor network implementations, even though each sensor network supports the same transport protocol and uses the same hardware specification. However, these applications' operational requirements can be generalized and can be used to derive standard application layer interfaces between sensor networks and sensor network service providers.

This International Standard specifies generic sensor network application interfaces based on the generalized sensor network applications' operational requirements with consideration on sensor network hardware constraints.

Information technology — Sensor networks — Generic Sensor Network Application Interface

1 Scope

This International Standard specifies the interfaces between the application layers of service providers and sensor network gateways, which is Protocol A in interface 3, defined in ISO/IEC 29182-5.

This International Standard covers

— description of generic sensor network applications' operational requirements,

— description of sensor network capabilities, and

— mandatory and optional interfaces between the application layers of service providers and sensor network gateways

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 29182-2:2013, *Information technology — Sensor networks: Sensor Network Reference Architecture (SNRA) — Part 2: Vocabulary and terminology*

ISO/IEC 29182-5:2013, *Information technology — Sensor networks: Sensor Network Reference Architecture (SNRA) — Part 5: Interface definitions*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1
actuator
device that provides a physical output in response to an input signal in a predetermined way

[SOURCE: ISO/IEC 29182-2:2013, 2.1.1]

3.2
application layer
layer that provides means for the application processes to access the OSI environment

[SOURCE: ISO/IEC 29182-2:2013, 2.3.1.1]

3.3
authentication
act of verifying the claimed identity of an entity

[SOURCE: ISO/IEC 29182-2:2013, 2.6.1]

Note 1 to entry: Entity may include sensor, actuator, or sensor network element.

3.4
authorization
granting of rights, which includes the granting of access based on access rights

[SOURCE: ISO/IEC 29182-2:2013, 2.6.2]

3.5
continuous mode
sensor data query mode with duration value (*d*) and interval value (*t*)

3.6
event mode
sensor data query mode with event conditions. Sensor networks keep collecting sensor data and send them only, if the conditions are met

3.7
identification
process of recognizing an entity by using its attributes, identifier, etc.

[SOURCE: ISO/IEC 29182-2:2013, 2.7.2]

3.8
instant mode
sensor data query mode for an immediate one- time response from a sensor network

3.9
onTime mode
sensor data query mode with an action time

3.10
personal area network
network consisting of sensor nodes, communication devices, or networked peripheral devices all in the vicinity of a person

[SOURCE: ISO/IEC 29182-2:2013, 2.1.4]

3.11
PAN coordinator
device which is responsible for formation and maintenance of a PAN

3.12
pull mode
sensor data delivery mode with an explicit sensor data query

3.13
push mode
sensor data delivery mode without any explicit sensor data query

Note 1 to entry: Push mode sensor data delivery may be triggered by an explicit start request and may be terminated by an explicit stop request. It is an implementation issue.

3.14
sensor
device that observes and measures a physical property of a natural phenomenon or man-made process and converts that measurement into a signal

[SOURCE: ISO/IEC 29182-2:2013, 2.1.5]

Note 1 to entry: Signal can be electrical, chemical, etc.

3.15
sensor network
system of spatially distributed sensor nodes interacting with each other and, depending on applications, possibly with other infrastructure in order to acquire, process, transfer, and provide information extracted from its environment with a primary function of information gathering and possible control capability

[SOURCE: ISO/IEC 29182-2:2013, 2.1.6]

Note 1 to entry: Distinguishing features of a sensor network can include wide area coverage, use of radio networks, flexibility of purpose, self-organization, openness, and providing data for multiple applications.

3.16
sensor network application
use case of sensor networks, which provides a set of functions to users to meet defined requirements

[SOURCE: ISO/IEC 29182-2:2013, 2.2.2]

EXAMPLE Monitoring forests to detect natural fires; monitoring seismic activity; monitoring pollution levels in environment.

3.17
sensor network client
application software that uses information provided by a sensor network

3.18
sensor network gateway
sensor network element that connects a sensor network to another network with different architectures or protocols, permitting information exchange between them

[SOURCE: ISO/IEC 29182-2:2013, 2.1.7]

Note 1 to entry: Sensor network gateway functionalities may include address or protocol translation.

3.19
sensor network resource
entity related to a sensor network which may be a sensor network gateway, a PAN coordinator (if any), a sensor node or a transducer

3.20
sensor network service
set of functionalities offered by individual sensor network elements or the sensor network

[SOURCE: ISO/IEC 29182-2:2013, 2.2.3]

EXAMPLE Generating an alarm signal if the measurement made at a sensor exceeds or drops out of certain prescribed range; providing average sensor measurements over a given geographic area.

3.21
transducer
device converting energy from one domain into another, calibrated to minimize the errors in the conversion process. It could be a sensor or an actuator

[SOURCE: IEEE Std 1451.1-1999]

3.22
user
any person, organization, process, device, program or system which uses services provided by others, and may benefit from the operation of a sensor network

[SOURCE: ISO/IEC 29182-2:2013, 2.8.5]

4 Symbols and abbreviated terms

This document uses the following abbreviations and acronyms:

CoAP	Constrained Application Protocol
DNS	Domain Name System
O&M	Observations and Measurements in SWE
PAN	Personal Area Network
SensorML	Sensor Model Language in SWE
SNC	Sensor Network Client
SWE	Sensor Web Enablement
TEDS	Transducer Electronic Data Sheet

5 Conventions

In this part of ISO/IEC 30128:

The keyword “shall” is used to indicate requirements strictly to be followed in order to conform to the document and from which no deviation is permitted.

The keyword “should” is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others, or that a certain course of action is preferred but not necessarily required, or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.

The keyword “may” is used to indicate a course of action permissible within the limits of the document.

The keyword “can” is used for statements of possibility and capability, whether material, physical or causal.

6 Overview of sensor network applications

6.1 Communication model

Sensor network applications provide sensor-associated or actuator-associated services for users by interacting with at least one sensor network. When it comes to sensor network service providers, they need to communicate with lots of different types of sensor networks to provide integrated services for users. On the other hand, sensor networks provide sensor network services to multiple sensor network service providers to maximize their benefits. In this context, the communication model between sensor network service providers and sensor networks can be summarized in two cases.

In one case, an interface is defined by a sensor network that provides sensor network services based on the interface. Then, sensor network service providers implement the interface defined by each sensor network to communicate with the sensor network. This case is illustrated in Figure 1. If there is no standard interface between sensor networks and sensor network service providers, sensor network service providers need to implement different interfaces to communicate with different sensor networks. In Figure 1, the sensor network service provider needs to implement interface A, interface B, and interface C altogether to provide integrated services for users. In view of that there are many different sensor network providers in the world, implementing all of these sensor network providers’ interfaces is a tedious and time consuming process, and it is definitely not a good solution.

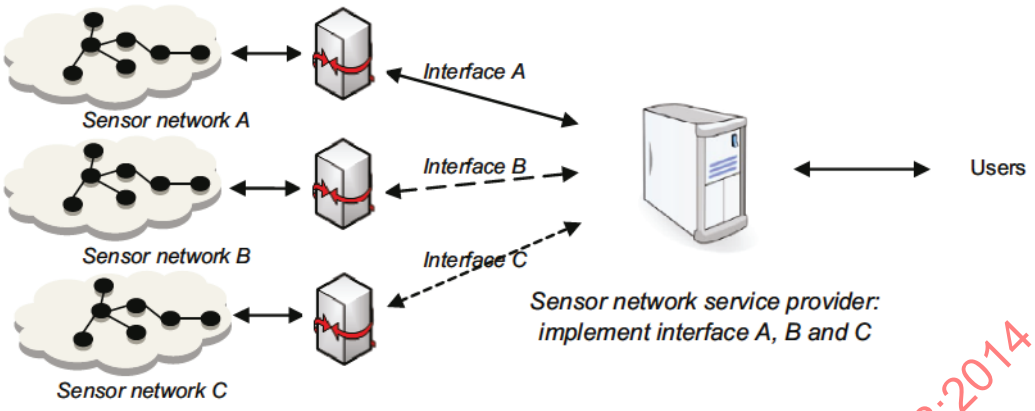

Figure 1 — Communication model: Multiple sensor networks and one sensor network service provider

In the other case, each sensor network service provider defines its own interface. Thus, sensor networks need to implement each interface to communicate with the sensor network service provider. This case is illustrated in Figure 2. If there are multiple sensor network service providers with different interfaces, then sensor networks need to implement each interface provided by each sensor network service provider. From a sensor network providers' point of view, they need to develop a single sensor network that can implement all interfaces or they need to develop different sensor networks for different sensor network service providers to provide the same sensor network services. Both situations are not reasonable from the sensor network providers' perspective, because they increase sensor network development cost and time. As a consequence, this impedes the proliferation of sensor network usage.

