ISO/IEC JTC 1/WG 7
Working Group on Sensor Networks

Document Number:	N135
Date:	2011-01-19
Replace:	
Document Type:	Working Draft Text
Document Title:	Pre-2nd Working Draft of ISO/IEC 20005, Information technology — Sensor Networks — Services and Interfaces Supporting Collaborative Information Processing in Intelligent Sensor Networks
Document Source:	Project Editor
Document Status:	This document is circulated for comments by WG 7 members (1 month period). This document and comments received will be considered at the 3rd JTC 1/WG 7 meeting in Sophia Antipolis.
Action ID:	COM
Due Date:	2011-02-20
No. of Pages:	69

ISO/IEC JTC 1/WG 7 Convenor:

Dr. Yongjin Kim, Modacom Co., Ltd (Email: cap@modacom.co.kr)

ISO/IEC JTC 1/WG 7 Secretariat:

Ms. Jooran Lee, Korean Standards Association (Email: jooran@kisi.or.kr)

Reference number of working document: **ISO/IEC JTC 1/ WG7 N XXX**

Date: 2011-01-10

Reference number of document: **ISO/IEC 20005**

Committee identification: ISO/IEC JTC 1/WG 7

Secretariat: KATS

Information technology — Sensor Networks — Services and Interfaces Supporting Collaborative Information Processing in Intelligent Sensor Networks

Warning

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: International Standard
Document subtype: If applicable
Document stage: (20) Preparatory stage
Document language: E

Contents

Page

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

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 drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

Technical Committee ISO/IEC JTC 1, Information Technology, Working Group 7, *Sensor Networks*, prepared ISO/IEC 20005.

Introduction

Intelligent sensor networks are becoming increasingly attractive in a wide range of applications to meet generic challenges from the dynamic changes of deploying environment, network status and application performance requirement. Collaborative information processing (CIP), which closely integrates information processing algorithms with collaboration mechanisms, is an essential technology helping intelligent sensor networks to guarantee system performance in real application scenarios. This standard specifies services and interfaces supporting CIP in intelligent sensor networks.

Information technology — Sensor Networks — Services and Interfaces Supporting Collaborative Information Processing in Intelligent Sensor Networks

1 Scope

This international standard specifies services and interfaces supporting collaborative information processing (CIP) in intelligent sensor networks which includes:

- CIP functionalities and CIP functional model
- Common services supporting CIP
- Common service interfaces to CIP

2 Normative references

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

ISO/IEC JTC1 SGSN N149, *SGSN Technical Document Version 3.*

ITU-T Recommendation Y.2221, *Requirements for support of Ubiquitous Sensor Network (USN) applications and services in NGN environment (2009).*

ITU-T Recommendation X.902 | ISO/IEC 10746-2 :2010, *Information technology – Open distributed processing – Reference Model: Foundations.*

ITU-T Recommendation X.903 | ISO/IEC 10746-3 :2010, *Information technology – Open distributed processing – Reference Model: Architecture.*

ISO/IEC JTC1 WD 29182-1, *Information technology – Sensor Networks: Sensor Network Reference Architecture (SNRA) – Part 1: General Overview and Requirements.*

ISO/IEC JTC1 WD 29182-2, *Information technology – Sensor Networks: Sensor Network Reference Architecture (SNRA) – Part 2: Vocabulary and Terminology.*

ISO/IEC JTC1 WD 29182-4, *Information technology – Sensor Networks: Sensor Network Reference Architecture (SNRA) – Part 4: Entity Models.*

ISO/IEC JTC1 WD 29182-6, *Information technology – Sensor Networks: Sensor Network Reference Architecture (SNRA) – Part 6: Application Profiles.*

ISO/IEC JTC1 WD 29182-7, *Information technology – Sensor Networks: Sensor Network Reference Architecture (SNRA) – Part 7: Interoperability guidelines.*

(To be added)

3 Terms and definitions

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

3.1
actuator
An actuator is a device that performs physical response caused by input signal. [ISO/IEC JTC1 WD 29182-2]

3.2
collaborative information processing
A form of information processing in which multiple discrete components or entities participate in a manner of collaboration, in order to enhance processing efficiency and to improve quality and reliability of the results.

3.3
event
Anything that happens or is contemplated as happening at an instant or over an interval of time. [OGC document 09-032]

3.4
information
Knowledge concerning objects, such as facts, events, things, processes, or ideas, including concepts, which within a certain context has a particular meaning. [ISO/IEC 2382-1]

3.5
information processing
The manipulation of data or information so that new data or information which is implicit in the original be appeared in a useful form, or with which further information processing can be applied and/or be utilized to make a response suitable within the context of an objective, problem or situation.

3.6
sensor network
A sensor network is a system of spatially distributed sensor network nodes interacting with each other and, depending on applications, interacting with other infrastructure in order to acquire, process, transfer, and provide information extracted from the physical world with a primary function of information gathering. [ISO/IEC JTC1 WD 29182-2]

3.7
sensor network application
The sensor network application is a user case of sensor networks supporting a set of sensor network services for users. [ISO/IEC JTC1 SGSN N149]

3.8
sensor network service
A structural set of capabilities or functions which are offered by the sensor nodes or sensor networks.

3.9
sensor node
A sensor node is a device that consists of at least one sensor and zero or more actuators, and has processing and networking capabilities through wired or wireless means. [ISO/IEC JTC1 SGSN N149]

3.10
service set or service subset
A group or subgroup of services organized to provide common mechanisms or facilities to meet certain requirements from users or applications.

3.11
viewpoint
A form of abstraction achieved using a selected set of architectural concepts and structuring rules, in order to focus on particular concerns within a system. [ISO/IEC 10746]

4 Abbreviations

For the purposes of this document, the following abbreviations apply.

CDE	Capability Declaration Entity
CIP	Collaborative Information Processing
CRSE	Communication Requirement Specification Entity
CS	Core Service
CSPE	Collaborative Strategy Planning Entity
ES	Enhanced Service
FAR	False Alarming Rate
FCR	Functional Capability Requirement
GSR	Generalized System Requirement
OSI/RM	Open System Interconnection/Reference Model
QoS	Quality of Service
SAP	Service Access Point

5 General Description

Sensor networks have been widely deployed in different application domains ranging from environment monitoring, transportation, industries, and healthcare etc. Wired/wireless sensor networks can be regarded as an extension of Internet towards the physical world. In order to meet challenges from intrinsic environment complexity, large orders of magnitude network scaling and dynamic application requirements, intelligent sensor networks are developed to provide new system capabilities such as environment self-adaptability, dynamic task supporting and autonomous system maintenance. Collaborative information processing (CIP) is an essential technology to implement new functionalities of intelligent sensor network and finally to provide improved services to users of intelligent sensor networks.

This clause gives general description of sensor networks and requirements of intelligent sensor networks. Overviews of CIP and its functional models are presented, followed by general explanation on core and enhanced services supporting CIP in intelligent sensor networks.

5.1 Sensor networks overview

Compared with the traditional networks, sensor networks not only provide information transmission service but also provide information sensing, processing, provision and other services. Figure 1 shows an overview of sensor networks system from the layer architectural view.

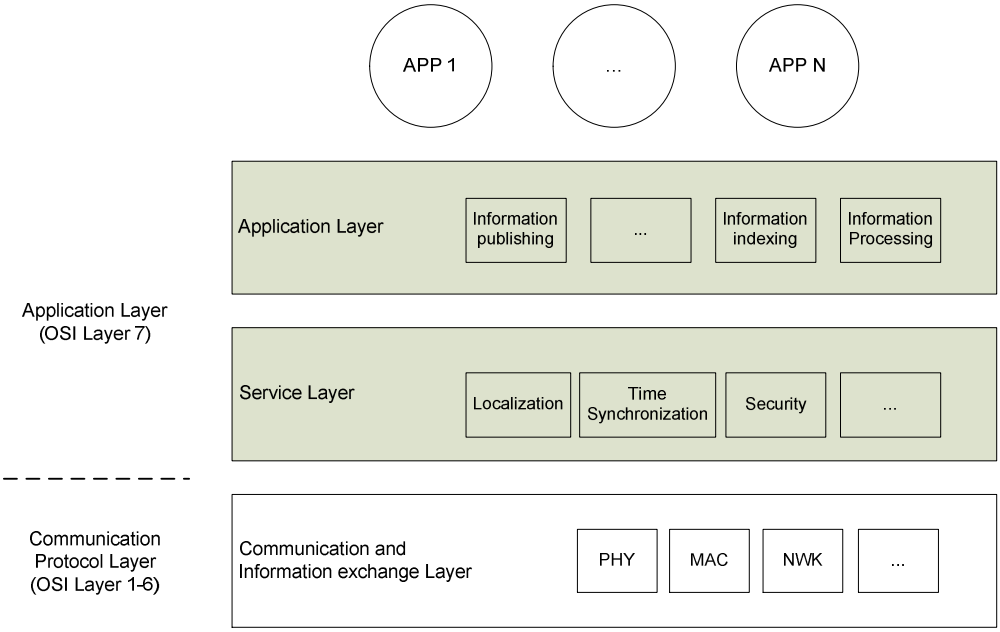

Figure 1 — Layer overview of sensor networks system architecture

Communication and information exchange layer implements functionalities fulfilled by the lower layers in the OSI/RM stack, including physical layer, data link layer, network layer, and transport layer. Application layer provides services to individual applications and implements functions such as information publishing, information indexing and information processing etc. Between application layer and communication and information exchange layer, the service layer provides generic common services to entities in the above application layer. In the context of sensor networks, a lot of generic common services need to be implemented including localization service, time synchronization service, security service and other services.

5.2 Requirements of intelligent sensor networks

Besides the generalized system requirements (GSR) and generalized functional capability requirements (FCR) of sensor networks, there are additional unique requirements on intelligent sensor networks to meet challenges from the dynamic changes of deploying environment, network status and application performance requirement.

- **Environmental self-adaptability:** Intelligent sensor network shall adapt to obtain required system performances if the physical environment changes. As an example, an intelligent sensor network based anti-intrusion system should guarantee consistent system performance such as false alarming rate (FAR) when the environment in which the network is deployed changes.

- **Dynamic task supporting:** Intelligent sensor network shall support dynamic tasks including dynamic task assigning, dynamic service-providing and dynamic quality of service (QoS).

- **Autonomous system maintenance:** Intelligent sensor network shall autonomously maintain system functionalities in case of network scaling, node mobility and node failures.

5.3 Collaborative information processing overview

The key difference between traditional telecommunication infrastructures and sensor networks based information service system is that sensor networks based information service system collects low-level sensory data, extracts application-specific information from these sensory data, and make attempt to obtain high-level data, information, and knowledge about physical world.

Integrated with other entities such as sensory information description, sensor identification and sensory information storage, CIP concerns on how to resource-efficiently fulfil dynamic tasks specified by information service consumer. Though different sensor network application scenarios normally require scenario-specific services, collaboration is an indispensable requirement for senor network based information service to handle constraints in energy, computing, storage and communication bandwidth. To information service provider, it also has to deal with technical challenges from issues such as task dynamics, measurement uncertainty, node mobility and environmental changing.

The aim of CIP in sensor networks is to improve system efficiency, enhance quality of service and guarantee system performance. It provides efficient mechanisms and/or protocols to meet generic challenges from the dynamic changes of deploying environment, network status and application performance requirement.

CIP can be viewed from three distinct viewpoints. Figure 2 shows a three-dimensional conceptual model of CIP.

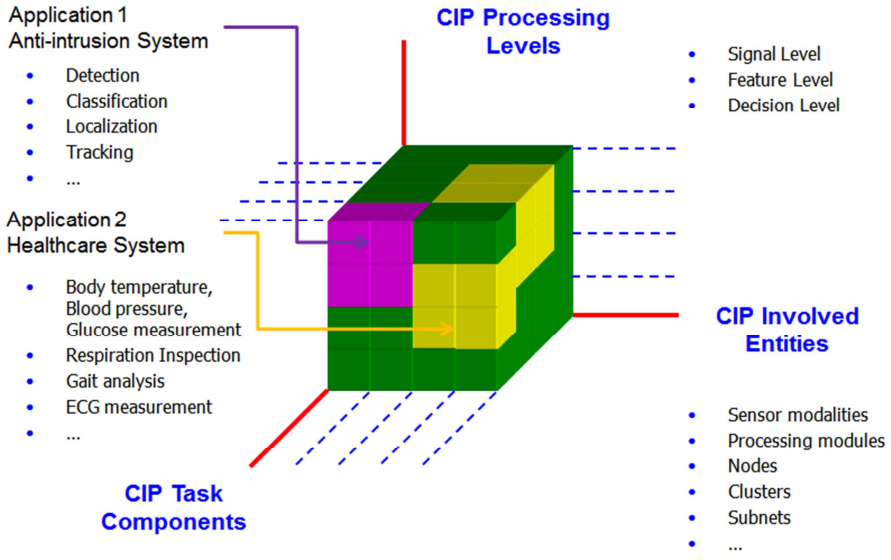

Figure 2 — Conceptual model of collaborative information processing

The first viewpoint is CIP Processing Level viewpoint. In this viewpoint, CIP can be implemented on different processing levels, which includes data, feature and decision processing levels. The second viewpoint is CIP Involved Entity viewpoint. Involved Entities in CIP could be sensor modalities, processing modules, nodes, clusters, and even subnets. CIP can thirdly be viewed from Task Component perspective. Elements in this viewpoint depend on the specific application scenarios of sensor networks. In an anti-intrusion application, target detection, localization, classification, and tracking could be task components for security services. In healthcare context, task components may include blood pressure/temperature measurement, respiration inspection, and gait analysis. Specific selections and combinations using elements from these three viewpoints correspond to different application task implementations, or personalized services using sensor networks.

5.4 Functional model of collaborative information processing

Figure 3 shows a functional model of collaborative Information Processing from functional entities point of views. In this model, CIP can be characterized by three distinct entities, which is named as capability declaration entity (CDE), collaborative strategy planning entity (CSPE) and communication requirement specification entity (CRSE).