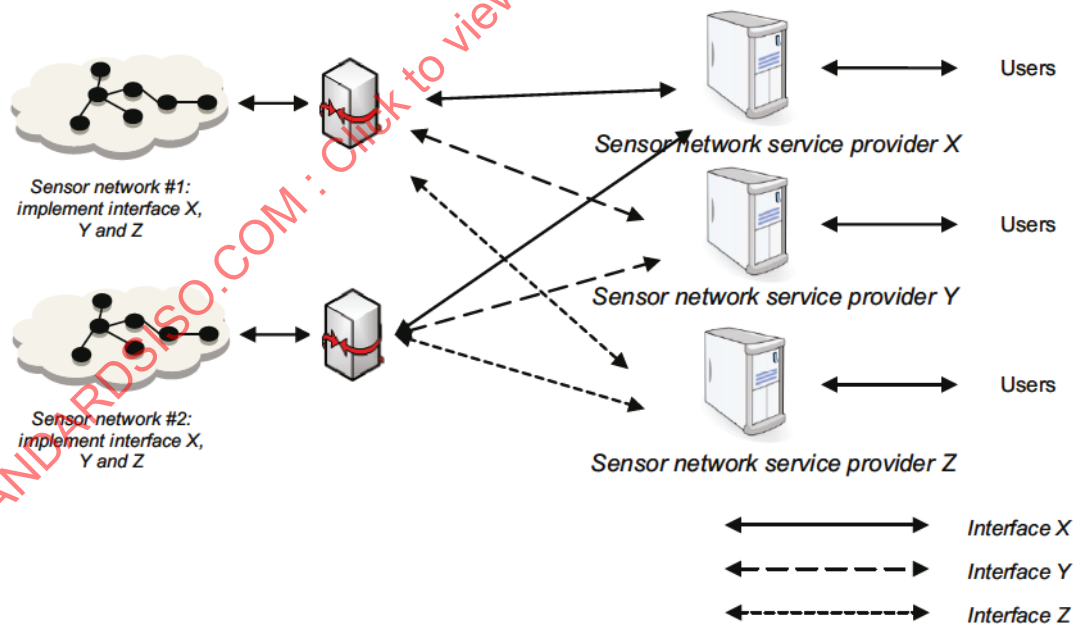

Figure 2 — Communication model: One sensor network and multiple sensor network service providers

In this regard, standard interface specification between sensor networks and sensor network service providers is very important. Moreover, the standard interface enables sensor network implementation free from specific sensor network applications' requirements, so it derives cost-effective mass production of sensor networks.

From this standard's point of view, the role of sensor network gateways is very important. A sensor network gateway is a front-end of a sensor network, in respect to sensor network service providers. Therefore, if a standard interface between sensor networks and sensor network service providers is defined, the interface will reside between the sensor network gateway and the sensor network providers. In a case where a smart-device plays the role of a standalone sensor node without a separate sensor network gateway, then the smart-device may be regarded as a sensor network gateway and a sensor node at the same time.

Basically, a sensor network gateway translates sensor network service providers' messages into sensor networks' messages (e.g., ZigBee messages, Bluetooth messages, CoAP message, etc.) and vice versa. But in some cases, a sensor network gateway also needs to handle time synchronization to deliver sensor network clients' message to sensor nodes. That is because when a sensor network service provider sends a message to sensor nodes, the sensor nodes may be in a sleep mode for power saving. In this case, a sensor network gateway should know the life cycle of each sensor node in order to keep-and-forward messages to sensor nodes. Or, when a sensor node or a transducer registers itself to a sensor network service provider, a sensor network gateway may need to attach more information on the registration message from a sensor node to include hierarchical topology information. In general, if a gateway is implemented on a high performance machine, the gateway can perform complex operations on behalf of capability-constrained sensor nodes. In this regard, in some cases, a sensor network's computing capabilities means the sensor network gateways' capabilities. In summary, sensor network gateways play a very important role in sensor networks and they are different from other domain's gateways.

6.2 Sensor network client operations

6.2.1 Overview of sensor network client operations

There are various kinds of sensor network applications: logistics and supply chain management application, energy and utility distribution application, industrial production automation monitoring and controlling application, healthcare, medical application, etc.

These sensor network applications generally consume sensor data collected by sensor networks and some applications control appropriate actuators to handle observed situation. Other sensor network applications monitor the status of sensor networks and if necessary, they control the sensor networks corresponding to observed sensor networks' status. In this international standard, sensor network application operations are classified into three categories: sensor manipulation, actuator manipulation, and sensor network monitoring and controlling.

Regarding sensor manipulation, there are several ways to collect sensor data. Roughly, sensor network manipulation mode is classified into push mode and pull mode. And pull mode is further classified into instant mode, continuous mode, event mode, and onTime mode.

Regarding actuator manipulation, there may be various kinds of actuators: on/off actuator, text/image displaying actuator, multi-functional complex actuator, etc. All these actuators manipulation is performed by issuing action requests with appropriate parameters. After processing actuators, sensor networks shall respond with an appropriate action status.

Regarding sensor network monitoring and controlling, a sensor network client needs to know the current status of sensor networks and needs to control sensor networks according to the sensor network client's policy. In case of sensor network controlling, it may be triggered irrespective of monitoring process, or may follow after monitoring process. A sensor network client may issue a sensor network control request based on administrative purposes, or may issue a sensor network control request to manipulate monitoring results. Sensor network controlling may include several operations, such as reset, shutdown, and reconfiguration.

In this international standard, a sensor network client (SNC) is referred to as an application software residing in a sensor network service provider.

6.2.2 Sensor manipulation

6.2.2.1 Overview of sensor manipulation

There are various ways to collect sensor data from sensor networks. From the sensor networks' point of view, it provides collected sensor data in two ways. One way is pushing collected sensor data without any consideration on specific requests from sensor network clients. It is called a push mode. The other way is collecting sensor data and delivering sensor data according to specific requests from sensor network clients. This is called a pull mode. The pull mode is further classified into instant mode, continuous mode, and event mode based on how to request sensor data from sensor network clients' point of view. Sensor network clients may request sensor data only one time (instant mode) or may request sensor data repetitively (continuous mode). In some cases, sensor network clients may request sensor data only some events occur (event mode). In some other cases, sensor network clients may request these pull mode sensor data manipulations with an exact action time when to collect and deliver sensor data. It is called an onTime mode.

6.2.2.2 Push mode

Figure 3 depicts a push mode sensor manipulation.

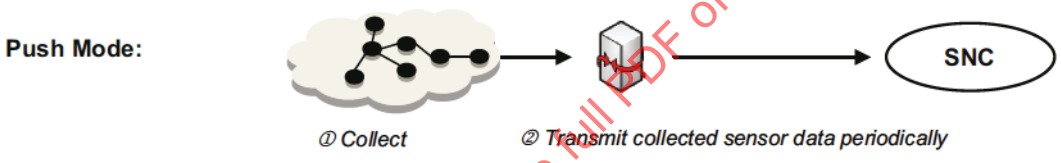

Figure 3 — Sensor data collection: Push mode

In case of a push mode, a sensor network collects and transmits sensor data based on its configuration on collecting and transmitting. From an implementation point of view, a push mode sensor manipulation may be implemented in two ways. The simplest way is in which a sensor network just pushes sensor data when it is connected to a sensor network client. Or, it may be implemented in a way that the sensor network starts pushing sensor data when it receives a "start" command and stops pushing when it receives a "stop" command from a sensor network client. In this case, the "start" and "stop" commands are issued by a sensor network client, but it doesn't change the sensor data collection and transmit configuration of the sensor network itself.

A push mode is used mainly by simple types of sensor network monitoring applications. In this case, the expected computing complexity of sensor networks is very low. Sensor networks just push collected sensor data to a waiting sensor network client and data processing is up to the sensor network client.

6.2.2.3 Pull mode: instant mode

Figure 4 depicts an instant mode of a pull mode sensor manipulation.

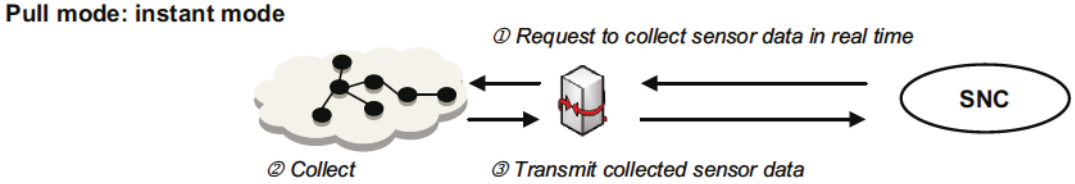

Figure 4 — Sensor data collection: Pull mode-instant mode

In case of an instant mode, a sensor network client sends a sensor data collection request to a sensor network in order to collect sensor data in real time. On receipt of the request, the sensor network starts collecting sensor data from sensor nodes. After the sensor data collection ends, the sensor network transmits collected sensor data to the requesting sensor network client. Sensing and transmitting are performed once in real-time.

An instant mode is a very basic pull mode operation. Whenever a sensor network client needs to get sensor data, it issues an instant sensor data request to a target sensor network. An instant sensor data request may include conditions. For example, a sensor network client wants to know the temperature value of place#1 only if the temperature value is over 30 °C. Then, the sensor network client may send an instant query with the condition to the sensor network. Then the sensor network responds with a temperature value only if the temperature value is over 30 °C. If the request includes conditions, the sensor network needs to check the conditions before sending the collected sensor data to the sensor network client. Condition checking capability is an additional feature of the sensor networks. Therefore, a sensor network shall inform sensor network clients whether it has condition checking capability or not beforehand.

In comparison to a push mode, expected computing complexity of sensor networks to perform instant mode operation is higher. A sensor network shall at the minimum process instant sensor data requests and respond.

6.2.2.4 Pull mode: continuous mode

Figure 5 depicts a continuous mode of a pull mode sensor manipulation.

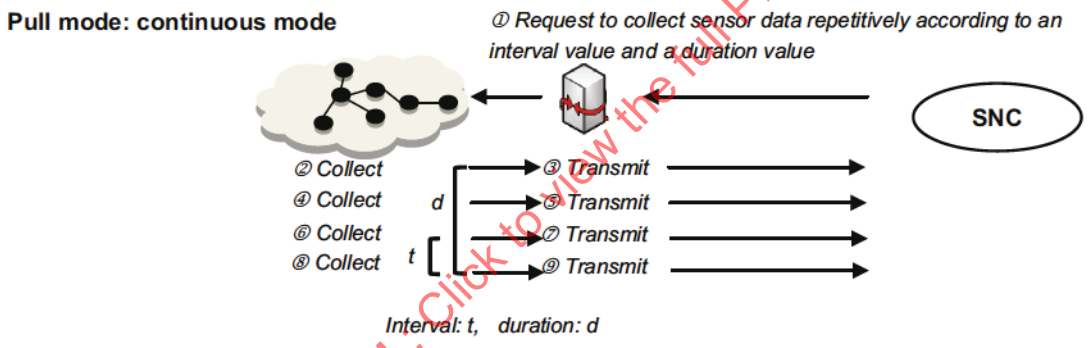

Figure 5 — Sensor data collection: Pull mode-continuous mode

In case of a continuous mode, a sensor network client sends a sensor data request to a sensor network in order to collect sensor data. On receipt of the request, the sensor network collects sensor data at every interval value (t) during duration value (d), and transmits collected sensor data to the requesting sensor network client repetitively.

A continuous mode is used by sensor network applications which need sensor data with a specific interval value and a specific duration value. A continuous sensor data request may include conditions. As in instant mode, sensor networks with condition checking capability are needed for continuous mode operations. Therefore, a sensor network shall inform sensor network clients whether it supports condition checking capability or not beforehand. In addition, if there is a need to stop a continuous mode operation, the sensor network client should send a stop processing request to the sensor network. Stop processing is an additional feature of a sensor network so it shall be known to the sensor network client beforehand.

In comparison to an instant mode, a continuous mode operation requires higher sensor networks' computing complexity.

6.2.2.5 Pull mode: event mode

Figure 6 depicts an event mode of a pull mode sensor manipulation.

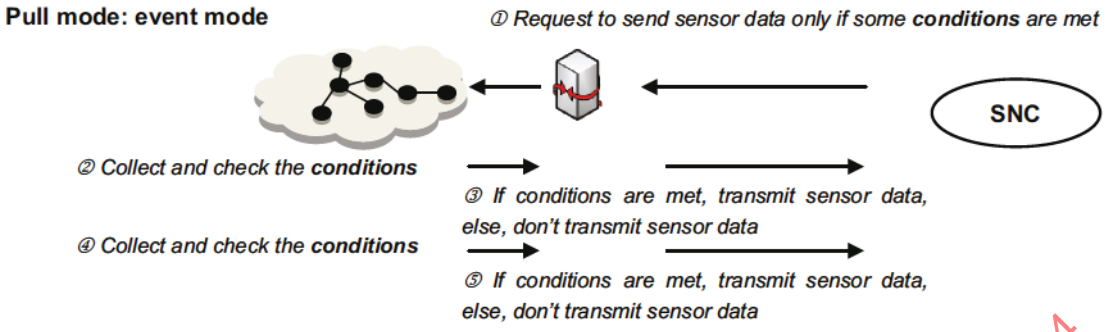

Figure 6 — Sensor data collection: Pull mode-event mode

In the case of an event mode, a sensor network client sends a sensor data collection request to a sensor network in order to collect sensor data and to send back the collected sensor data only if specified conditions are met. The conditions are given by a sensor network client. The differences between an event mode and other pull modes with conditions are that conditions in an event mode are mandatory. And an event mode sensor data requests are processed by the sensor network continuously until the sensor network client requests to stop processing explicitly. The sensor network keeps collecting sensor data based on its collecting configuration like in a push mode and checks the conditions which are specified by a sensor network client. Only if the conditions are met, the sensor network sends collected sensor data to a requesting sensor network client. For example, a sensor network client needs temperature data from sensor network#1 only when observed temperature values are higher than certain threshold to turn on a cooling system. Then, a sensor network client may request "send temperature data collected from sensor network#1 only if the temperature value is higher than 30 °C.

An event mode may be used by disaster monitoring applications effectively. Disaster monitoring applications need to check abnormal conditions from monitoring areas. By using event mode sensor data requests, disaster monitoring applications reduce unnecessary traffic between sensor networks and sensor network clients. It results in effective application processing and extending the sensor network's battery life by reducing unnecessary signal transmission.

6.2.2.6 Pull mode: onTime mode

Figure 7 depicts an onTime mode of a pull mode sensor manipulation.

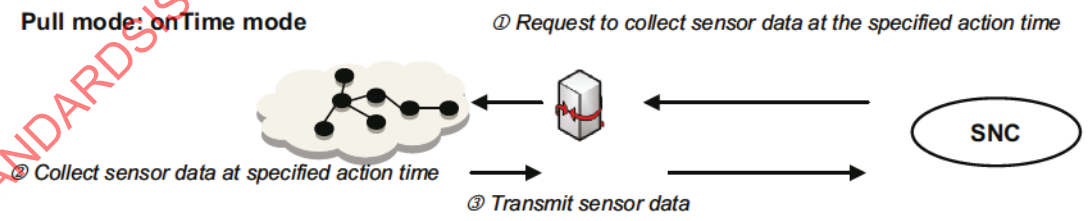

Figure 7 — Sensor data collection: Pull mode-onTime mode

In case of an onTime mode, a sensor network client sends a sensor data collection request to a sensor network to collect sensor data at a specific time (i.e. action time) and the sensor network thus collects sensor data at the specified time. An onTime mode is used by mixing with other modes: an instant mode sensor data request, continuous mode sensor data request, event mode sensor data request, or actuator manipulation. For example, if an instant mode sensor data request includes an action time field, then the sensor network shall process the instant mode sensor request at the specified action time. Therefore, in an onTime mode processing, time synchronization between sensor network clients and sensor networks is very crucial.