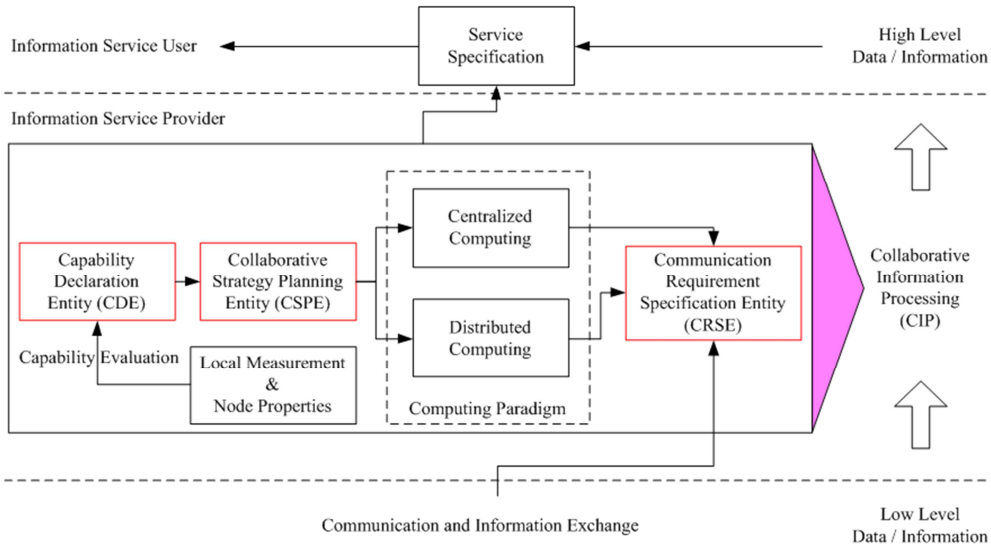

Figure 3 — Functional model of collaborative information processing

Capability declaration entity (CDE) declares capabilities of one sensor node to other nodes. Capabilities include not only individual node information on sensing modality configuration, sensing range, residual energy, location, storage and communication bandwidth etc., but also include certain characteristic information of sensory data collected by individual sensor node. One of the representative characteristics on sensory data is signal-to-noise ratio (SNR) value. Other characteristics include signal energy, estimated distance from target and sensor nodes, and state parameter prediction, etc. In other words, one sensor node should qualify itself to be a CIP participant before any actual CIP procedure is triggered. CDE requires a preliminary local capability evaluation process which uses information of local measurement and node property.

Collaborative strategy planning entity (CSPE) is the second and probably the most important entity in CIP. CSPE uses available information provided by CDE and forms global or regional maps or scopes on signal and information processing problems. With certain cost functions or utility measures, CSPE tries to find a resource-efficient solution to collaborative strategy planning problem, with which the best information processing performance can be achieved at the same time. Two computing paradigms can be used in the implementation of resulting solution from CSPE. One is centralized computing paradigm; the other is distributed computing paradigm.

Communication requirement specification entity (CRSE) acts as interface between information service provider and Communication and Information Exchange. CRSE defines parameters, languages or protocols to clearly describe requirements on communication and information exchange. Different requirements, such as end-to-end delay, time jitter, bit error and other QoS parameters should be specified.

5.5 Services supporting CIP Overview

A lot of generic common services can be provided by the service layer as shown in Figure 1. In the context of intelligent sensor networks, this standard specifies a subset of these generic common services which interface with CIP entities in the application layer and support implementation of corresponding CIP entity functionalities.

Services supporting CIP can be conceptually divided into two classes: core services (CS) and enhanced services (ES), as shown in Figure 4. Core services include fundamental and essential services which can be provided directly and individually to CIP entities. Enhance services is implemented through service combination and integration of two or more core services or other generic common services provided by the service layer.

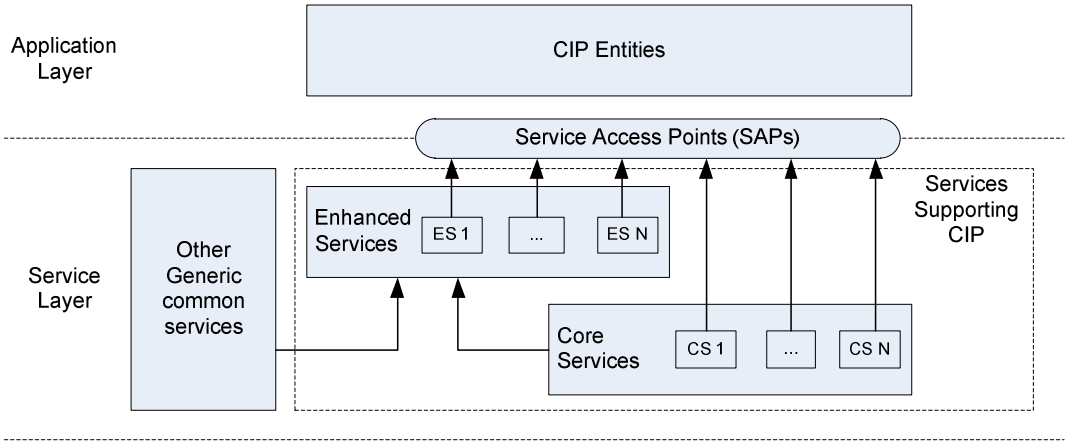

Figure 4 —Overview of services supporting CIP

5.5.1 Core services supporting CIP

The core services (CS) supporting CIP includes:

- **Event service:** This service implements functionalities concerning on the process of event subscription, registration, cancellation and un-subscription. Event may be generated due to environmental change, new physical signal occurrence and network status dynamics.

- **Logical grouping service:** This service implements functionalities concerning on the process of establishment and management of logical group for the implementation of CIP in the application layer. Logical grouping service provides mechanism for establishing collaborative relationship among entities in intelligent sensor network.

- **Data synchronization and registration service:** This service implements functionalities to synchronize different types of data sources and provides registration functions.

- **Information description service:** This service provides mechanisms to establish ways or methods to describe information in intelligent sensor network. Information can be the input parameters to CIP processes, and it can also be the output results from CIP processes

- **Node-to-node inter-activation service:** This service provides mechanisms to initiate tasks executing in one node by another node or to trigger modules from one node to another node. Dynamic tasking can be supported by this core service.

- **Parameter adaptation service:** This service provides mechanisms to adapt or reconfigure parameters for CIP. Parameter adaptation service is one of the essential services to guarantee system performance in case of dynamic changes of deploying environment and application requirement.

5.5.2 Enhanced services supporting CIP

The enhanced services (ES) supporting CIP includes:

- **QoS management service:** This service provides mechanisms to define and update QoS profiles, and to apply QoS profiles. In intelligent sensor networks, QoS shall be considered from both information processing perspective and communication processing perspective. QoS management service uses logical grouping service and parameter adaptation service.

- **CIP-driven scheduling service:** This service provides functions to control and schedule node states upon the request of CIP entities instead of node management entities in intelligent sensor networks. This service can help to implement application-oriented networking and on demand task scheduling. CIP-driven scheduling service use event service, logical grouping service, parameter adaptation service and

node-to-node inter-activation service, and other generic common services including neighbour finding service.

- **Adaptive sensing service:** This service provides mechanisms to adaptively apply sensing rules and mechanism according to different event occurrence and in different contexts in intelligent sensor network. Adaptive sensing service can provide autonomous system maintenance and system adaptability in intelligent sensor network. Adaptive sensing service uses event service, information description service and other generic common services including sensor configuration service.

6 Core Services and Interfaces Specifications

This clause specifies core services (CS) supporting CIP in intelligent sensor networks. Service primitives and parameters of primitives are defined for each core service. In Table 1, the names of service access points (SAPs) through which specific service is provided are listed.

Table 1 — Core services and the names of SAPs

Service name	SAP name
Event service	EVENT-SAP
Logical grouping service	LG-SAP
Data synchronization and registration service	SYNREG-SAP
Information description service	INFO-SAP
Node-to-node inter-activation service	N2NACT-SAP
Parameter adaptation service	PAR-SAP

6.1 Event service

In this subclause, event service in the service layer is defined. Event service is provided through EVENT-SAP. The EVENT-SAP is the logical interface between event service entity in the service layer and CIP entity in the application layer. This logical interface incorporates a set of primitives and their definitions. These primitives and definitions are described conceptually here, but through this the process of the parameters exchanged between the service layer and the application layer can be understood. Table 2 lists the primitives supported by the EVENT-SAP. Table 3 outlines primitive parameters.

Table 2 — EVENT-SAP primitive summary

Name	Request	Indication	Response	Confirm
EVENT-SUB	6.1.1	6.1.2		6.1.3
EVENT-REG		6.1.4		
EVENT-UNSUB	6.1.5			6.1.6

Table 3 — EVENT-SAP primitive parameters

Name	Type	Valid range	Description
EVSubSourceID	Integer	0x0000-0xFFFF	Source node ID of event subscription
EVSubDestinationID	Integer	0x0000-0xFFFF	Destination node ID of event subscription
EVSubModel	Enumeration	CHANNEL, TYPE, FILTER, GROUP	Event subscription models.
EVSubValue	Integer	Subscription model specific	Value for specific event subscription model.
EV_Time	Double	Implementation specific	A time indication for the event occurrence, as provided by the service layer. Return from destination node.
EVSubResultCode	Enumeration	SUCCESS, INVALID_MODEL, INVALID_VALUE, FAILED	Result code of event service operation.

6.1.1 EVENT-SUB.request

This primitive requests the process of event subscription from the application layer.

6.1.1.1 Definition of service primitives

The syntax of this primitive is:

EVENT-SUB.request {

EVSubSourceID,

EVSubDestinationID,

EVSubModel,

EVSubValue

}

Table 3 defines the parameters of this primitive.

6.1.1.2 When generated

This occurs by CIP entity to subscribe event service from the entity of the service layer.

6.1.1.3 Effect of receipt

On receipt of this primitive, the entity providing the event service implements event subscription in the EVSubDestinationID node for the EVSubSourceID node. Four types of event subscription models are provided: CHANNEL, TYPE, FILTER and GROUP.

In CHANNEL event subscription model, event channels are like television channels – if you view a channel, you receive all program broadcast on that channel. Subscribers in this model may listen to several channels. On the other hand, an event is not necessarily associated with one specific channel – it may be distributed over several channels. In TYPE event subscription model, events are filtered using EVSubValue based upon their types. When a subscriber is only interested in a subset of all notifications available at a producer, FILTER event subscription model can be used. EVSubValue specifies filter rules. The subscriber defines the filter criteria which are checked by the producer. Subscribers using the same filter criteria can be grouped together. A producer in GROUP event subscription model delivers event occurrences to the whole group. Group is identified by EVSubValue.

6.1.2 EVENT-SUB.indication

This primitive indicates the event subscription from the service layer to CIP entity.

6.1.2.1 Definition of service primitives

The syntax of this primitive is:

EVENT-SUB.indication {

EVSubSourceID,

EVSubDestinationID,

EVSubModel,

EVSubValue

}

Table 3 defines the parameters of this primitive.

6.1.2.2 When generated

This occurs when the service layer indicates an event subscription to CIP entity.

6.1.2.3 Effect of receipt

On receipt of this primitive, the CIP entity is indicated an event subscription.

6.1.3 EVENT-SUB.confirm

This primitive confirms an event subscription from the service layer to the CIP entity.

6.1.3.1 Definition of service primitives

The syntax of this primitive is:

EVENT-SUB.confirm {

EVSubSourceID,

EVSubDestinationID,

EVSubResultCode

}

Table 3 defines the parameters of this primitive.

6.1.3.2 When generated

This primitive reports a result of event subscription request.

6.1.3.3 Effect of receipt

If the EVSubResultCode is SUCCESS, it means the event subscription is successful, otherwise an error is indicated.

6.1.4 EVENT-REG.indication

This primitive indicates the event occurrence from the service layer to CIP entity.

6.1.4.1 Definition of service primitives

The syntax of this primitive is:

EVENT-REG.indication {

EVSubSourceID,

EVSubDestinationID,

EV_Time

}

Table 3 defines the parameters of this primitive.

6.1.4.2 When generated

This occurs when the service layer indicates event occurrences to CIP entity. If one or more events occur or are detected, this primitive indication may be generated more than once.

6.1.4.3 Effect of receipt

On receipt of this primitive, the CIP entity is indicated the occurrence of events. The time when event occurs or the time when event is detected is provided by EV_Time.

6.1.5 EVENT-UNSUB.request

This primitive requests the cancellation of event subscription from the application layer.

6.1.5.1 Definition of service primitives

The syntax of this primitive is:

EVENT-UNSUB.request {

EVSubSourceID,

EVSubDestinationID,

EVSubmodel,

EVSubValue

}

Table 3 defines the parameters of this primitive.

6.1.5.2 When generated

This occurs by CIP entity to unsubscribe event service from the entity of the service layer.

6.1.5.3 Effect of receipt

On receipt of this primitive, the entity providing the event service cancels event subscription in the EVSubDestinationID node for the EVSubSourceID node.

6.1.6 EVENT-UNSUB.confirm

This primitive confirms an event subscription cancellation from the service layer to the CIP entity.

6.1.6.1 Definition of service primitives

The syntax of this primitive is:

EVENT-UNSUB.confirm {

EVSubSourceID,

EVSubDestinationID,

EVSubResultCode

}

Table 3 defines the parameters of this primitive.

6.1.6.2 When generated

This primitive confirms cancellation of event subscription.

6.1.6.3 Effect of receipt

If the EVSubResultCode is SUCCESS, it means the event subscription cancellation is successful, otherwise an error is indicated.

6.2 Logical grouping service

In this subclause, logical grouping service in the service layer is defined. Logical grouping service is provided through LG-SAP. The LG-SAP is the logical interface between logical grouping service entity in the service layer and CIP entity in the application layer. This logical interface incorporates a set of primitives and their definitions. These primitives and definitions are described conceptually here, but through this the process of

the parameters exchanged between the service layer and the application layer can be understood. Table 4 lists the primitives supported by the LG-SAP. Table 5 outlines primitive parameters.