In comparison to other pull mode operations, an onTime mode operation requires a higher computing complexity. At least, a sensor network shall provide action time processing complexity and other pull mode operation processing complexity.

6.2.3 Actuator manipulation

Figure 8 depicts an actuation manipulation.

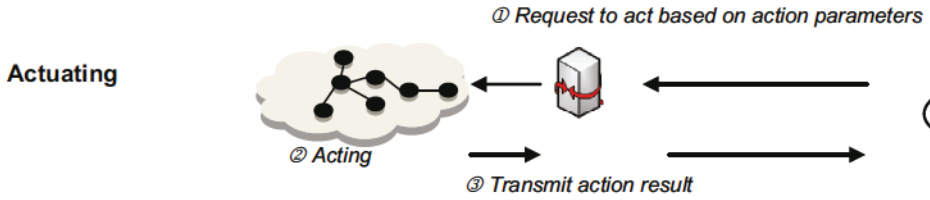

Figure 8 — Actuator manipulation

A sensor network client manipulates an actuator by specifying action parameters. Then the actuator responds with an action result to the requesting sensor network client. A sensor network client shall specify target actuator identifiers, action type, and action parameters and may specify an action time optionally.

Action time processing is an additional feature of sensor networks. Therefore, sensor networks shall inform sensor network clients whether they support action time processing capability or not beforehand.

Regarding complexity of actuator manipulation, it is proportional to the complexity of action parameters. Because of diversity of actuator types and actuation parameters, detailed specification of the action types and action parameters is not addressed in this international standard.

6.2.4 Sensor network monitoring and controlling

Figure 9 depicts sensor network monitoring and controlling.

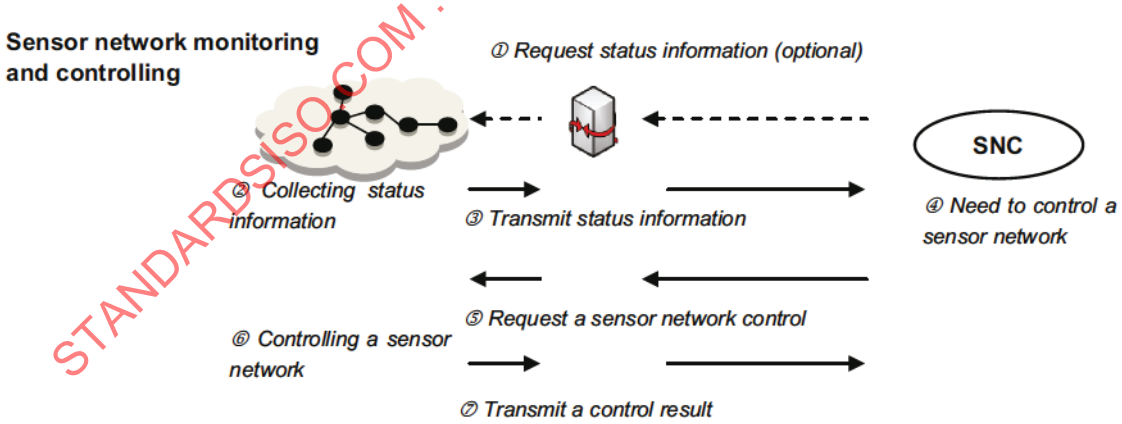

Figure 9 — Sensor network monitoring and controlling

Some sensor network clients may need sensor network status information, such as topology information, remaining battery life of each sensor node, and accessibility information of each sensor node. This information is collected from sensor networks by a monitoring process. Sensor network monitoring is implemented in two ways. Some sensor networks collect and send status information based on their configuration autonomously. Others collect and send status information only if there is a request from a certain sensor network client. A sensor network shall advertise its monitoring mode by registration

procedure beforehand. And sensor network clients shall collect status information based on the advertised monitoring mode of sensor networks.

Some sensor network clients may need to control sensor networks by issuing reset, reconfiguration, etc. These control functionalities are implemented by sensor networks selectively based on the sensor networks' computing constraints and sensor network provider's policies.

7 Overview of sensor network capabilities

Sensor networks have different computing capabilities. Currently many sensor networks support only push mode sensor data processing. However, for some specialized sensor network services, such as military services or medical services, powerful sensor nodes are required. As device technology and communication technology evolve, small and smart devices with high-speed computing and communication capabilities and increasing memory size emerge in the marketplace. This means the spectrum of sensor network capability is diverse and wide. If possible, exploiting sensor networks' computing capabilities is reasonable for sensor network clients because it can reduce sensor network clients' workload and unnecessary traffic cost.

When it comes to sensor network capabilities, some sensor networks are very simple and can only support push mode sensor data requests. Some others support complex computing, such as instant mode sensor data requests. And others support more complex computing, such as continuous mode sensor data requests and event mode sensor data requests. Or, some sensor network even can handle multiple sensor data requests at the same time. These kinds of sensor networks' capabilities are decided by sensor network providers' constraints and policies, sensor network clients' requirements and policies, sensor network hardware and operating system's constraints, etc. The therefore, it is very important that sensor networks shall advertise sensor networks' capabilities beforehand. Based on advertised information, sensor network clients shall exploit sensor networks effectively by utilizing optimal functions among provided capabilities. These sensor networks' capabilities information shall be included in sensor network metadata. Sensor network metadata is metadata of sensor networks and it shall include sensor network's identification information, capabilities information, etc. The sensor network metadata is delivered to the sensor network clients by registration procedures.

Sensor network capabilities may be provided by each transducer, each sensor node or can be provided by sensor network gateways. This is an implementation issue.

8 Security considerations

When it comes to sensor data, security consideration is always of high importance. This is to do with the nature of sensor data. Sensor data reveals the current real status of specific environment, personal health, specific factories, specific plant, etc. Therefore, owners of the data need appropriate security measures (authentication for the users and secure exchange of the data) to provide sensor data to the public. On the other hand, the users of sensor data require appropriate levels of credibility on sensor data providers and sensor data. Users should convince themselves whether the users can trust sensor network providers and reliability of sensor data or not. Therefore, sensor data providers should also provide a proof of their authenticity to sensor data users.

In this regard, appropriate authentication and confidentiality preservation measures shall be taken into consideration when the interface between sensor networks and sensor network clients is designed. Optionally, a sensor network may require appropriate level of authorization measures to provide different level of services to different sensor network clients.

Detailed specification on concrete security measures is beyond of this international standard.

9 Data model of sensor data and metadata

This international standard does not specify a concrete data model for sensor data and sensor network metadata. When it comes to sensor data, SWE O&M may be applied or specification in 10.2.3 may be

applied. Regarding sensor network metadata, SWE SensorML, and the TEDSs defined in ISO/IEC/IEEE 21450, ISO/IEC/IEEE 21451-4, and ISO/IEC/IEEE 21451-7 may be applied or Annex A may be applied.

Note Concrete location schema can be found in ISO 19123

Note Concrete metadata for geographic information can be found in ISO 19115

10 Generic sensor network application interface specification

10.1 Overview of generic sensor network application interface

10.1.1 General

Generic sensor network application interface consists of a mandatory interface and an optional interface. The mandatory interface is the interface which shall be implemented by sensor networks and sensor network clients. It includes registration interface, deregistration interface, and sensing data report interface. Figure 10 depicts mandatory interfaces. In this international standard, each interface is expressed as a message.

Usually, sensor manipulation and actuator manipulation require some extent of processing time within sensor networks. Therefore, it is reasonable that those kinds of operations are handled asynchronously. But it is an implementation issue.

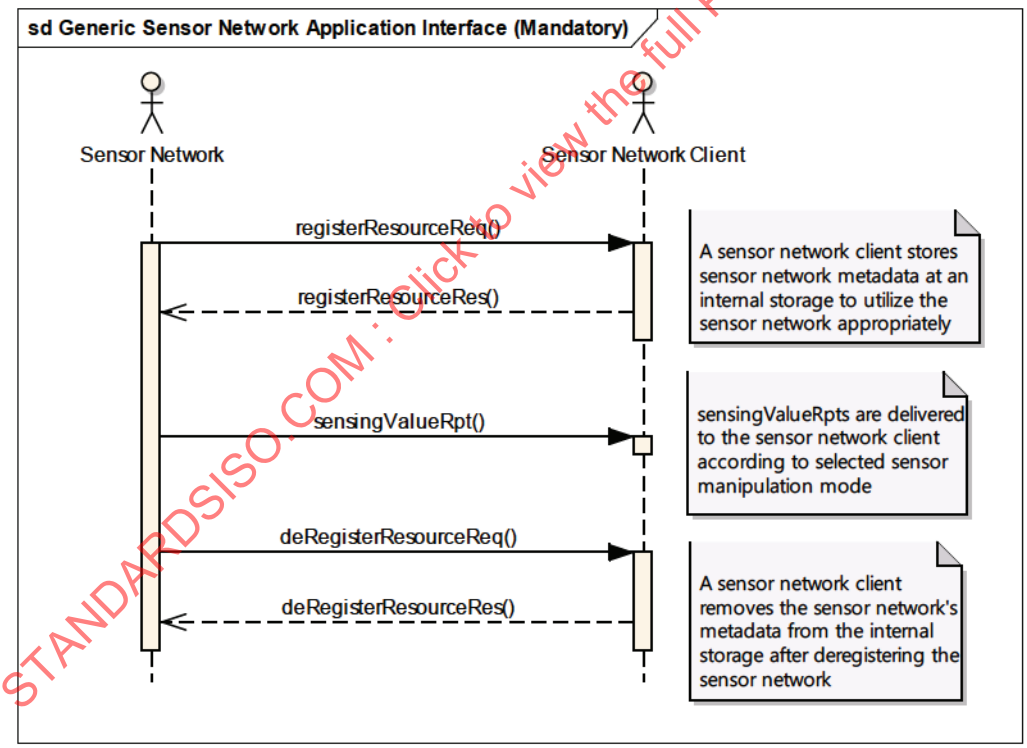

Figure 10 — Mandatory sensor network application interface flow

Meanwhile optional interface may be implemented by sensor networks and sensor network clients. Based on the constraints and policies on sensor networks, optional interface may be implemented selectively. Figure 11 depicts optional interfaces. Pull mode sensor manipulation interface, actuator manipulation interface, sensor network monitoring and controlling interface, scalability interface, and message handling interface are classified as optional interface. If optional interface is implemented, then implementation shall be in conformance with this international standard.

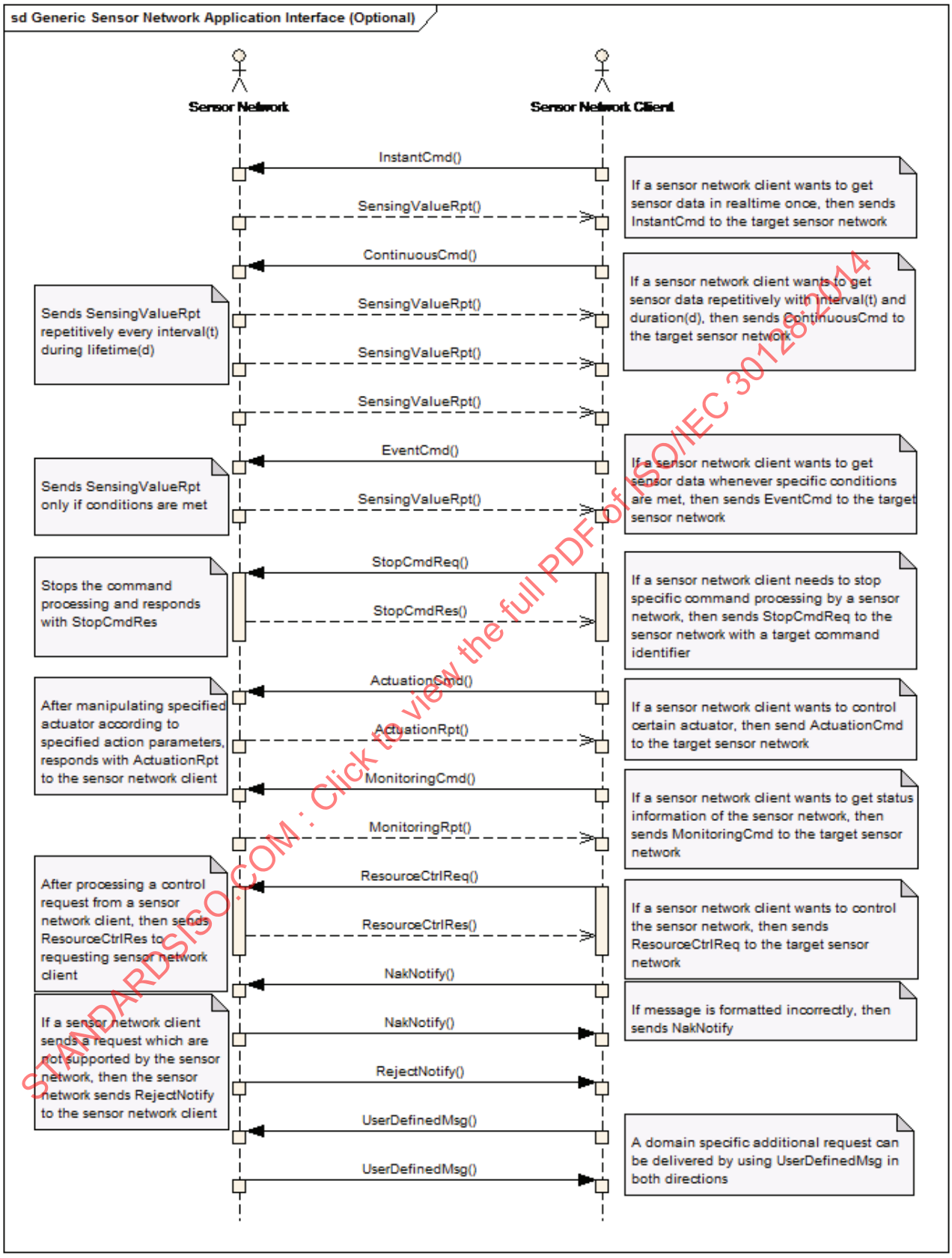

Figure 11 — Optional sensor network application interface flow

10.1.2 Base structure

Generic sensor network interface is specified as a set of messages. All messages are expressed in ASN.1 expression. Table 1 shows message base structure.