Table 4 — LG-SAP primitive summary

Name	Request	Indication	Response	Confirm
LG-ESTABLISH	6.2.1	6.2.2		6.2.3
LG-MEMBERIN	6.2.4			6.2.5
LG-MEMBEROUT	6.2.6			6.2.7
LG-DISMISS	6.2.8	6.2.9		6.2.10
LG-QUERY	6.2.11			6.2.12

Table 5 — LG-SAP primitive parameters

Name	Type	Valid range	Description
LGRequestorID	Integer	0x0000-0xFFFF	Requestor node ID of logical grouping
LGCoordinatorID	Integer	0x0000-0xFFFF	Coordinator node ID of logical group.
LGMaxNum	Integer	0x0000-0xFFFF	The maximum number of logical group members
LGMemberINID	Integer	0x0000-0xFFFF	Member ID that requests joining in a logical group.
LGMemberOUTID	Integer	0x0000-0xFFFF	Member ID that requests quitting a logical group.
LGAttributeNum	Integer	0x00-0xFF	The number of logical group attributes.
LGAttribute		Implementation specific	Data structure of name and value of specific logical group attribute. Return from the coordinator node.
LGResultCode	Enumeration	SUCCESS, FAILED, MEMBER_NUM_OVERFLOW	Result code of logical grouping service operation.

		INVALID_LG_ATTRIBUTE	

6.2.1 LG-ESTABLISH.request

This primitive requests the establishment of a logical group from the application layer.

6.2.1.1 Definition of service primitives

The syntax of this primitive is:

LG-ESTABLISH.request {

LGRequestorID,

LGCoordinatorID,

LGMaxNum

}

Table 5 defines the parameters of this primitive.

6.2.1.2 When generated

This occurs by CIP entity from the application layer to request establishment of a logical group.

6.2.1.3 Effect of receipt

On receipt of this primitive, the LGCoordinatorID node establishes a logical group and declares itself as the coordinator of the new logical group. The LGCoordinatorID is used as the name or identifier of the new logical group. A logical group membership table with up to LGMaxNum entries is established and hence maintained within the LGCoordinatorID node. A node can only be the coordinator of no more than one logical group, but it can be simultaneously a member of multiple logical groups.

6.2.2 LG-ESTABLISH.indication

This primitive indicates the logical group establishment from the service layer to the local CIP entity.

6.2.2.1 Definition of service primitives

The syntax of this primitive is:

LG-ESTABLISH.indication {

LGRequestorID,

LGCoordinatorID,

LGMaxNum

}

Table 5 defines the parameters of this primitive.

6.2.2.2 When generated

This occurs when the service layer indicates establishment of a logical group to CIP entity.

6.2.2.3 Effect of receipt

On receipt of this primitive, the CIP entity is indicated the establishment of a logical group and hence attributes of this logical group can be queried.

6.2.3 LG-ESTABLISH.confirm

This primitive confirms a logical group establishment from the service layer to the CIP entity in the application layer.

6.2.3.1 Definition of service primitives

The syntax of this primitive is:

LG-ESTABLISH.confirm {

LGRequestorID,

LGCoordinatorID,

LGResultCode

}

Table 5 defines the parameters of this primitive.

6.2.3.2 When generated

This primitive reports a result of logical group establishment request.

6.2.3.3 Effect of receipt

If the LGResultCode is SUCCESS, it means a logical group coordinated by the LGCoordinatorID node is successful. If the LGResultCode is FAILED, an error is indicated to the LGRequestorID node.

6.2.4 LG-MEMBERIN.request

This primitive requests a membership of a logical group from the application layer.

6.2.4.1 Definition of service primitives

The syntax of this primitive is:

LG-MEMBERIN.request {

LGCoordinatorID,

LGMemberINID

}

Table 5 defines the parameters of this primitive.

6.2.4.2 When generated

This occurs by CIP entity from the application layer in the LGMemberINID node to request a membership of a logical group which is coordinated by the LGCoordinatorID node.

6.2.4.3 Effect of receipt

On receipt of this primitive, if the current member number is less than the maximum member number of the logical group, the LGCoordinatorID node adds the LGMemberINID into the membership table of the logical group and the current member number is increased by 1. Otherwise, a LGResultCode value is generated to indicate a MEMBER_NUM_OVERFLOW error.

6.2.5 LG-MEMBERIN.confirm

This primitive confirms a result of the request of a membership of a logical group to CIP entity in the application layer.

6.2.5.1 Definition of service primitives

The syntax of this primitive is:

LG-MEMBERIN.confirm {

LGCoordinatorID,

LGMemberINID,

LGResultCode

}

Table 5 defines the parameters of this primitive.

6.2.5.2 When generated

This primitive reports a result of membership request to a logical group.

6.2.5.3 Effect of receipt

If the LGResultCode is SUCCESS, it means that the LGMemberINID node successfully joins a logical group coordinated by the LGCoordinatorID node. If the LGResultCode is MEMBER_NUM_OVERFLOW, the membership request is not successful due to the constraint of the maximum member number. Otherwise, a FAILED error is confirmed.

6.2.6 LG-MEMBEROUT.request

This primitive requests a membership cancellation of a logical group from the application layer.

6.2.6.1 Definition of service primitives

The syntax of this primitive is:

LG-MEMBEROUT.request {

LGCoordinatorID,

LGMemberOUTID

}

Table 5 defines the parameters of this primitive.

6.2.6.2 When generated

This occurs by CIP entity from the application layer in the LGMemberOUTID node to request a membership cancellation of a logical group which is coordinated by the LGCoordinatorID node.

6.2.6.3 Effect of receipt

On receipt of this primitive, the LGCoordinatorID node deletes the LGMemberOUTID from the membership table of the logical group. Correspondingly, the current member number is decreased by 1.

6.2.7 LG-MEMBEROUT.confirm

This primitive confirms result of the request of a membership cancellation of a logical group to CIP entity in the application layer.

6.2.7.1 Definition of service primitives

The syntax of this primitive is:

LG-MEMBERIN.confirm {

LGCoordinatorID,

LGMemberOUTID,

LGResultCode

}

Table 5 defines the parameters of this primitive.

6.2.7.2 When generated

This primitive reports a result of membership cancellation request to a logical group.

6.2.7.3 Effect of receipt

If the LGResultCode is SUCCESS, it means that the LGMemberINID node successfully quits a logical group coordinated by the LGCoordinatorID node. Otherwise, a FAILED error is confirmed.

6.2.8 LG-DISMISS.request

This primitive requests the dismissal of a logical group from the application layer.

6.2.8.1 Definition of service primitives

The syntax of this primitive is:

LG-DISMISS.request {

LGRequestorID,

LGCoordinatorID

}

Table 5 defines the parameters of this primitive.

6.2.8.2 When generated

This occurs by CIP entity from the application layer to request dismissal of a logical group in the LGCoordinatorID node.

6.2.8.3 Effect of receipt

On receipt of this primitive, the LGCoordinatorID node frees memory for the membership table and the attribute variables of current logical group. Once the operation is successful, the LGCoordinatorID node flags itself as a non-coordinator node and hence can be acted as a new coordinator upon a request of new group establishment. The memberships of this node to other multiple logical groups are not affected.

6.2.9 LG-DISMISS.indication

This primitive indicates the logical group dismissal from the service layer to the local CIP entity.

6.2.9.1 Definition of service primitives

The syntax of this primitive is:

LG-DISMISS.indication {

LGRequestorID,

LGCoordinatorID

}

Table 5 defines the parameters of this primitive.

6.2.9.2 When generated

This occurs when the service layer indicates dismissal of a logical group to CIP entity in the LGCoordinatorID node.

6.2.9.3 Effect of receipt

On receipt of this primitive, the CIP entity in the LGCoordinatorID node is indicated the dismissal of a logical group.

6.2.10 LG-DISMISS.confirm

This primitive confirms a logical group dismissal from the service layer to the CIP entity in the LGRequestorID node.

6.2.10.1 Definition of service primitives

The syntax of this primitive is:

LG-DISMISS.confirm {

LGRequestorID,

LGCoordinatorID,

LGResultCode

}

Table 5 defines the parameters of this primitive.

6.2.10.2 When generated

This primitive reports a result of logical group dismissal request.

6.2.10.3 Effect of receipt

If the LGResultCode is SUCCESS, it means a logical group coordinated by the LGCoordinatorID node has now be successfully dismissed. If the LGResultCode is FAILED, an error is indicated to the LGRequestorID node.

6.2.11 LG-QUERY.request

This primitive queries logical group attributes by CIP entities in the application layer.

6.2.11.1 Definition of service primitives

The syntax of this primitive is:

LG-QUERY.request {

LGRequestorID,

LGCoordinatorID,

LGAttributeNum,

LGAttribute

}

Table 5 defines the parameters of this primitive.

6.2.11.2 When generated

This occurs by CIP entity from the application layer in the LGRequestorID node to query an attribute of a logical group which is coordinated by the LGCoordinatorID node.

6.2.11.3 Effect of receipt

On receipt of this primitive, the LGCoordinatorID node queries LGAttributeNum attributes of current logical group. The attribute names and values are structured by LGAttribute.

6.2.12 LG-QUERY.confirm

This primitive returns the results of logical group attributes to CIP entity in the application layer.

6.2.12.1 Definition of service primitives

The syntax of this primitive is:

LG-MEMBERIN.confirm {

LGRequestorID,

LGCoordinatorID,

LGAttributeNum,

LGAttribute,

LGResultCode

}

Table 5 defines the parameters of this primitive.

6.2.12.2 When generated

This primitive returns the result of querying attribute of a logical group coordinated by the LGCoordinatorID node.

6.2.12.3 Effect of receipt

If the LGResultCode is SUCCESS, it means that LGAttributeNum attributes are successfully returned in the LGAttribute parameter. Otherwise, a FAILED error is confirmed.

6.3 Data synchronization and registration service

In this subclause, data synchronization and registration service in the service layer is defined. Data synchronization and registration service is provided through SYNREG-SAP. The SYNREG-SAP is the logical interface between data synchronization and registration service entity in the service layer and CIP entity in the application layer. This logical interface incorporates a set of primitives and their definitions. These primitives and definitions are described conceptually here, but through this the process of the parameters exchanged between the service layer and the application layer can be understood. Table 6 lists the primitives supported by the SYNREG-SAP. Table 7 outlines primitive parameters.

Table 6 — SYNREG-SAP primitive summary

Name	Request	Indication	Response	Confirm
SYNREG-SYNQUERY	6.3.1			6.3.2
SYNREG-SYNEXEC	6.3.3	6.3.4		6.3.5
SYNREG-REGQUERY	6.3.6			6.3.7
SYNREG-REGEXEC	6.3.8	6.3.9		6.3.10

Table 7 — SYNREG-SAP primitive parameters

Name	Type	Valid range	Description
SYNREGSrcID	Integer	0x0000-0xFFFF	Source node ID of data synchronization and registration service.
SYNREGDstID	Integer	0x0000-0xFFFF	Destination node ID of data synchronization and registration service.
SYNTimeRef	Double	Implementation specific	Value of time reference in data synchronization process. Return from destination node.
SYNExecVal	Double	Implementation specific	Value for data synchronization process in destination node.
REGRef		Implementation specific	Data structure of name and value of specific

			reference attribute in data registration process. Return from destination node.
REGRefDimension	Integer	0x00-0xFF	The dimensions of REGRef data for data registration process.
REGExecVal	Variable length octets	Implementation specific	Multi-dimensional data for data registration process in destination node.
SYNREGResultCode	Enumeration	SUCCESS, INVALID_REF_NAME, FAILED	Result code of data synchronization and registration service.

6.3.1 SYNREG-SYNQUERY.request

This primitive queries values of reference time for data synchronization by CIP entities in the application layer.

6.3.1.1 Definition of service primitives

The syntax of this primitive is:

SYNREG-SYNQUERY.request {

SYNREGSrcID,

SYNREGDstID,

SYNTimeRef

}

Table 7 defines the parameters of this primitive.

6.3.1.2 When generated

This occurs by CIP entity from the application layer to query value of reference time from the SYNREGDstID node which is required by data synchronization process in the SYNREGSrcID node.

6.3.1.3 Effect of receipt

On receipt of this primitive, the SYNREGDstID node gets current reference time for generating sensory data, which is returned with SYNTimeRef.

6.3.2 SYNREG-SYNQUERY.confirm

This primitive returns the values of reference time to CIP entity in the application layer.

6.3.2.1 Definition of service primitives

The syntax of this primitive is:

SYNREG-SYNQUERY.confirm {

SYNREGSrcID,

SYNREGDstID,

SYNTimeRef,

SYNREGResultCode

}

Table 7 defines the parameters of this primitive.

6.3.2.2 When generated

This primitive returns the result of querying values of reference time from the SYNREGDstID node.

6.3.2.3 Effect of receipt

If the SYNREGResultCode is SUCCESS, it means that the value of reference time for generating sensory data is successfully returned in the SYNTimeRef parameter. Otherwise, a FAILED error is confirmed.

6.3.3 SYNREG-SYNEXEC.request

This primitive requests the execution of data synchronization by CIP entity in the application layer.

6.3.3.1 Definition of service primitives

The syntax of this primitive is:

SYNREG-SYNEXEC.request {

SYNREGSrcID,

SYNREGDstID,

SYNExecVal

}

Table 7 defines the parameters of this primitive.

6.3.3.2 When generated

This occurs by CIP entity from the application layer in the SYNREGSrcID node to request the execution of data synchronization in the SYNREGDstID node.

6.3.3.3 Effect of receipt

On receipt of this primitive, the SYNREGDstID node executes data synchronization process locally, in which SYNExecVal is used to synchronize local reference time for generating sensory data in the SYNREGSrcID node.

6.3.4 SYNREG-SYNEXEC.indication

This primitive indicates the execution of data synchronization process from the service layer to the local CIP entity.

6.3.4.1 Definition of service primitives

The syntax of this primitive is:

SYNREG-SYNEXEC.indication {

SYNREGSrcID,

SYNREGDstID,

SYNExecVal

}

Table 7 defines the parameters of this primitive.