NOTE ASN.1 compilers produce implementation level codes with interfaces defined in ISO/IEC 30128 as an input

Table 1 — Message base structure

```
MessageType ::= PrintableString(“REGISTERRESOURCEREQ” | “REGISTERRESOURCERES” |
“DEREGISTERRESOURCEREQ” | “DEREGISTERRESOURCERES” | “INSTANTCMD” | “CONTINUOUS-
CMD” | “EVENTCMD” | “ACTUATIONCMD” | “MONITORINGCMD”| “SENSINGVALUERPT” | “ACTU-
ATIONRPT” | “MONITORINGRPT” | “STOPCMDREQ” | “STOPCMDRES” | “RESOURCECTRLREQ”|
“RESOURCECTRLRES” | “NAKNOTIFY” | “REJECTNOTIFY” | “USERDEFINEDMESSAGE”)

MessageBody ::= CHOICE {
 registerResourceReq      [0] RegisterResourceReq,
 registerResourceRes      [1] RegisterResourceRes,
 deRegisterResourceReq    [2] DeRegisterResourceReq,
 deRegisterResourceRes    [3] DeRegisterResourceRes,
 instantCmd               [4] InstantCmd,
 continuousCmd            [5] ContinuousCmd,
 eventCmd                 [6] EventCmd,
 sensingValueRpt          [7] SensingValueRpt,
 actuationCmd             [8] ActuationCmd,
 actuationRpt             [9] ActuationRpt,
 monitoringCmd            [10] MonitoringCmd,
 monitoringRpt            [11] MonitoringRpt,
 stopCmdReq               [12] StopCmdReq,
 stopCmdRes               [13] StopCmdRes,
 resourceCtrlReq          [14] ResourceCtrlReq,
 resourceCtrlRes          [15] ResourceCtrlRes,
 nakNotify                [16] NakNotify,
 rejectNotify             [17] RejectNotify,
 userDefinedMessage       PrintableString
}



Message ::= SEQUENCE {

 sender        PrintableString, -- sender id
 receiver      [0] PrintableString, -- receiver id
 messageType   MessageType,
 messageBody   MessageBody
}
```


All messages are formatted according to a base structure. 10.2 and 10.3 specify detailed definition of each message.

10.1.3 Sensing types and measurement units

In this international standard, sensing types and measurement units are not specified. They can be defined according to the implementation domain. Annex B gives an example specification on sensing types and measurement units. ISO/IEC/IEEE 21450, Physical Units, ISO/IEC/IEEE 21451-4 Physical Units, and ISO/IEC/IEEE 21451-7 Sensor Types may be applied.

10.2 Mandatory operations

10.2.1 Registration

Registration is an operation to register a sensor network to a sensor network client. Important functions here are decision on sensor network resource identifiers and advertisement of sensor network metadata. Sensor network resource identifiers are used by sensor network clients when sensor network clients address specific sensor network resources.

Regarding identifier assignment, it can be handled in two ways. A sensor network client may assign identifiers to sensor network resources based on its own identifier assigning scheme. In this case, duplication can be avoided easily but sensor networks should keep a mapping table which maps identifiers for sensor network clients onto identifiers used within sensor networks and vice versa. In another way, sensor networks suggest preferable identifiers which can be easily transformed into identifiers used within sensor networks without a mapping table. In this case, sensor network clients shall ensure that identifiers are unique.

Table 2 shows registration message structure and Table 3 shows registration message information.

Table 2 — Registration message structure

```
RegisterResourceType ::= PrintableString("GATENODE" | "PAN" | "SENSORNODE" | "TRANSDUCER")

RegisterResourceReq ::= SEQUENCE {

    description GateNode,
    resourceType RegisterResourceType DEFAULT "GATENODE"

}

IDList ::= SEQUENCE SIZE(1..MAX) OF PrintableString

RetCode ::= PrintableString("SUCCESS" | "BADREQUEST" | "ERROR" | "UNDEFINED" | "STORED")

RegisterResourceRes ::= SEQUENCE {

   retCode RetCode,

   idList IDList

}
```

Table 3 — Registration message information

RegisterResourceReq		
Field	Description	Etc.
description	*description* includes resource (gateway, PAN coordinator, sensor node or transducer) metadata. It shall include an identifier field to suggest an identifier which a resource prefers to use. A sensor network client shall validate a suggested identifier to avoid any conflict with other identifiers when it receives RegisterResourceReq. *description* shall include resource's capability information regarding sensor manipulation, actuator manipulation and sensor network management. In this standard, a concrete data model of description is out of scope. It can be described as like Annex A, TEDS, or in SensorML.	Regarding resource's capability information, see 9.3
resourceType	Type of resource	"GATENODE", "PAN", "SENSORNODE", "TRANSDUCER".
RegisterResourceRes		
retCode	Registration result NOTE retCode may be extended to handle more situations.	"SUCCESS", BADREQUEST", "ERROR"
idList	List of registered identifiers. If a description includes descriptions of multiple sensor network resources, then idList includes multiple identifiers which are registered successfully.	

A sensor network comprises of one gateway and PAN coordinators optionally. A PAN coordinator is consisted of at least one sensor node. A sensor node is composed of more than one transducer. A transducer is a sensor or an actuator. To provide sensor network services, each element of a sensor network shall register at sensor network clients.

When it comes to a registration order, a hierarchical order shall be kept. The gateway shall be registered first and a PAN coordinator shall be registered next. Then sensor nodes shall be registered and after that transducers shall be registered finally. Or, all elements of a sensor network may be registered altogether. Figure 12 shows two ways of registrations.

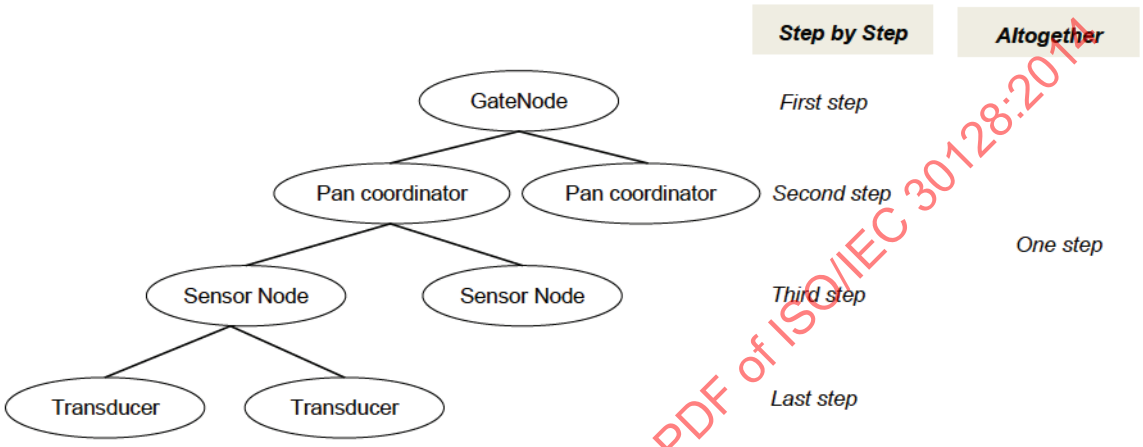

Figure 12 — Sensor network registration steps

When a sensor network resource registers at a sensor network client, it can propose its identifier or the sensor network client may assign a unique identifier to the sensor network resource. Hierarchical structure may be used as an identification scheme for sensor networks. It may have a hierarchical structure such as "*gatenodeId:panId:nodeId:transducerId*". Each id should be a unique identifier under the parent node. A *transducerId* is unique under the sensor node with *noteId*. A *nodeId* is unique under the PAN with *panId*. A *panId* is unique under the gateway with *gatenodeId*. *gatenodeId* is a unique gateway identifier and there needs a way to ensure uniqueness of *gatenodeId* within a service domain. DNS may be used to ensure the uniqueness of a gateway identifier in a global scale. Otherwise, other mechanisms may be used to ensure uniqueness of gateway identifiers in a given service domain. The advantage of this hierarchical identification scheme is that gateways, PAN coordinators and sensor nodes are free from managing an identifier mapping table.

Table 4 shows example description schema of a sensor network gateway.