6.3.4.2 When generated

This occurs when the service layer indicates the execution of data synchronization process to CIP entity.

6.3.4.3 Effect of receipt

On receipt of this primitive, the CIP entity is indicated the execution of data synchronization process.

6.3.5 SYNREG-SYNEXEC.confirm

This primitive confirms the execution of data synchronization from the service layer to the CIP entity.

6.3.5.1 Definition of service primitives

The syntax of this primitive is:

SYNREG-SYNEXEC.confirm {

SYNREGSrcID,

SYNREGDstID,

SYNREGResultCode

}

Table 7 defines the parameters of this primitive.

6.3.5.2 When generated

This primitive reports execution result of data synchronization process to the CIP entity.

6.3.5.3 Effect of receipt

If the SYNREGResultCode is SUCCESS, it means that the reference time for generating sensory data in the SYNREGDstID node is successfully synchronized with the reference time in the SYNREGSrcID node. If the SYNREGResultCode is FAILED, an error is indicated to the SYNREGSrcID node.

6.3.6 SYNREG-REGQUERY.request

This primitive queries reference attributes for data registration process by CIP entities in the application layer.

6.3.6.1 Definition of service primitives

The syntax of this primitive is:

SYNREG-REGQUERY.request {

SYNREGSrcID,

SYNREGDstID,

REGRef

}

Table 7 defines the parameters of this primitive.

6.3.6.2 When generated

This occurs by CIP entity from the application layer to query value of the REGRef attribute from the SYNREGDstID node which is required by data registration process in the SYNREGSrcID node.

6.3.6.3 Effect of receipt

On receipt of this primitive, the SYNREGSrcID node requests the value of the REGRef attribute in the SYNREGDstID node.

6.3.7 SYNREG-REGQUERY.confirm

This primitive returns a result of querying value of the REGRef attribute to CIP entity in the application layer.

6.3.7.1 Definition of service primitives

The syntax of this primitive is:

SYNREG-REGQUERY.confirm {

SYNREGSrcID,

SYNREGDstID,

REGRef,

SYNREGResultCode

}

Table 7 defines the parameters of this primitive.

6.3.7.2 When generated

This primitive returns the result of querying REGRef attribute value in the SYNREGDstID node.

6.3.7.3 Effect of receipt

If the SYNREGResultCode is SUCCESS, it means that the value of the REGRef attribute is successfully returned in the REGRef parameter. If the SYNREGResultCode is INVALID_REF_NAME, it means that the REGRef attribute is not valid in the SYNREGDstID node. Otherwise, a FAILED error is confirmed.

6.3.8 SYNREG-REGEXEC.request

This primitive requests the execution of data registration by CIP entity in the application layer.

6.3.8.1 Definition of service primitives

The syntax of this primitive is:

SYNREG-REGEXEC.request {

SYNREGSrcID,

SYNREGDstID,

REGRefDimension,

REGExecVal

}

Table 7 defines the parameters of this primitive.

6.3.8.2 When generated

This occurs by CIP entity from the application layer in the SYNREGSrcID node to request the execution of data registration in the SYNREGDstID node.

6.3.8.3 Effect of receipt

On receipt of this primitive, the SYNREGDstID node executes data registration process locally, in which REGExecVal is used to perform data registration. The dimension of REGExecVal is specified by REGRefDimension.

6.3.9 SYNREG-REGEXEC.indication

This primitive indicates the execution of data registration process from the service layer to the local CIP entity.

6.3.9.1 Definition of service primitives

The syntax of this primitive is:

SYNREG-REGEXEC.indication {

SYNREGSrcID,

SYNREGDstID

}

Table 7 defines the parameters of this primitive.

6.3.9.2 When generated

This occurs when the service layer indicates the execution of data registration process to CIP entity in the SYNREGDstID node.

6.3.9.3 Effect of receipt

On receipt of this primitive, the CIP entity in the SYNREGDstID node is indicated the execution of data registration process.

6.3.10 SYNREG-REGEXEC.confirm

This primitive confirms the execution of data registration from the service layer to the CIP entity.

6.3.10.1 Definition of service primitives

The syntax of this primitive is:

SYNREG-REGEXEC.confirm {

SYNREGSrcID,

SYNREGDstID,

SYNREGResultCode

}

Table 7 defines the parameters of this primitive.

6.3.10.2 When generated

This primitive reports execution result of data registration process to the CIP entity in the SYNREGSrcID node.

6.3.10.3 Effect of receipt

If the SYNREGResultCode is SUCCESS, it means that the data registration process is successfully executed within the SYNREGDstID node. If the SYNREGResultCode is FAILED, an error is indicated to the SYNREGSrcID node.

6.4 Information description service

In this subclause, information description service in the service layer is defined. Information description service is provided through INFO-SAP. The INFO-SAP is the logical interface between information description service entity in the service layer and CIP entity in the application layer. This logical interface incorporates a set of primitives and their definitions. These primitives and definitions are described conceptually here, but through this the process of the parameters exchanged between the service layer and the application layer can be understood. Table 8 lists the primitives supported by the INFO-SAP. Table 9 outlines primitive parameters.

Table 8 — INFO-SAP primitive summary

Name	Request	Indication	Response	Confirm
INFO-LEVELGET	6.4.1			6.4.2
INFO-LEVELSET	6.4.3	6.4.4		6.4.5
INFO-DATA	6.4.6	6.4.7		6.4.8

Table 9 — INFO-SAP primitive parameters

Name	Type	Valid range	Description
InfoSrcID	Integer	0x0000-0xFFFF	Source node ID of information description

			service.
InfoDstID	Integer	0x0000-0xFFFF	Destination node ID of information description service.
LevelVal	Enumeration	RAW_LEVEL, FEATURE_LEVEL, DECISION_LEVEL	Difference information description levels.
InfoDataDimension	Integer	0x00-0xFF	The dimensions of InfoData.
InfoData	Variable length octets	Implementation specific	Multi-dimensional data which are transferred between source node and destination node.
InfoResultCode	Enumeration	SUCCESS, FAILED, LEVEL_NOT_SUPPORT	Result code of information description primitives.

6.4.1 INFO-LEVELGET.request

This primitive requests information description levels by CIP entities in the application layer.

6.4.1.1 Definition of service primitives

The syntax of this primitive is:

INFO-LEVELGET.request {

InfoSrcID,

InfoDstID,

LevelVal

}

Table 9 defines the parameters of this primitive.

6.4.1.2 When generated

This occurs by CIP entity from the application layer to query information description level of the InfoDstID node which is required by the InfoSrcID node.

6.4.1.3 Effect of receipt

On receipt of this primitive, the InfoDstID node gets current information description level.

6.4.2 INFO-LEVELGET.confirm

This primitive returns information description level to CIP entity in the application layer.

6.4.2.1 Definition of service primitives

The syntax of this primitive is:

INFO-LEVELGET.confirm {

InfoSrcID,

InfoDstID,

LevelVal,

InfoResultCode

}

Table 9 defines the parameters of this primitive.

6.4.2.2 When generated

This primitive returns the result of querying information description level to the InfoSrcID node.

6.4.2.3 Effect of receipt

If the InfoResultCode is SUCCESS, it means that the information description level of the InfoDstID node is successfully returned in the LevelVal parameter. Otherwise, a FAILED error is confirmed.

6.4.3 INFO-LEVELSET.request

This primitive sets the information description level by CIP entity in the application layer.

6.4.3.1 Definition of service primitives

The syntax of this primitive is:

INFO-LEVELSET.request {

InfoSrcID,

InfoDstID,

LevelVal

}

Table 9 defines the parameters of this primitive.

6.4.3.2 When generated

This occurs by CIP entity from the application layer in the InfoSrcID node to set the information description level of the InfoDstID node to LevelVal.

6.4.3.3 Effect of receipt

On receipt of this primitive, the InfoDstID node sets the information description level. Information description levels conceptually correspond to phases of information processing. Different levels feature different types, structures and length of information. Three information description levels are defined: RAW_LEVEL, FEATURE_LEVEL and DECISION_LEVEL. Once new information description level is set successfully, correspondent information processing procedures or algorithms may be trigger or be applied. A node may simultaneously support more than one information description levels.

6.4.4 INFO-LEVELSET.indication

This primitive indicates the set operation of information description levels from the service layer to the local CIP entity.

6.4.4.1 Definition of service primitives

The syntax of this primitive is:

INFO-LEVELSET.indication {

InfoSrcID,

InfoDstID,

LevelVal

}

Table 9 defines the parameters of this primitive.

6.4.4.2 When generated

This occurs when the service layer indicates the set operation of information description levels to CIP entity in the InfoDstID node.

6.4.4.3 Effect of receipt

On receipt of this primitive, the CIP entity in the InfoDstID node is indicated the set operation of information description levels in the service layer.

6.4.5 INFO-LEVELSET.confirm

This primitive confirms the set of information description levels from the service layer to the CIP entity.

6.4.5.1 Definition of service primitives

The syntax of this primitive is:

INFO-LEVELSET.confirm {

InfoSrcID,

InfoDstID,

LevelVal,

InfoResultCode

}

Table 9 defines the parameters of this primitive.

6.4.5.2 When generated

This primitive reports the result of setting information description levels to the CIP entity in the InfoSrcID node.

6.4.5.3 Effect of receipt

If the InfoResultCode is SUCCESS, it means that the information description level in the InfoDstID node is successfully set to LevelVal. If the InfoResultCode is LEVEL_NOT_SUPPORT, it means that the LevelVal information description level is not supported or cannot be provided by the InfoDstID node probably due to the capability of that node. If the InfoResultCode is FAILED, an error is indicated to the InfoSrcID node.

6.4.6 INFO-DATA.request

This primitive requests the transferring of information of specific information description levels by CIP entity in the application layer.

6.4.6.1 Definition of service primitives

The syntax of this primitive is:

INFO-DATA.request {

InfoSrcID,

InfoDstID,

LevelVal,

InfoDataDimension,

InfoData

}

Table 9 defines the parameters of this primitive.

6.4.6.2 When generated

This occurs by CIP entity from the application layer in the InfoSrcID node to request the transfer of information of specific information description level of the InfoDstID node.

6.4.6.3 Effect of receipt

On receipt of this primitive, the entity of the service layer in the InfoSrcID node sends the InfoData in LevelVal information description level to the peer entity of the service layer in the InfoDstID node. InfoDataDimension specifies the dimension of InfoData.

6.4.7 INFO-DATA.indication

This primitive indicates to the CIP entity in the application layer that data unit of specific information description level has been received.

6.4.7.1 Definition of service primitives

The syntax of this primitive is:

INFO-DATA.indication {

InfoSrcID,

InfoDstID,

LevelVal,

InfoDataDimension,

InfoData

}

Table 9 defines the parameters of this primitive.

6.4.7.2 When generated

This occurs when the service layer indicates the reception of data unit of specific information description level to CIP entity in the InfoDstID node.

6.4.7.3 Effect of receipt

On receipt of this primitive, the CIP entity in the InfoDstID node is indicated that data unit of LevelVal information description level has been received. InfoDataDimension specifies the dimension of InfoData.

6.4.8 INFO-DATA.confirm

This primitive confirms the transfer of data unit of specific information description level from the service layer to the CIP entity.

6.4.8.1 Definition of service primitives

The syntax of this primitive is:

INFO-DATA.confirm {

InfoSrcID,

InfoDstID,

LevelVal,

InfoResultCode

}

Table 9 defines the parameters of this primitive.

6.4.8.2 When generated

This primitive reports the result of transferring data unit of the LevelVal information description level to the CIP entity in the InfoSrcID node.

6.4.8.3 Effect of receipt

If the InfoResultCode is SUCCESS, it means that a data unit of the LevelVal information description level is successfully transferred from the InfoSrcID node to the InfoDstID node. Otherwise, an FAILED error is indicated to the InfoSrcID node.

6.5 Node-to-node inter-activation service

In this subclause, node-to-node inter-activation service in the service layer is defined. Node-to-node inter-activation is provided through N2NACT-SAP. The N2NACT-SAP is the logical interface between node-to-node inter-activation service entity in the service layer and CIP entity in the application layer. This logical interface incorporates a set of primitives and their definitions. These primitives and definitions are described conceptually here, but through this the process of the parameters exchanged between the service layer and the application layer can be understood. Table 10 lists the primitives supported by the N2NACT-SAP. Table 11 outlines primitive parameters.

Table 10 — N2NACT-SAP primitive summary

Name	Request	Indication	Response	Confirm
N2NACT	6.5.1			6.5.2

Table 11 — N2NACT-SAP primitive parameters

Name	Type	Valid range	Description
N2NSrcID	Integer	0x0000-0xFFFF	Source node ID of node-to-node inter-activation service.
N2NDstID	Integer	0x0000-0xFFFF	Destination node ID of node-to-node inter-activation service.
N2NACTMode	Enumeration	NORMAL, RESERVATION	Difference node-to-node inter-activation modes.
N2NACTDataAttributeNum	Integer	0x00-0xFF	The number of attributes used in node-to-node inter-activation process.
N2NACTDataAttribute	Variable length octets	Implementation specific	Data structure of attribute values for node-to-node inter-activation process.
N2NResultCode	Enumeration	SUCCESS, FAILED, MODE_NOT_SUPPORT	Result code of node-to-node inter-activation process.

6.5.1 N2NACT.request

This primitive implements the inter-activation process between two nodes which can be requested by CIP entity in the application layer.

6.5.1.1 Definition of service primitives

The syntax of this primitive is:

N2NACT.request {

N2NSrcID,

N2NDstID,

N2NACTMode,

N2NACTDataAttributeNum,

N2NACTDataAttribute

}

Table 11 defines the parameters of this primitive.

6.5.1.2 When generated

This occurs by CIP entity from the application layer to request activation in the N2NDstID node from the N2NSrcID node.

6.5.1.3 Effect of receipt

On receipt of this primitive, the entity of the service layer in the N2NSrcID node sends the activation request to the peer entity of the service layer in the N2NDstID node. If the N2NDstID node receives the request successfully, the activation process is executed in the N2NACTMode mode. Two different modes are supported in this service. One is named as NORMAL activation mode, in which the N2NDstID node triggers specific tasks or activate modules immediately using N2NACTDataAttribute included in this primitive once the request is received successfully. The other mode is RESERVATION mode. The activation process is reserved in the N2NDstID node and later be triggered automatically in a future time. The context data for activation process is given by N2NACTDataAttribute in the primitive. N2NACTDataAttributeNum specifies the number of attribute in N2NACTDataAttribute.

6.5.2 N2NACT.confirm

This primitive confirms the result of node-to-node inter-activation process from the service layer to the CIP entity.

6.5.2.1 Definition of service primitives

The syntax of this primitive is:

N2NACT.confirm {

N2NSrcID,

N2NDstID,

N2NACTMode,

N2NResultCode

}

Table 11 defines the parameters of this primitive.

6.5.2.2 When generated

This primitive reports the result of node-to-node inter-activation process to the CIP entity in the N2NSrcID node.

6.5.3.3 Effect of receipt

If the N2NResultCode is SUCCESS, it means that the N2NDstID node is successfully activated by the N2NSrcID node with the N2NACTMode mode. If N2NACTMode is NORMAL, then the N2NDstID node is already in active. If N2NACTMode is RESERVATION, the N2NDstID node will be activated automatically in a reserved future time. If the N2NResultCode is MODE_NOT_SUPPORT, it means that the N2NDstID node does not support reservation interactivation mode.

6.6 Parameter adaptation service

In this subclause, parameter adaptation service in the service layer is defined. Parameter adaptation service is provided through PAR-SAP. The PAR-SAP is the logical interface between parameter adaptation service entity in the service layer and CIP entity in the application layer. This logical interface incorporates a set of primitives and their definitions. These primitives and definitions are described conceptually here, but through this the process of the parameters exchanged between the service layer and the application layer can be understood. Table 12 lists the primitives supported by the PAR-SAP. Table 13 outlines primitive parameters.

Table 12 — PAR-SAP primitive summary

Name	Request	Indication	Response	Confirm
PAR	6.6.1	6.6.2		6.6.3

Table 13 — PAR-SAP primitive parameters

Name	Type	Valid range	Description
PARSrcID	Integer	0x0000-0xFFFF	Source node ID of parameter adaptation service.
PARDstID	Integer	0x0000-0xFFFF	Destination node ID of parameter adaptation service.
PARParameterName	String	Implementation specific	Parameter name in adaptation or query process.
PARParameterLength	Integer	0-65535	Length of PARParameter in parameter adaptation process.

PARParameter	Variable length octet	Implementation specific	Value for PARParameterName in parameter adaptation process.
PARResultCode	Enumeration	SUCCESS, FAILED, INVALID_PARAMETER	Result code of parameter adaptation service.

6.6.1 PAR.request

This primitive requests the parameter adaptation process by CIP entity in the application layer.

6.6.1.1 Definition of service primitives

The syntax of this primitive is:

PAR.request {

PARSrcID,

PARDstID,

PARParameterName,

PARParameterLength,

PARParameter

}

Table 13 defines the parameters of this primitive.

6.6.1.2 When generated

This occurs by CIP entity from the application layer in the PARSrcID node to request the parameter adaptation in the PARDstID node.

6.6.1.3 Effect of receipt

On receipt of this primitive, the entity of the service layer in the PARDstID node tries to retrieve its local parameter with the PARParameterName name. If the node successfully finds the PARParameterName parameter, the value of this parameter is updated with PARParameter. PARParameterLength specifies the length of PARParameter. Otherwise, an error code is generated.

6.6.2 PAR.indication

This primitive indicates to the CIP entity in the application layer that the request of parameter adaptation has been received.

6.6.2.1 Definition of service primitives

The syntax of this primitive is:

PAR.indication {

PARSrcID,

PARDstID,

PARParameterName,

}

Table 13 defines the parameters of this primitive.

6.6.2.2 When generated

This occurs when the service layer indicates the parameter adaptation process to CIP entity in the PARDstID node.

6.6.2.3 Effect of receipt

On receipt of this primitive, the CIP entity is indicated that the request of PARParameterName parameter adaptation has been received.

6.6.3 PAR.confirm

This primitive confirms the result of parameter adaptation from the service layer to the CIP entity.

6.6.3.1 Definition of service primitives

The syntax of this primitive is:

PAR.confirm {

PARSrcID,

PARDstID,

PARParameterName,

PARResultCode

}

Table 13 defines the parameters of this primitive.

6.6.3.2 When generated

This primitive reports the result of parameter adaptation process to the CIP entity in the PARSrcID node.

6.6.3.3 Effect of receipt

If the PARResultCode is SUCCESS, it means that the PARParameterName parameter in the PARDstID node is successfully updated. If the PARResultCode is INVALID_PARAMETER, it means that the PARParameterName parameter is not defined or valid in the PARDstID node. Otherwise, an FAILED error is indicated to the InfoSrcID node.

7 Enhanced Services and Interfaces Specifications

This clause specifies enhanced services (ES) supporting CIP in intelligent sensor networks. Service primitives and parameters of primitives are defined for each enhanced service. In Table 14, the names of service access points (SAPs) through which specific service is provided are listed.

Table 14 — Enhanced services and the names of SAPs

Service name	SAP name
QoS management service	QoS-SAP
CIP-driven scheduling service	SCHEDULING-SAP
Adaptive sensing service	ADSENSING-SAP

7.1 QoS management service

In this subclause, QoS management service in the service layer is defined. This service provides mechanisms to define and update QoS profiles, and to apply QoS profiles. QoS management service uses logical grouping service and parameter adaptation service. Figure 5 shows the relationship among QoS management service, logical grouping service and parameter adaptation service.

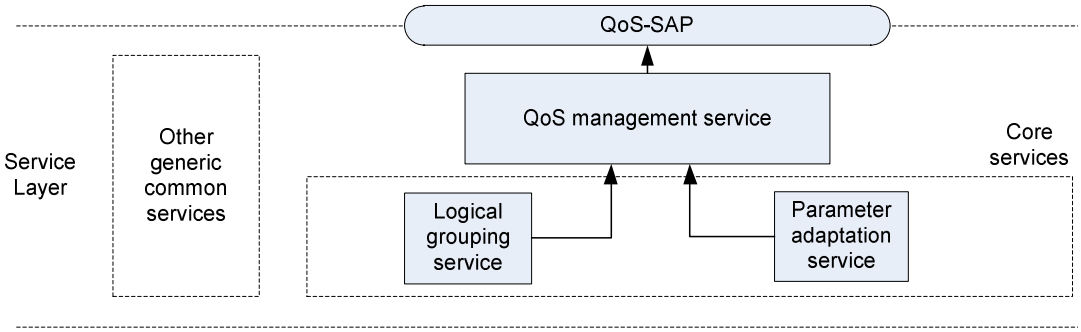

Figure 5 — Relationship among QoS management service and other services

QoS management service is provided through QoS-SAP. The QoS-SAP is the logical interface between QoS management entity in the service layer and CIP entity in the application layer. This logical interface incorporates a set of primitives and their definitions. These primitives and definitions are described conceptually here, but through this the process of the parameters exchanged between the service layer and the application layer can be understood. Table 15 lists the primitives supported by the QoS-SAP. Table 16 outlines primitive parameters.

Table 15 — QoS-SAP primitive summary

Name	Request	Indication	Response	Confirm
QoS-PROFILE-ESTABLISH	7.1.1	7.1.2		7.1.3
QoS-PROFILE-UPDATE	7.1.4			7.1.5
QoS-PROFILE-APPLY	7.1.6			7.1.7
QoS-PROFILE-DELETE	7.1.8	7.1.9		7.1.10

Table 16 — QoS-SAP primitive parameters

Name	Type	Valid range	Description
QoSRequestorID	Integer	0x0000-0xFFFF	Node ID of QoS management service requestor.
QoSProfileManagerID	Integer	0x0000-0xFFFF	Node ID of QoS profile manager.
QoSProfileID	Integer	0x00-0xFF	Profile ID for QoS management service.
QoSProfileUpdateMode	Integer	0x00-0xFF	The profile updating mode for QoS management service: 0x00 = Both logical group list and parameter list 0x01 = Logical group list only 0x02 = Parameter list only 0x03-0xFF = reserved
QoSProfileLGlist	Variable length octets	Implementation specific	Node ID list of coordinator for Logical groups in a QoS profile.
QoSProfilePARlist	Variable length octets	Implementation specific	Parameter name and value list for logical groups in a QoS profile.
QoSResultCode	Enumeration	SUCCESS, ERROR, SERVICE_NOT_SUPPORT, LG_UPDATE_FAILED, PARAMETER_UPDATE_FAILED, INVALID_PROFILE_ID, INVALID_MODE	Result code of QoS management service.

7.1.1 QoS-PROFILE-ESTABLISH.request

This primitive requests the establishment of a QoS Profile for QoS management from the application layer.

7.1.1.1 Definition of service primitives

The syntax of this primitive is:

QoS-PROFILE-ESTABLISH.request {

QoSRequestorID,

QoSProfileManagerID,

QoSProfileID

}

Table 16 defines the parameters of this primitive.

7.1.1.2 When generated

This occurs by CIP entity from the application layer in the QoSRequestorID node to request establishment of a QoS profile in the QoSProfileManagerID node.

7.1.1.3 Effect of receipt

On receipt of this primitive, the QoSProfileManagerID node first checks if QoS management service is supported by the node itself. If supported, a new record is created to record establishment of a new QoS profile. The new QoS profile will be identified by QoSProfileID.

7.1.2 QoS-PROFILE-ESTABLISH.indication

This primitive indicates the establishment of a QoS profile from the service layer to the application layer.

7.1.2.1 Definition of service primitives

The syntax of this primitive is:

QoS-PROFILE-ESTABLISH.indication {

QoSRequestorID,

QoSProfileManagerID,

QoSProfileID

}

Table 16 defines the parameters of this primitive.

7.1.2.2 When generated

This occurs by the QoS management service entity to indicate establishment of a QoS profile to CIP entity in the QoSProfileManagerID node.

7.1.2.3 Effect of receipt

On receipt of this primitive, the CIP entity in QoSProfileManagerID node is indicated establishment of a new QoS profile identified by QoSProfileID.

7.1.3 QoS-PROFILE-ESTABLISH.confirm

This primitive confirms a new QoS profile establishment from the service layer to the CIP entity in the application layer.

7.1.3.1 Definition of service primitives

The syntax of this primitive is:

QoS-PROFILE-ESTABLISH.confirm {

QoSRequestorID,

QoSProfileManagerID,

QoSProfileID,

QoSResultCode

}

Table 16 defines the parameters of this primitive.

7.1.3.2 When generated

This primitive reports a result of a QoS profile establishment request.

7.1.3.3 Effect of receipt

If the QoSResultCode is SUCCESS, it means a new QoS profile identified by QoSProfileID in the QoSProfileManagerID node is successfully established. If the QoSResultCode is INVALID_PROFILE_ID, it means that a QoS profile identified by QoSProfileID exists already in the QoSProfileManagerID node or the value of QoSProfileID is not valid. If the QoSResultCode is SERVICE_NOT_SUPPORT, the QoSProfileManagerID node does not support QoS management service. If the QoSResultCode is ERROR, an error is indicated to the QoSRequestorID node.

7.1.4 QoS-PROFILE-UPDATE.request

This primitive requests to update a QoS Profile for QoS management from the application layer.

7.1.4.1 Definition of service primitives

The syntax of this primitive is:

QoS-PROFILE-UPDATE.request {

QoSRequestorID,

QoSProfileManagerID,

QoSProfileID,

QoSProfileUpdateMode,

QoSProfileLGlist,

QoSProfilePARlist

}

Table 16 defines the parameters of this primitive.

7.1.4.2 When generated

This occurs by CIP entity from the application layer in the QoSRequestorID node to request to update a QoS profile identified by QoSProfileID in the QoSProfileManagerID node.

7.1.4.3 Effect of receipt

On receipt of this primitive, the QoSProfileManagerID node is requested to update the QoS profile identified by QosProfileID. The updating mode is defined by QoSProfileUpdateMode. If QoSProfileUpdateMode is 0x00, the record of QoSProfileID profile is updated with both QoSProfileLGlist and QoSProfilePARlist. If QoSProfileUpdateMode is 0x01, only logical group list in QoSProfileID profile record is updated with QoSProfileLGlist; The QoSProfilePARlist shall be NULL. If QoSProfileUpdateMode is 0x02, only parameter list in QoSProfileID profile record is updated with QoSProfilePARlist; The QoSProfileLGlist shall be NULL. Otherwise, QoS updating fails.

7.1.5 QoS-PROFILE-UPDATE.confirm

This primitive confirms the result of updating a QoS Profile to the application layer.

7.1.5.1 Definition of service primitives

The syntax of this primitive is:

QoS-PROFILE-UPDATE.confirm {

QoSRequestorID,

QoSProfileManagerID,

QoSProfileID,

QoSProfileUpdateMode,

QoSResultCode

}

Table 16 defines the parameters of this primitive.

7.1.5.2 When generated

This occurs by the QoS management service from the service layer to confirm the result of updating a QoS profile identified by QoSProfileID using QoSProfileUpdateMode mode in the QoSProfileManagerID node.

7.1.5.3 Effect of receipt

On receipt of this primitive, the QoSRequestorID node is confirmed the result of updating QoSProfileID profile in the QoSProfileManagerID node. If QoSResultCode is SUCCESS, the profile updating is successful. If QoSResultCode is LG_UPDATE_FAILED, it means logical group list in the QoSProfileID profile is not updated successfully. If QoSResultCode is PARAMETER_UPDATE_FAILED, it means parameter list updating in the QoSProfileID profile fails. If QoSResultCode is INVALID_MODE, it means that the value of QoSProfileUpdateMode is not valid. If the QoSResultCode is ERROR, an error is indicated to the QoSRequestorID node.

7.1.6 QoS-PROFILE-APPLY.request

This primitive requests to apply a QoS Profile for QoS management from the application layer.