Table 4 — Gateway description example

```
SupportedTransportProtocol ::= PrintableString(“XML_OVER_TCP” | “XML_OVER_UDP” |
“XML_OVER_HTTP” | “TEXT_OVER_TCP”)

SupportedTransportProtocolList ::= SEQUENCE OF SupportedTransportProtocol

SupportedCommand ::= PrintableString(“COMMAND_PUSH” | “COMMAND_INSTANT” | “COM-
MAND_EVENT” | “COMMAND_CONTINUOUS” | “COMMAND_CONTROL” | “COMMAND_MONITORING” |
“COMMAND_ACTUATION”)

SupportedCommandList ::= SEQUENCE OF SupportedCommand

SupportedCommandAttribute ::= PrintableString(“ATTRIBUTE_CONDITION” | “ATTRIB-
UTE_FUNCTION” | “ATTRIBUTE_DURATION” | “ATTRIBUTE_SHUTDOWN” | “ATTRIBUTE_RESET”
| “ATTRIBUTE_REBOOT” | “ATTRIBUTE_START_SENSING” | “ATTRIBUTE_STOP_SENSING” |
“ATTRIBUTE_PAN_ID_CHANGE” | “ATTRIBUTE_CHANNEL_ID_CHANGE”)

SupportedCommandAttributeList ::= SEQUENCE OF SupportedCommandAttribute

GateNodeSupportedOperationList ::= SEQUENCE {
   supportedCommandList SupportedCommandList,
   supportedCommandAttributeList SupportedCommandAttributeList
}
Time ::= SEQUENCE {
  time INTEGER(0 .. MAX),
  unit TimeUnit DEFAULT “SEC”
}
Location ::= SEQUENCE {
  longitude REAL,
  latitude REAL,
  altitude REAL
}
MonitoringMode ::= PrintableString(“MONITORING_PULL” | “MONITORING_PUSH”)
GateNode ::= SEQUENCE {
  id                PrintableString, -- format: host name or ip
  url               [0] PrintableString OPTIONAL,
  manufacturer      PrintableString OPTIONAL,
  productNo         [1] PrintableString OPTIONAL,
  location          Location OPTIONAL,
  dateTime          [2] UTCTime OPTIONAL,
  supportedTransportProtocolList    [3] SupportedTransportProtocolList OPTIONAL,
  supportedOperationList            [4] GateNodeSupportedOperationList,
  panList             [5] PanList OPTIONAL,
  monitoringMode      [6] MonitoringMode OPTIONAL,
  monitoringPeriod    INTEGER(0 .. MAX) OPTIONAL -- in seconds
}
```

In this International Standard, a data model of description is not restricted. But description (metadata) shall include capability information about supported operations. That is because a sensor network client shall know what kinds of operations are provided by the sensor network beforehand.

10.2.2 DeRegistration

Deregistration is an operation to break the link between a sensor network and a sensor network client. Multiple elements of sensor networks may be deregistered at a time by using one DeRegistration message. If a parent element is deregistered, then all child elements are deregistered altogether.

Table 5 shows deregistration message structure and Table 6 shows deregistration message information.

Table 5 — Deregistration message structure

```
DeRegisterResourceReq ::= SEQUENCE {

 idList IDList       /* a list of identifiers to deregister from sensor network
clients */
}
DeRegisterResourceRes ::= SEQUENCE {
  retCode RetCode,
  idList IDList

}
```

Table 6 — Deregistration message information

DeRegisterResourceReq		
Field	Description	Etc.
idList	List of sensor network resource identifiers to deregister from a sensor network client	
DeRegisterResourceRes		
retCode	Deregistration result NOTE retCode may be extended to handle more situations.	"SUCCESS", BADREQUEST", "ERROR"
idList	A list of deregistered sensor network resource identifiers from a sensor network client	

10.2.3 Sensing value report

Sensing value reporting is the operation to deliver collected sensor data from a sensor network to a sensor network client. The way of sending sensor data is based on a sensor manipulation mode.

Table 7 shows sensing value report message structure and Table 8 sows sensing value report message information.

Table 7 — Sensing value report message structure

```
SensingValueRpt ::= SEQUENCE {

  commandID    INTEGER(0..MAX) OPTIONAL,

  sensingValueList SEQUENCE OF SensingValueList

}
```

Table 8 — Sensing value report message information

SensingValueRpt		
Field	Description	Etc.
commandID	Command identifier to match a command and corresponding reports	In a push mode, a command identifier may be omitted
sensingValueList	List of collected sensor data or processed data. A sensor constructs SensingValue, a sensor node constructs SensingValueList. A gateway constructs SensingValueRpt. In this international standard, detailed structure of sensingValueList and SensingValue are not restricted. They can be expressed in O&M or can be like the following Table 9.	

Table 9 is an example of sensor data. It can be used for sensingValueList.

Table 9 — Collected sensor data structure example

```
Function ::= PrintableString("MIN" | "MAX" | "AVG" | "SUM")

SensingValue ::= SEQUENCE {

 targetID PrintableString,                          /* sensor identifier */

 sensorType PrintableString,                        /* sensor type */

 value PrintableString,                             /* measured value */

 function Function OPTIONAL                         /* applied aggregation func-
tion to value */

}

SensingValueList ::= SEQUENCE {

 timeStamp PrintableString,                          /* sending time */

 nodeID PrintableString,                            /* sensor node identifier */

 values SEQUENCE SIZE(1..MAX) OF SensingValue    /* list of sensor data collected
by a sensor node*/

}
```

10.3 Optional operations

10.3.1 Instant command

Instant command is a message to enforce sensing to sensor network resources. It specifies target resource identifiers, sensing types, conditions to check if any, and an action time if necessary.

Table 10 shows Instant command message structure and Table 11 shows Instant command message information.

Table 10 — Instant command message structure

```
Function ::= PrintableString("MIN" | "MAX" | "AVG" | "SUM")
LogicalOp ::= PrintableString("OR" | "AND")
RelationalOp ::= PrintableString("GT" | "GE" | "LT" | "LE" | "EQ" | "NE")

SensingType ::= SEQUENCE {
 function       [0] Function OPTIONAL,     /* aggregation function which to be
applied to the sensor data */
 sensorType      [1] PrintableString,     /* type of sensor data */
 unit            PrintableString         /* measurement unit of the sensor data
*/
}
SensingTypeList ::= SEQUENCE OF SensingType

ConditionType ::= SEQUENCE {
 type PrintableString,                   /* sensor data type */
 relationalOp RelationalOp,
 value PrintableString,
 logicalOp LogicalOp OPTIONAL
}
ConditionList ::= SEQUENCE OF ConditionType

InstantCmd ::= SEQUENCE {
 commandID INTEGER(0..MAX),              /* command identifier */
 targetIDList IDList,                    /* resource identifier list to perform
this command */
 sensingTypeList SensingTypeList,        /* sensing type list to collect */
 actionTime UTCTime OPTIONAL,            /* if it is set, command is processed at
actionTime by sensor nodes */
 conditionList ConditionList OPTIONAL    /* condition list if any */
}
```

Table 11 — Instant command information

InstantCmd		
Field	Description	Etc.
commandID	Command identifier to match a command with corresponding reports	
targetIDList	List of sensor network resource identifiers which handle this command. A target can be a transducer, a sensor node, a PAN coordinator, or a gateway.	
sensingTypeList	List of sensor types	
actionTime	Action time to process this command	0 means this command needs to be handled when the command is received by a resource
conditionList	List of conditions which shall be checked by the resources before sending sensor data	An order of checking condition is from the rightmost to the leftmost

10.3.2 Continuous command

Continuous command is a message to enforce a continuous sensing to sensor network resources. It specifies target resource identifiers, sensing types, conditions to check if any, a period value, a duration time and an action time if necessary.

Table 12 shows Continuous command message structure and Table 13 shows Continuous command message information.

Table 12 — Continuous command message structure

```
ContinuousCmd ::= SEQUENCE {

commandID               INTEGER(0..MAX),

targetIDList            IDList,

sensingTypeList         SensingTypeList,

actionTime          UTCTime OPTIONAL,       /* if it is set, this query is pro-
cessed at actionTime by sensor nodes */

period           Time,

duration        [0] Time OPTIONAL,

conditionList   [1] ConditionList OPTIONAL

}
```

Table 13 — Continuous command message information

ContinuousCmd		
Field	Description	Etc.
commandID	Command identifier to match a command with corresponding reports	
targetIDList	List of sensor network resource identifiers which handle this command. A target can be transducer, a sensor node, a PAN coordinator, or a gateway.	
sensingTypeList	List of sensor types	
actionTime	Action time to process this command	0 means this command needs to be handled when the command is received by a resource
period	Interval time to collect sensor data	
duration	Duration time to collect sensor data. If this field does not exist, then *duration* value is infinite.	
conditionList	List of conditions which shall be checked by the resources before sending sensor data	An order of checking condition is from the rightmost to the leftmost

10.3.3 Event command

Event command is a message to enforce an event-based sensing to sensor network resources. It specifies target resource identifiers, sensing types, conditions to check, and an action time if necessary.