7.1.6.1 Definition of service primitives

The syntax of this primitive is:

QoS-PROFILE-APPLY.request {

QoSRequestorID,

QoSProfileManagerID,

QoSProfileID

}

Table 16 defines the parameters of this primitive.

7.1.6.2 When generated

This occurs by CIP entity from the application layer in the QoSRequestorID node to request to apply a QoS profile identified by QoSProfileID.

7.1.6.3 Effect of receipt

On receipt of this primitive, the QoSProfileManagerID node is requested to apply the QoS profile identified by QosProfileID. When a QoS profile is applied, the QoSProfileManagerID should trigger a series of processes using logical grouping service and parameter adaptation service. All logical groups defined by QoSProfileLGlist shall update defined parameters with defined values specified by QoSProfilePARlist.

7.1.7 QoS-PROFILE-APPLY.confirm

This primitive confirms the result of applying a QoS Profile to the application layer.

7.1.7.1 Definition of service primitives

The syntax of this primitive is:

QoS-PROFILE-APPLY.confirm {

QoSRequestorID,

QoSProfileManagerID,

QoSProfileID,

QoSResultCode

}

Table 16 defines the parameters of this primitive.

7.1.7.2 When generated

This occurs by the QoS management service from the service layer to confirm the result of applying a QoS profile identified by QoSProfileID in the QoSProfileManagerID node.

7.1.7.3 Effect of receipt

On receipt of this primitive, the QoSRequestorID node is confirmed the result of applying QoSProfileID profile. If QoSResultCode is SUCCESS, the profile is applied successfully. Otherwise, an ERROR QoSResultCode is confirmed by QoS management service entities in the service layer.

7.1.8 QoS-PROFILE-DELETE.request

This primitive requests to delete a QoS Profile for QoS management from the application layer.

7.1.8.1 Definition of service primitives

The syntax of this primitive is:

QoS-PROFILE-DELETE.request {

QoSRequestorID,

QoSProfileManagerID,

QoSProfileID

}

Table 16 defines the parameters of this primitive.

7.1.8.2 When generated

This occurs by CIP entity from the application layer in the QoSRequestorID node to request deletion of a QoS profile in the QoSProfileManagerID node.

7.1.8.3 Effect of receipt

On receipt of this primitive, the QoSProfileManagerID node first tries to find a QoS profile record identified by QoSProfileID. If the QoSProfileID profile is found successfully, the QoS profile record is then deleted and the memory is released.

7.1.9 QoS-PROFILE-DELETE.indication

This primitive indicates the deletion of a QoS profile from the service layer to the application layer.

7.1.9.1 Definition of service primitives

The syntax of this primitive is:

QoS-PROFILE-DELETE.indication {

QoSRequestorID,

QoSProfileManagerID,

QoSProfileID

}

Table 16 defines the parameters of this primitive.

7.1.9.2 When generated

This occurs by the QoS management service entity to indicate a deletion of a QoS profile to CIP entity in the QoSProfileManagerID node.

7.1.9.3 Effect of receipt

On receipt of this primitive, the CIP entity in QoSProfileManagerID node is indicated the deletion of a QoS profile identified by QoSProfileID.

7.1.10 QoS-PROFILE-DELETE.confirm

This primitive confirms a QoS profile deletion from the service layer to the CIP entity in the application layer.

7.1.10.1 Definition of service primitives

The syntax of this primitive is:

QoS-PROFILE-DELETE.confirm {

QoSRequestorID,

QoSProfileManagerID,

QoSProfileID,

QoSResultCode

}

Table 16 defines the parameters of this primitive.

7.1.10.2 When generated

This primitive reports a result of a QoS profile deletion request.

7.1.10.3 Effect of receipt

If the QoSResultCode is SUCCESS, it means the QoS profile identified by QoSProfileID in the QoSProfileManagerID node is successfully deleted. If the QoSResultCode is INVALID_PROFILE_ID, it means that a QoS profile identified by QoSProfileID is not found in the QoSProfileManagerID node or the value of QoSProfileID is not valid. If the QoSResultCode is ERROR, an error is indicated to the QoSRequestorID node.

7.2 CIP-driven scheduling service

In this subclause, CIP-driven scheduling service in the service layer is defined. This service provides functions to control and schedule node states upon the request of CIP entities instead of node management entities in intelligent sensor networks. CIP-driven scheduling service uses event service, logical grouping service, parameter adaptation service and node-to-node inter-activation service, and other generic common services including neighbour finding service. Figure 6 shows the relationship among CIP-driven scheduling service and other services.

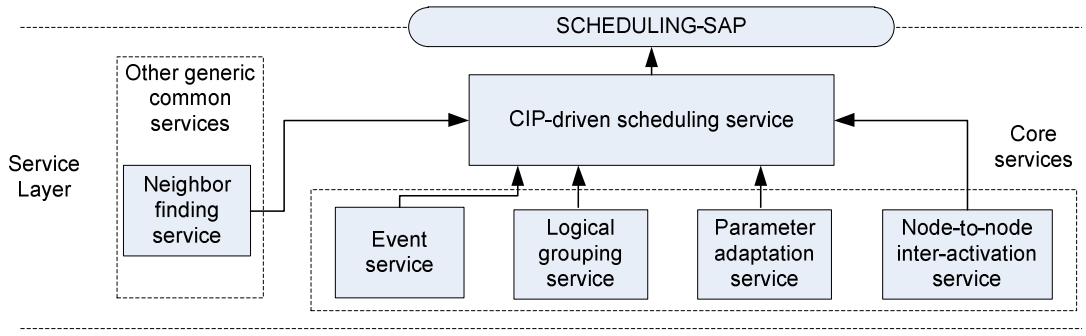

Figure 6 — Relationship among CIP-driven scheduling service and other services

CIP-driven scheduling service is provided through SCHEDULING-SAP. The SCHEDULING-SAP is the logical interface between CIP-driven scheduling service in the service layer and CIP entity in the application layer. This logical interface incorporates a set of primitives and their definitions. These primitives and definitions are described conceptually here, but through this the process of the parameters exchanged between the service layer and the application layer can be understood. Table 17 lists the primitives supported by the SCHEDULING-SAP. Table 18 outlines primitive parameters.

Table 17 — SCHEDULING-SAP primitive summary

Name	Request	Indication	Response	Confirm
SCHEDULING-SCHEME-ESTABLISH	7.2.1	7.2.2		7.2.3
SCHEDULING-SCHEME-UPDATE	7.2.4			7.2.5
SCHEDULING-SCHEME-APPLY	7.2.6			7.2.7
SCHEDULING-SCHEME-DELETE	7.2.8	7.2.9		7.2.10

Table 18 —SCHEDULING-SAP primitive parameters

Name	Type	Valid range	Description
SCHEDULINGRequestorID	Integer	0x0000-0xFFFF	Node ID of CIP-driven scheduling service requestor.
SCHEDULINGSchemeManagerID	Integer	0x0000-0xFFFF	Node ID of scheme manager.
SCHEDULINGSchemeID	Integer	0x00-0xFF	Scheme ID for CIP-driven scheduling service.
SCHEDULINGMode	Integer	0x00-0xFF	The scheduling mode for CIP-driven scheduling service: 0x00 = Non-prediction based mode 0x01 = Prediction

			based mode 0x02-0xFF = reserved
SchemeEVSubNodelist	Variable length octets	Implementation specific	Event subscription node ID list for CIP-driven scheduling service.
SchemeEVSubTypelist	Variable length octets	Implementation specific	Event type list for each event subscription nodes.
SchemePARlist	Variable length octets	Implementation specific	Parameter name and value list for CIP-driven scheduling service.
SCHEDULINGResultCode	Enumeration	SUCCESS, ERROR, SERVICE_NOT_SUPPORT, PARAMETER_UPDATE_FAILED, INVALID_SCHEME_ID, INVALID_MODE	Result code of CIP-driven scheduling service.

7.2.1 SCHEDULING-SCHEME-ESTABLISH.request

This primitive requests the establishment of a scheme for CIP-driven scheduling service from the application layer.

7.2.1.1 Definition of service primitives

The syntax of this primitive is:

SCHEDULING-SCHEME-ESTABLISH.request {

SCHEDULINGRequestorID,

SCHEDULINGSchemeManagerID,

SCHEDULINGSchemeID

}

Table 18 defines the parameters of this primitive.

7.2.1.2 When generated

This occurs by CIP entity from the application layer in the SCHEDULINGRequestorID node to request establishment of a scheme in the SCHEDULINGSchemeManagerID node.

7.2.1.3 Effect of receipt

On receipt of this primitive, the SCHEDULINGSchemeManagerID node first checks if CIP-driven scheduling service is supported by the node itself. If supported, a new record is created to record establishment of a new scheduling scheme. The new scheme will be identified by SCHEDULINGSchemeID.

7.2.2 SCHEDULING-SCHEME-ESTABLISH.indication

This primitive indicates the establishment of a scheme for CIP-driven scheduling service from the service layer to the application layer.

7.2.2.1 Definition of service primitives

The syntax of this primitive is:

SCHEDULING-SCHEME-ESTABLISH.indication {

SCHEDULINGRequestorID,

SCHEDULINGSchemeManagerID,

SCHEDULINGSchemeID

}

Table 18 defines the parameters of this primitive.

7.2.2.2 When generated

This occurs by the CIP-driven scheduling service entity to indicate establishment of a scheme to CIP entity in the SCHEDULINGSchemeManagerID node.

7.2.2.3 Effect of receipt

On receipt of this primitive, the CIP entity in the SCHEDULINGSchemeManagerID node is indicated establishment of a new scheme identified by SCHEDULINGSchemeID.

7.2.3 SCHEDULING-SCHEME-ESTABLISH.confirm

This primitive confirms a new scheme for CIP-driven scheduling service from the service layer to the CIP entity in the application layer.

7.2.3.1 Definition of service primitives

The syntax of this primitive is:

SCHEDULING_SCHEME-ESTABLISH.confirm {

SCHEDULINGRequestorID,

SCHEDULINGSchemeManagerID,

SCHEDULINGSchemeID,

SCHEDULINGResultCode

}

Table 18 defines the parameters of this primitive.

7.2.3.2 When generated

This primitive reports a result of a scheme establishment request.

7.2.3.3 Effect of receipt

If the SCHEDULINGResultCode is SUCCESS, it means a new scheme identified by SCHEDULINGSchemeID in the SCHEDULINGSchemeManagerID node is successfully established. If the SCHEDULINGResultCode is INVALID_SCHEME_ID, it means that a scheme identified by SCHEDULINGSchemeID exists already in the SCHEDULINGSchemeManagerID node or the value of SCHEDULINGSchemeID is not valid. If the SCHEDULINGResultCode is SERVICE_NOT_SUPPORT, the SCHEDULINGSchemeManagerID node does not support CIP-driven scheduling service. If the SCHEDULINGResultCode is ERROR, an error is indicated to the SCHEDULINGRequestorID node.

7.2.4 SCHEDULING-SCHEME-UPDATE.request

This primitive requests to update a scheme for CIP-driven scheduling service from the application layer.

7.2.4.1 Definition of service primitives

The syntax of this primitive is:

SCHEDULING-SCHEME-UPDATE.request {

SCHEDULINGRequestorID,

SCHEDULINGSchemeManagerID,

SCHEDULINGSchemeID,

SCHEDULINGMode,

SchemeEVSubNodelist,

SchemeEVSubTypelist,

SchemePARlist

}

Table 18 defines the parameters of this primitive.

7.2.4.2 When generated

This occurs by CIP entity from the application layer in the SCHEDULINGRequestorID node to request to update a scheme identified by SCHEDULINGSchemeID in the SCHEDULINGSchemeManagerID node.

7.2.4.3 Effect of receipt

On receipt of this primitive, the SCHEDULINGSchemeManagerID node is requested to update the scheme identified by SCHEDULINGSchemeID. SCHEDULINGMode defines a mode of CIP-driven scheduling. If SCHEDULINGMode is 0x00, the CIP-driven scheduling operates in a non-prediction based mode. If SCHEDULINGMode is 0x01, the CIP-driven scheduling operates in a prediction based mode.

SchemeEVSubNodelist, SchemeEVSubTypelist and SchemePARlist define event subscription node list, event type list and parameter list in the scheme.

7.2.5 SCHEDULING-SCHEME-UPDATE.confirm

This primitive confirms the result of updating a scheme to the application layer.

7.2.5.1 Definition of service primitives

The syntax of this primitive is:

SCHEDULING-SCHEME-UPDATE.confirm {

SCHEDULINGRequestorID,

SCHEDULINGSchemeManagerID,

SCHEDULINGSchemeID,

SCHEDULINGResultCode

}

Table 18 defines the parameters of this primitive.

7.2.5.2 When generated

This occurs by the CIP-driven scheduling service from the service layer to confirm the result of updating a scheme identified by SCHEDULINGSchemeID in the SCHEDULINGSchemeManagerID node.

7.2.5.3 Effect of receipt

On receipt of this primitive, the SCHEDULINGRequestorID node is confirmed the result of updating SCHEDULINGSchemeID scheme in the SCHEDULINGSchemeManagerID node. If SCHEDULINGResultCode is SUCCESS, the scheme is updated successfully. If SCHEDULINGResultCode is PARAMETER_UPDATE_FAILED, it means that, at least one parameter in the SCHEDULINGSchemeID scheme is not updated successfully. If SCHEDULINGResultCode is INVALID_MODE, it means that the value of SCHEDULINGMode is not supported. If the SCHEDULINGResultCode is ERROR, an error is indicated to the SCHEDULINGRequestorID node.

7.2.6 SCHEDULING-SCHEME-APPLY.request

This primitive requests to apply a scheme for CIP-driven scheduling service from the application layer.

7.2.6.1 Definition of service primitives

The syntax of this primitive is:

SCHEDULING-SCHEME-APPLY.request {

SCHEDULINGRequestorID,

SCHEDULINGSchemeManagerID,

SCHEDULINGSchemeID

}

Table 18 defines the parameters of this primitive.

7.2.6.2 When generated

This occurs by CIP entity from the application layer in the SCHEDULINGRequestorID node to request to apply a scheme identified by SCHEDULINGSchemeID.

7.2.6.3 Effect of receipt

On receipt of this primitive, the SCHEDULINGSchemeManagerID node is requested to apply the scheme identified by SCHEDULINGSchemeID. When a scheduling scheme is applied, the SCHEDULINGSchemeManagerID should trigger a series of processes using event service, logical grouping service and parameter adaptation service and node-to-node inter-activation service, and other generic common services including neighbour finding service. Events indicated from event service may be succeeded by neighbour finding service. Logical grouping based on neighbour relationship is executed. According to different scheduling mode, node-to-node inter-activation process with different inter-activation mode is performed, followed by a parameter adaptation process via parameter adaptation service.

7.2.7 SCHEDULING-SCHEME-APPLY.confirm

This primitive confirms the result of applying a scheduling scheme to the application layer.

7.2.7.1 Definition of service primitives

The syntax of this primitive is:

SCHEDULING-SCHEME-APPLY.confirm {

SCHEDULINGRequestorID,

SCHEDULINGSchemeManagerID,

SCHEDULINGSchemeID,

SCHEDULINGResultCode

}

Table 18 defines the parameters of this primitive.

7.2.7.2 When generated

This occurs by the CIP-driven scheduling service entity from the service layer to confirm the result of applying a scheme identified by SCHEDULINGSchemeID in the SCHEDULINGSchemeManagerID node.

7.2.7.3 Effect of receipt

On receipt of this primitive, the SCHEDULINGRequestorID node is confirmed the result of applying SCHEDULINGSchemeID scheme. If SCHEDULINGResultCode is SUCCESS, the scheme is applied successfully. Otherwise, an ERROR SCHEDULINGResultCode is confirmed by CIP-driven scheduling service entities in the service layer.

7.2.8 SCHEDULING-SCHEME-DELETE.request

This primitive requests to delete a scheme for CIP-driven scheduling service from the application layer.

7.2.8.1 Definition of service primitives

The syntax of this primitive is:

SCHEDULING-SCHEME-DELETE.request {

SCHEDULINGRequestorID,

SCHEDULINGSchemeManagerID,

SCHEDULINGSchemeID

}

Table 18 defines the parameters of this primitive.

7.2.8.2 When generated

This occurs by CIP entity from the application layer in the SCHEDULINGRequestorID node to request deletion of a scheme in the SCHEDULINGSchemeManagerID node.

7.2.8.3 Effect of receipt

On receipt of this primitive, the SCHEDULINGSchemeManagerID node first tries to find a scheme identified by SCHEDULINGSchemeID. If the SCHEDULINGSchemeID scheme is found successfully, the scheme is then deleted and the memory is released.

7.2.9 SCHEDULING-SCHEME-DELETE.indication

This primitive indicates the deletion of a scheduling scheme from the service layer to the application layer.

7.2.9.1 Definition of service primitives

The syntax of this primitive is:

SCHEDULING-SCHEME-DELETE.indication {

SCHEDULINGRequestorID,

SCHEDULINGSchemeManagerID,

SCHEDULINGSchemeID

}

Table 18 defines the parameters of this primitive.

7.2.9.2 When generated

This occurs by the CIP-driven scheduling service entity to indicate a deletion of a scheme to CIP entity in the SCHEDULINGSchemeManagerID node.

7.2.9.3 Effect of receipt

On receipt of this primitive, the CIP entity in SCHEDULINGSchemeManagerID node is indicated the deletion of a scheme identified by SCHEDULINGSchemeID.

7.2.10 SCHEDULING-SCHEME-DELETE.confirm

This primitive confirms a scheme deletion from the service layer to the CIP entity in the application layer.

7.2.10.1 Definition of service primitives

The syntax of this primitive is:

SCHEDULING-SCHEME-DELETE.confirm {

SCHEDULINGRequestorID,

SCHEDULINGSchemeManagerID,

SCHEDULINGSchemeID,

SCHEDULINGResultCode

}

Table 18 defines the parameters of this primitive.

7.2.10.2 When generated

This primitive reports a result of a scheduling scheme deletion request.

7.2.10.3 Effect of receipt

If the SCHEDULINGResultCode is SUCCESS, it means the scheme identified by SCHEDULINGSchemeID in the SCHEDULINGSchemeManagerID node is successfully deleted. If the SCHEDULINGResultCode is INVALID_SCHEME_ID, it means that a scheme identified by SCHEDULINGSchemeID is not found in the SCHEDULINGSchemeManagerID node or the value of SCHEDULINGSchemeID is not valid. If the SCHEDULINGResultCode is ERROR, an error is indicated to the SCHEDULINGRequestorID node.

7.3 Adaptive sensing service

In this subclause, adaptive sensing service in the service layer is defined. This service provides mechanisms to adaptively apply sensing rules and mechanism according to different event occurrence and in different contexts in intelligent sensor network. Adaptive sensing service uses event service, parameter adaptation service and other generic common services including sensor configuration service. Figure 7 shows the relationship among adaptive sensing service and other service.

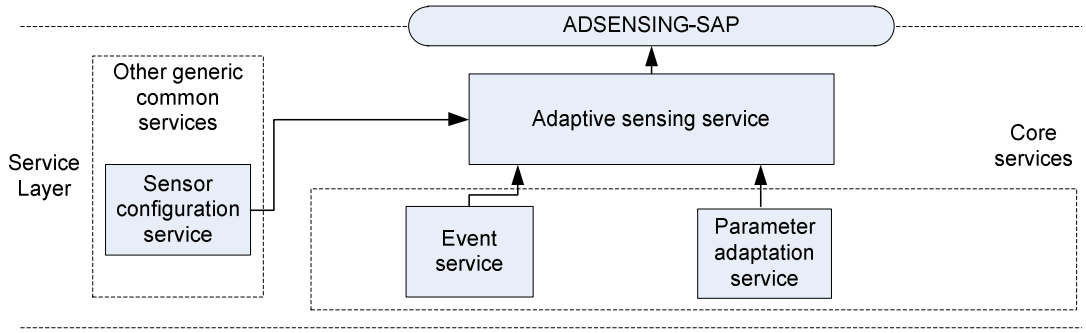

Figure 7 — Relationship among adaptive sensing service and other services

Adaptive sensing service is provided through ADSENSING-SAP. The ADSENSING-SAP is the logical interface between adaptive sensing service entity in the service layer and CIP entity in the application layer. This logical interface incorporates a set of primitives and their definitions. These primitives and definitions are described conceptually here, but through this the process of the parameters exchanged between the service layer and the application layer can be understood. Table 19 lists the primitives supported by the ADSENSING-SAP. Table 20 outlines primitive parameters.

Table 19 — ADSENSING-SAP primitive summary

Name	Request	Indication	Response	Confirm

ADSENSING-APPLY	7.3.1			7.3.2
ADSENSING-CANCEL	7.3.3			7.3.4

Table 20 —ADSENSING-SAP primitive parameters

Name	Type	Valid range	Description
ADSENSINGRequestorID	Integer	0x0000-0xFFFF	Node ID of adaptive sensing service requestor.
ADSENSINGTargetID	Integer	0x0000-0xFFFF	Target node ID of adaptive sensing service.
ADSENSINGTargetSensorID	Integer	0x00-0xFF	Sensor ID in target node.
ADSENSINGEVNodeList	Variable length octets	Implementation specific	Event node ID list for adaptive sensing service.
ADSENSINGEVTypeList	Variable length octets	Implementation specific	Event type list for each event nodes.
ADSENSINGSensorPARlist	Variable length octets	Implementation specific	Parameter name and value list to be used by sensor configuration in adaptive sensing service.
ADSENSINGResultCode	Enumeration	SUCCESS, ERROR, SERVICE_NOT_SUPPORT	Result code of adaptive sensing service.

7.3.1 ADSENSING-APPLY.request

This primitive requests the application of adaptive sensing mechanism from the application layer to the service layer.

7.3.1.1 Definition of service primitives

The syntax of this primitive is:

ADSENSING-APPLY.request {

ADSENSINGRequestorID,

ADSENSINGTargetID,

ADSENSINGTargetSensorID,

ADSENSINGEVNodeList,

ADSENSINGEVTypeList,

ADSENSINGSensorPARlist

}

Table 20 defines the parameters of this primitive.

7.3.1.2 When generated

This occurs by CIP entity from the application layer in the ADSENSINGRequestorID node to request application of an adaptive sensing mechanism in the ADSENSINGTargetID node.

7.2.1.3 Effect of receipt

On receipt of this primitive, the ADSENSINGTargetID node first checks if adaptive sensing service is supported by the node itself. If supported, configuration of sensor identified by ADSENSINGTargetSensorID in the ADSENSINGTargetID node is bound with events defined by ADSENSINGEVNodeList and ADSENSINGEVTypeList. ADSENSINGSensorPARlist provides parameters for sensor configuration.

7.3.2 ADSENSING-APPLY.confirm

This primitive confirms the result of requesting the application of adaptive sensing mechanism to the application layer.

7.3.2.1 Definition of service primitives

The syntax of this primitive is:

ADSENSING-APPLY.confirm {

ADSENSINGRequestorID,

ADSENSINGTargetID,

ADSENSINGTargetSensorID,

ADSENSINGResultCode

}

Table 20 defines the parameters of this primitive.

7.3.2.2 When generated

This occurs by the adaptive sensing service entity from the service layer to confirm the result of applying an adaptive sensing mechanism in the ADSENSINGTargetID node.

7.3.2.3 Effect of receipt

On receipt of this primitive, the ADSENSINGRequestorID node is confirmed the result of applying an adaptive sensing mechanism to the ADSENSINGTargetSensorID sensor. If ADSENSINGResultCode is SUCCESS, the mechanism is applied successfully. Otherwise, an ERROR ADSENSINGResultCode is confirmed by adaptive sensing service entities in the service layer.

7.3.3 ADSENSING-CANCEL.request

This primitive requests the cancellation of adaptive sensing mechanism from the application layer to the service layer.

7.3.3.1 Definition of service primitives

The syntax of this primitive is:

ADSENSING-CANCEL.request {

ADSENSINGRequestorID,

ADSENSINGTargetID,

ADSENSINGTargetSensorID

}

Table 20 defines the parameters of this primitive.

7.3.3.2 When generated

This occurs by CIP entity from the application layer in the ADSENSINGRequestorID node to request cancellation of an adaptive sensing mechanism in the ADSENSINGTargetID node.

7.3.3.3 Effect of receipt

On receipt of this primitive, all events currently bound with the ADSENSINGTargetSensorID sensor in the ADSENSINGTargetID node are unbound with the sensor.

7.3.4 ADSENSING-CANCEL.confirm

This primitive confirms the result of requesting the cancellation of adaptive sensing mechanism to the application layer.

7.3.4.1 Definition of service primitives

The syntax of this primitive is:

ADSENSING-CANCEL.confirm {

ADSENSINGRequestorID,

ADSENSINGTargetID,

ADSENSINGTargetSensorID,

ADSENSINGResultCode

}

Table 20 defines the parameters of this primitive.

7.3.4.2 When generated

This occurs by the adaptive sensing service entity from the service layer to confirm the result of cancelling an adaptive sensing mechanism in the ADSENSINGTargetID node.

7.3.4.3 Effect of receipt

On receipt of this primitive, the ADSENSINGRequestorID node is confirmed the result of cancelling an adaptive sensing mechanism to the ADSENSINGTargetSensorID sensor. If ADSENSINGResultCode is SUCCESS, the mechanism is cancelled successfully. Otherwise, an ERROR ADSENSINGResultCode is confirmed by adaptive sensing service entities in the service layer.

Annex – A Core Services and Interfaces Examples

This annex is informative. Examples of core services and interfaces in an intelligent sensor network based anti-intrusion system are given.

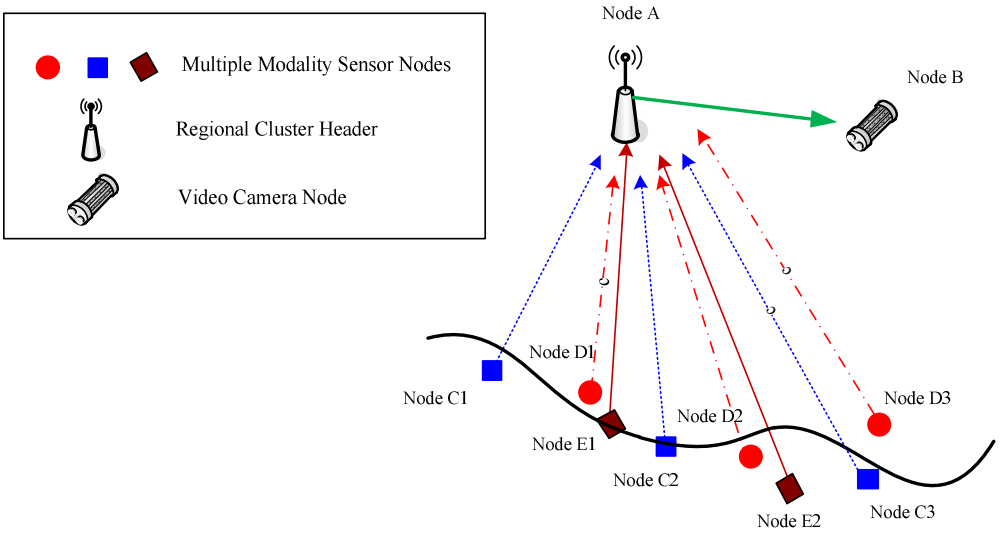

Figure 8 — Target detection and recognition based on core services supporting CIP