Table 14 shows Event command message structure and Table 15 shows Event command message information.

Table 14 — Event command message structure

```
EventCmd ::= SEQUENCE {

commandID           INTEGER(0..MAX),

targetIDList        IDList,

sensingTypeList     SensingTypeList,

actionTime          UTCTime OPTIONAL,      /* if it is set, this query is pro-
cessed at actionTime by sensor nodes */

conditionList       ConditionList
}
```

Table 15 — Event command message information

EventCmd		
Field	Description	Etc.
commandID	Command identifier to match a command with corresponding reports	
targetIDList	List of sensor network resource identifiers which handle this command. A target can be a transducer, a sensor node, a PAN coordinator, or a gateway.	
sensingTypeList	List of sensor types	
actionTime	Action time to process this command	0 means this command needs to be handled when the command is received by a resource
conditionList	List of conditions which shall be checked by the resources before sending sensor data	An order of checking condition is from the rightmost to the leftmost

10.3.4 Stop command

Stop command is to stop an on-going command processing. By using StopCmdReq, a sensor network client may request to stop on-going ContinuousCmd processing, EventCmd processing, push mode processing, actuator processing, or MonitoringCmd processing.

Table 16 shows Stop command message structure and Table 17 shows Stop command message information.

Table 16 — Stop command message structure

```
StopCmdReq ::= SEQUENCE {

commandID      INTEGER(0 .. MAX) /* commandID is handled uniquely to avoid any
duplication during operations */

}

StopCmdRes ::= SEQUENCE {

retCode        RetCode

}
```

Table 17 — Stop command message information

StopCmdReq		
Field	Description	Etc.
commandID	Command identifier to match a command with a corresponding response	
StopCmdRes		
retCode	Result of StopCmdReq processing NOTE retCode may be extended to handle more situations	"SUCCESS", BADREQUEST", "ERROR"

10.3.5 Actuation command

Actuation command is a message to order a target actuator to act complied with the actuationValue.

Table 18 shows Actuation command message structure and Table 19 shows Actuation command message information.

Table 18 — Actuation command message structure

```
ActuationCmd ::= SEQUENCE {
 commandID     INTEGER(0..MAX),
 targetIDList  IDList,          /* target resource identifier */
 actuatorType PrintableString,
 actionValue PrintableString
 actionTime UTCTime OPTIONAL,    /* if it is set, command is processed at
actionTime by sensor nodes */

}

ActuationValue ::= SEQUENCE {
 targetID PrintableString,
 actionValue PrintableString

}


ActuationValueList ::= SEQUENCE {
 actuationValue SEQUENCE SIZE(1..MAX) OF ActuationValue

}

ActuationRpt ::= SEQUENCE {
 commandID     INTEGER(0..MAX),
 actuationValueList ActuationValueList

}
```

Table 19 — Actuation command message information

ActuationCmd		
Field	Description	Etc.
commandID	Command identifier to match a command with a corresponding response	
targetIDList	List of resource identifiers to handle this command. A target can be a transducer, a sensor node, a PAN coordinator, or a gateway. if targetIDList includes gateway identifiers, PAN identifiers or sensor node identifiers, then it means actuators which are members of the specified targetID shall process the command only if "actuatorType" is matched.	
actuatorType	Type of actuator	
actionValue	Actuation parameters. Parameters are decided depending on the type of actuator and may be the structured type for complex actuators. Detailed design of parameters are out of scope.	
actionTime	Action time to process this command	0 means this command needs to be handled when the command is received by a resource
ActuationRpt		
commandID	Command identifier to match a command with a corresponding response	
actuationValue-List	list of actuation results by actuators	

10.3.6 Monitoring command

Monitoring command is a message to monitor status of sensor networks. It may be implemented in two ways. The simple way is to push MonitoringRpt repetitively based on a sensor network configuration without an explicit request from a sensor network client. The other way is by using both MonitoringCmd and MonitoringRpt. It is an implementation issue but shall be advertised to sensor network clients beforehand.

Table 20 shows Monitoring command message structure and Table 21 shows Monitoring command message information.

Table 20 — Monitoring command message structure

```
MonitoringType ::= PrintableString(“BATTERY” | “LOCATION” | “ISALIVE”)
MonitoringTypeList ::= SEQUENCE SIZE(1..MAX) OF MonitoringType
DataType ::= PrintableString(“BOOLEAN” | “INTEGER” | “FLOAT” | “DOUBLE” |
“STRING”)


MonitoringCmd ::= SEQUENCE {
 commandID        INTEGER(0..MAX),
 targetIDList     IDList,
 monitoringTypeList MonitoringTypeList

}


MonitoringValue ::= SEQUENCE {
 monitoringType    PrintableString,
 dataType          DataType,
 unit              PrintableString,
 value             PrintableString

}

MonitoringValueList ::= SEQUENCE SIZE(1..MAX) OF MonitoringValue


MonitoringRpt ::= SEQUENCE {
 commandID     INTEGER(0 .. MAX) OPTIONAL,
 targetID      PrintableString,
 monitoringValueList MonitoringValueList

}
```


Table 21 — Monitoring command message information

MonitoringCmd		
Field	Description	Etc.
commandID	Command identifier to match a command and a corresponding response	
targetIDList	List of sensor network resource identifiers to handle this command. A target can be a transducer, a sensor node, a PAN coordinator, or a gateway.	
monitoring-TypeList	list of monitoring types	BATTERY, LOCATION, ISALIVE

Table 21 *(continued)*

MonitoringCmd		
Field	Description	Etc.
MonitoringRpt		
commandID	Command identifier to match a command and corresponding response	In Push Mode, a command identifier may be omitted
targetID	List of sensor network resource identifiers which send MonitoringRpt message	
monitoringValueList	List of requested monitoring values	

10.3.7 Resource control

Resource control is to control a sensor network. By using ResourceCtrlReq a sensor network client requests a sensor network to reset, shutdown, reboot, start push sensor data, stop push sensor data, update PAN channel, etc.

Table 22 shows Resource control message structure and Table 23 shows Resource control message information.

Table 22 — Resource control message structure

```
ResourceCtrlReq ::= SEQUENCE {

 targetID      PrintableString,

 controlType   SupportedCommandAttribute, /* attribute of control */

controlValue   PrintableString

}



ResourceCtrlRes ::= SEQUENCE {

retCode      RetCode,

resultValue  PrintableString

}
```

Table 23 — Resource control message information

ResourceCtrlReq		
field	description	Etc.
targetID	Target sensor network resource identifier which to be controlled. A target can be a transducer, a sensor node, a PAN coordinator, or a gateway.	
controlType	Type of control NOTE controlType may be extended to handle more situations	"ATTRIBUTE_SHUTDOWN" | "ATTRIBUTE_RESET" | "ATTRIBUTE_REBOOT" | "ATTRIBUTE_START_SENSING" | "ATTRIBUTE_STOP_SENSING" | "ATTRIBUTE_PAN_ID_CHANGE" | "ATTRIBUTE_CHANNEL_ID_CHANGE"
controlValue	Control value	Decided depending on controlType
ResourceCtrlRes		
retCode	Result of the request NOTE retCode may be extended to handle more situations	"SUCCESS", BADREQUEST", "ERROR", "UNDEFINED"
resultValue	Explanation of the result	

NOTE a sensor network client can initiate push mode sensor data collection by using *ResourceCtrlReq* with controlType=" ATTRIBUTE_START_SENSING". And it can terminate push mode sensor data collection by using ResourceCtrlReq with controlType=" ATTRIBUTE_STOP_SENSING".

10.3.8 User-defined message

User-defined message is for scalability. If there is a predefined function between a specific sensor network client and a specific sensor network, they can exchange information by using UserDefinedMsg. The content of the *message* is predefined between a specific sensor network client and a specific sensor network.

Table 24 shows User-defined message structure and Table Table 25 shows User-defined message information.

Table 24 — UserDefined message structure

```
UserDefinedMsg ::= SEQUENCE {

 sourceID PrintableString, /* identifier of a message sender */

 targetID PrintableString,

 message PrintableString

}
```