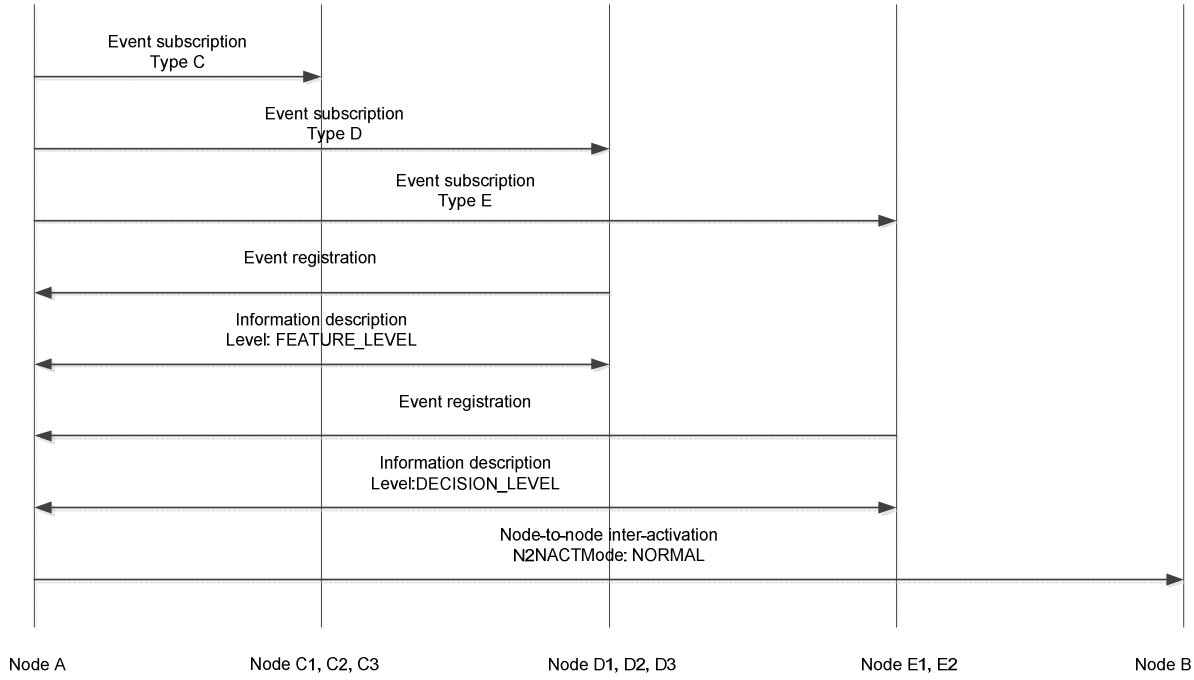

Figure 9 — Workflow based on core services

Target detection and recognition are very important information processing technologies for an anti-intrusion system. Usually, several kinds of sensors may be deployed. As shown in Figure 8, three different types of sensor nodes are deployed along the perimeter: Type C, Type D and Type E. When any of these sensor nodes detect a target, an event message will be generated and be forwarded to a regional cluster header (Node A in Figure 8). In order to recognize the target, Node A may request data or information from sensor

nodes deployed along the perimeter. The information description levels from these sensor nodes differ from each other. Upon the target recognition result, a video camera node (Node B in Figure 8) may be activated by Node A to get a further confirmation or evidence via an image capturing operation.

In order to support target detection and recognition, several core services shall be used, including event service, information description service and node-to-node inter-activation service. Figure 9 shows a workflow of using core services to support target detection and recognition.

Annex – B Enhanced Services and Interfaces Examples

This annex is informative. Examples of enhanced services and interfaces in an intelligent sensor network based anti-intrusion system are given.

Target tracking is another very important information processing technology for an anti-intrusion system. As shown in Figure 10, when the perimeter length is very long, hundreds of sensor nodes may be used to get a full converge along the perimeter. To improve system performance, the second virtual perimeter may be deployed. In Figure 10, there are three logical groups, which are coordinated respectively by Node A, Node B and Node F. A command & control node (Node G in Figure 10) works as a coordinator of Node A, Node B, and Node F.

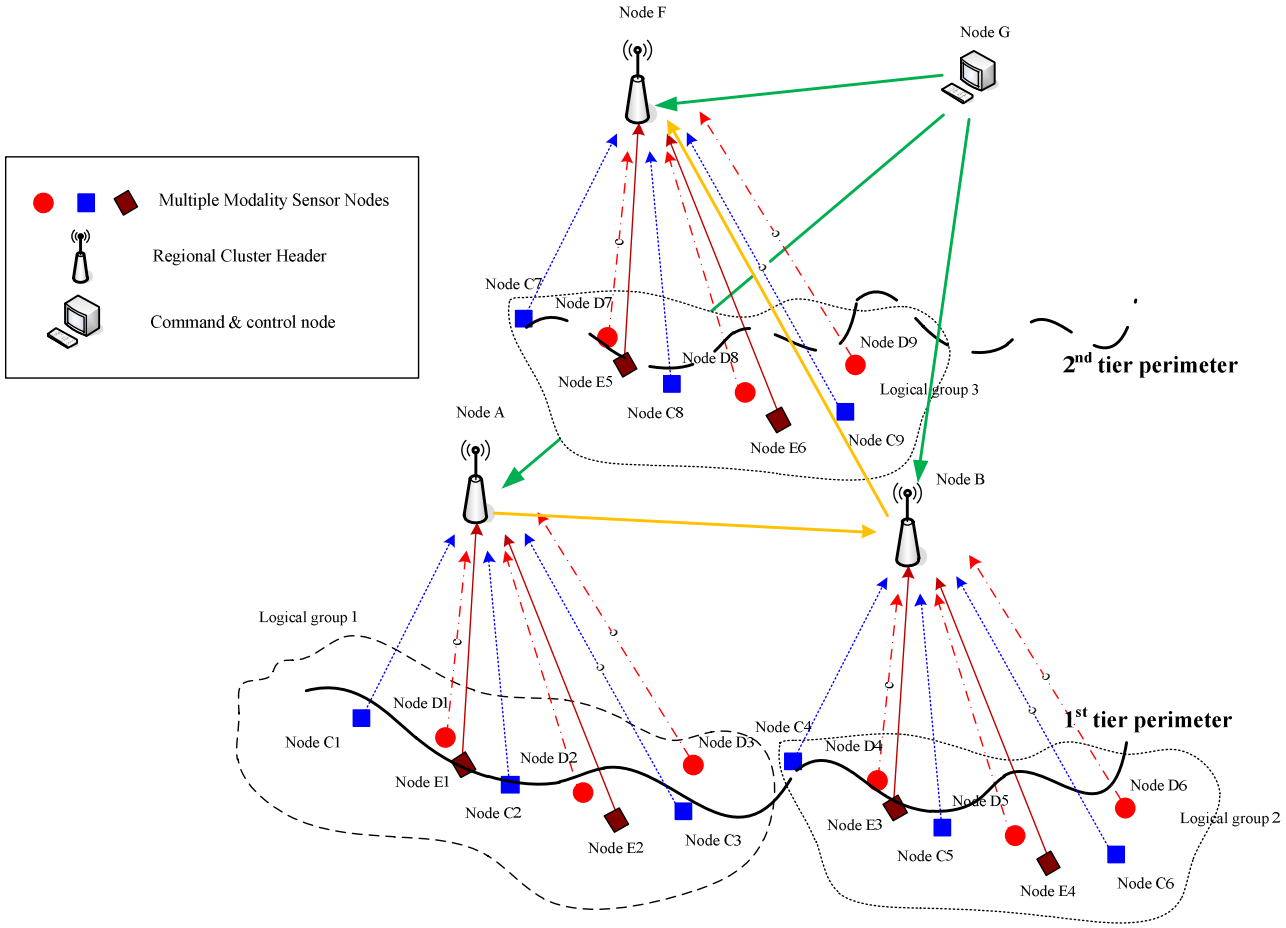

Figure 10 — Target tracking based on enhanced services supporting CIP

In implement target tracking, enhanced services supporting CIP including QoS management service and CIP-driven scheduling service shall be used. Figure 11 shows a workflow of using enhanced services to support target tracking in an anti-intrusion system.

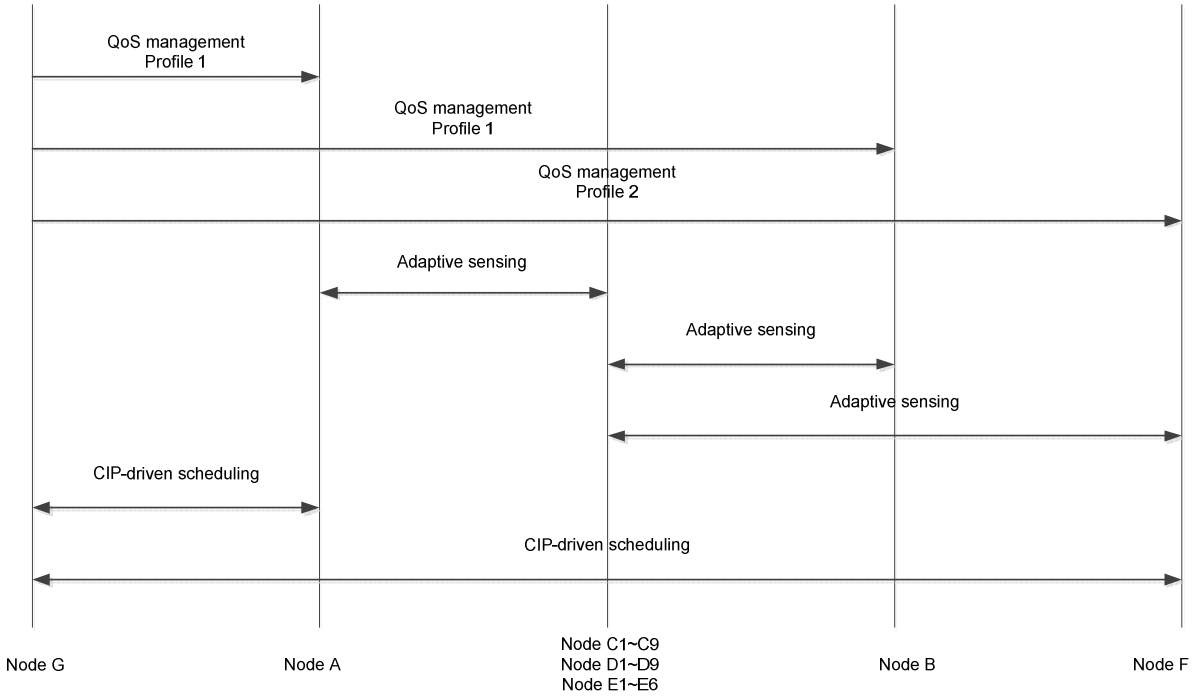

Figure 11 —Workflow based on enhanced services

Annex – C Bibliography

[1] ISO/IEC JTC1 WG7 N007, *Technical Document on Sensor Networks (Version 3)*, 2009

[2] ISO/IEC JTC1 WG7 N089, *1st Working Draft of ISO/IEC WD 29182-1, Information technology — Sensor Networks: Sensor Network Reference Architecture (SNRA) —Part 1: General overview and requirements*, 2010

[3] ISO/IEC JTC1 WG7 N090, *1st Working Draft of ISO/IEC WD 29182-6, Information technology — Sensor Networks: Sensor Network Reference Architecture (SNRA) —Part 6: Application Profiles*, 2010

[4] ISO/IEC JTC1 WG7 N091, *1st Working Draft of ISO/IEC WD 29182-7, Information technology —Sensor Networks: Sensor Network Reference Architecture (SNRA) —Part 7: Interoperability guidelines*, 2010

[5] ISO/IEC JTC1 WG7 N106, *PE's contribution on base document of 1st WD for ISO/IEC 29182-4, Information Technology — Sensor Networks: Sensor Network Reference Architecture (SNRA) — Part 4: Entity Models*, 2010

[6] ISO/IEC JTC1 WG7 N107, *NB of US contribution 1 on the ISO/IEC WD 29182, Sensor Network Reference Architecture - All parts 1 – 7*, 2010

[7] ISO/IEC JTC1 WG7 N108, *NB of US contribution 2 on the ISO/IEC 29182, Sensor Network Reference Architecture - All parts 1 – 7*, 2010

[8] ISO/IEC JTC1 WG7 N125, *1st Working Draft of ISO/IEC WD 29182-2, Information technology — Sensor Network Reference Architecture (SNRA) – Part 2: Vocabulary and Terminology*, 2010

[9] ISO/IEC JTC1 SC7 WG19 HYD-011, *ITU-T X.902 | ISO/IEC 10746-2 Information Technology – Open Distributed Processing – Reference Model – Foundations*, 2010

[10] ISO/IEC JTC1 SC7 WG19 HYD-015, *ITU-T X.903 | ISO/IEC 10746-3 Information Technology – Open Distributed Processing – Reference Model – Architecture*, 2010

[11] The SANY Consortium, *SANY: an open service architecture for sensor networks*, ISBN 978-3-00-028571-4, 2009

[12] Feng Zhao, Leonidas J. Guibas, *Wireless Sensor Networks: An Information Processing Approach*, ISBN 1-55860-914-8, 2004

[13] Open Geospatial Consortium Inc., OGC 09-138, *OGC Fusion Standards Study Engineering Report*, 2010

[14] Open Geospatial Consortium Inc., OGC 07-165, *OGC Sensor Web Enablement: Overview and High Level Architecture*, 2007

[15] Open Geospatial Consortium Inc., OGC 10-184, *OGC Fusion Standards Study, Phase 2 Engineering Report*, 2010

[16] Zigbee Alliance, Inc., Document 053474r17, *Zigbee Specification*, 2007

[17] D.L.Hall, J.Llinas, *Handbook of Multisensor Data fusion*, CRC Press, ISBN 0-8493-2379-7, 2001

[18] M.Ilyas, I.Mahgoub, Handbook of sensor networks: compact wireless and wired sensing systems, CRC Press, ISBN 0-8493-1968-4, 2005

[19] G.-Z.Yang, Body Sensor Networks, Springer-Verlag London Limited, ISBN 1-84628-272-1, 2006

[20] J.Yick, B.Mukherjee, D.Ghosal, *Wireless Sensor Network Survey*, Computer Networks, Vol.52, pp.2292-2330, 2008

[21] M.Wang, J.Cao, J.Li, S.Dasi, *Middleware for Wireless Sensor Networks: A Survey*, Journal of Computer Science and Technology, Vol.23, pp.305-325, 2008