INTERNATIONAL STANDARD

ISO/IEC 20005

First edition
2013-07-01

Information technology — Sensor networks — Services and interfaces supporting collaborative information processing in intelligent sensor networks

Technologies de l'information — Réseaux de capteurs — Services et interfaces prenant en charge le traitement d'information collaboratif dans les réseaux de capteurs intelligents

Reference number
ISO/IEC 20005:2013(E)

COPYRIGHT PROTECTED DOCUMENT

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

Published in Switzerland

Contents

Page

Foreword

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

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

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national 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.

ISO/IEC 20005 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

Introduction

Sensor networks have been widely deployed in different application domains including environment monitoring, transportation, manufacturing, chemical process, healthcare, home and buildings, and many other domains. Wired/wireless sensor networks can be regarded as an extension of the Internet interfacing the physical world. Intelligent sensor networks are increasingly attractive in a wide range of applications 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), which closely integrates information processing algorithms with collaboration mechanisms, is an essential technology enabling the intelligent sensor networks to enhance efficiency and to improve quality and reliability of information processing and its outputs in real application scenarios. This standard specifies services and interfaces supporting CIP in the 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 7498-1:1994, *Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model*

3 Terms and definitions

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

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

[SOURCE: ISO/IEC 29182-2]

3.2
collaborative information processing
form of information processing in which multiple sensor network elements collaborate, in order to enhance efficiency and improve the quality and reliability of the output

[SOURCE: ISO/IEC 29182-2]

3.3
data registration
process of transforming different sets of data into one coordinate system

3.4
data grouping
process of identifying a time interval common among different data sources and grouping data obtained in the time interval

3.5
event
anything that happens or is contemplated as happening at an instant or over an interval of time

3.6
sensor network
system of spatially distributed sensor nodes interacting with each other and, depending on applications, possibly with other infrastructure in order to acquire, process, transfer, and provide information extracted from its environment with a primary function of information gathering and possible control capability

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

[SOURCE: ISO/IEC 29182-2]

3.7
sensor network application
use case of sensor networks, which provide a set of functions to users to meet defined requirements

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

[SOURCE: ISO/IEC 29182-2]

3.8
sensor network service
set of functionalities offered by individual sensor network elements or the sensor network

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

[SOURCE: ISO/IEC 29182-2]

3.9
sensor node
sensor network element that includes at least one sensor and optionally actuators with communication capabilities and associated data processing capabilities

Note 1 to entry: It may include additional application capabilities.

[SOURCE: ISO/IEC 29182-2]

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

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 Systems Interconnection/Reference Model [ISO 7498-2:1989]

QoS Quality of Service

SAP Service Access Point

5 General description

5.1 Overview

A system composed of a sensor network or sensor networks attempts to fully integrate sensing, data/information transmission and processing and information provision processes to satisfy the system's application requirements for end users. Figure 1 shows a functional overview of sensor networks system from the layered architectural view.

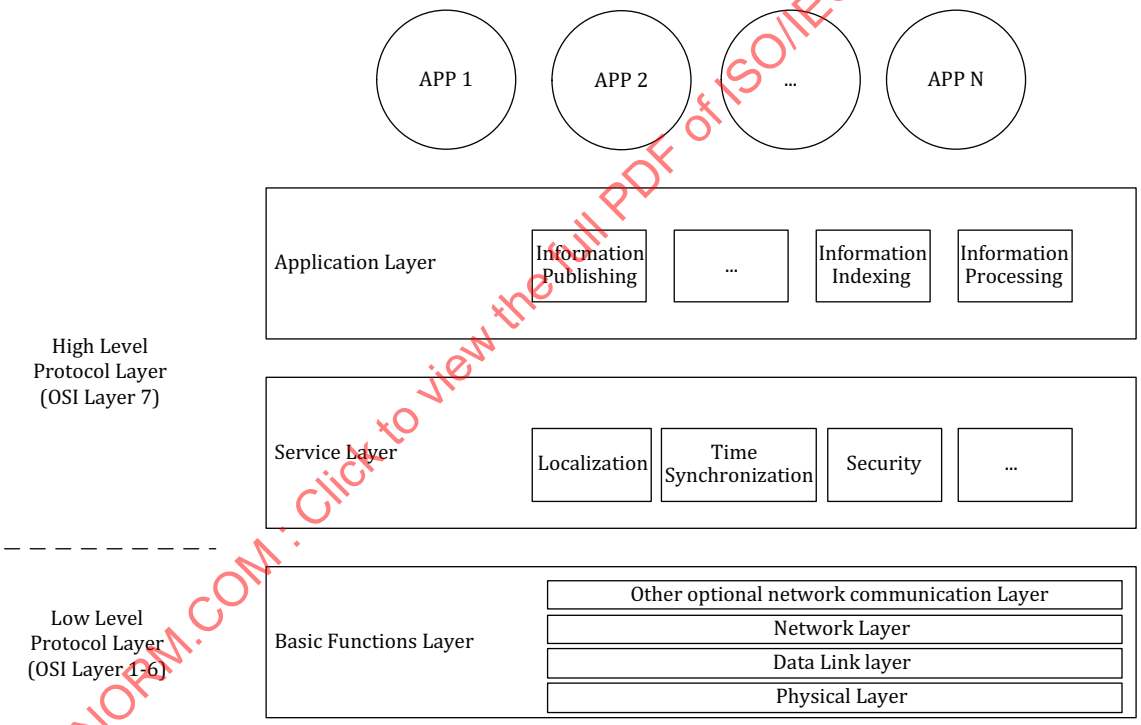

Figure 1 — Layer overview of sensor network system architecture

The Basic Functions Layer implements basic functionalities fulfilled by the lower layers in the Open Systems Interconnection Reference Model (OSI/RM in ISO/IEC 7498-1), including the physical layer, the data link layer, the network layer, and other optional network communication layers. Above the basic functions layer, there are the Application Layer and the Service Layer. The application layer provides services to individual applications and/or users and implements functions such as information publishing, information indexing and information processing, etc. Between the application layer and the basic functions layer, the service layer provides generic common services to entities in the application layer. Typical generic common services in the service layer include 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 that the intelligent sensor networks have to meet the challenges from intrinsic environment complexity, large orders of magnitude network scaling and dynamic application requirements.

— **Environmental self-adaptability:** An intelligent sensor network shall adapt to obtain required system performances if the physical environment of the sensor network's monitoring area changes. For example, an intelligent sensor network based anti-intrusion system should guarantee consistent system performances such as detection and false alarming rate (FAR) when the environment in which the sensor network is deployed changes.

— **Dynamic task supporting:** An intelligent sensor network shall support dynamic tasking including dynamic task assignment, dynamic task ordering by prioritization, dynamic service provisioning for information users/consumers, and dynamic quality of service (QoS) adjustment.

— **Autonomous system maintenance:** An intelligent sensor network shall autonomously maintain system functionalities in case of network scaling, node mobility, new node entrances, node exits, and node failures.

5.3 Overview of collaborative information processing

The key differences between traditional telecommunication infrastructures and information service systems based on sensor networks are that (1) sensor networks systems collect raw sensory data from physical world; and (2) from these data, extract application-specific information in order to obtain feature level data, decision level information, and knowledge about the physical world.

Integrated with metadata such as sensory information description, sensor identification, and sensory information location, CIP handles efficient resource management to provide the dynamic tasking to fulfil the requests demanded by information service consumers. Though different sensor network applications normally require application-specific services, the collaborative processing is an indispensable requirement for senor network based information service to handle constraints in power (e.g. batteries), computing resources, storage, and communication bandwidth. The collaborative processing also has to deal with technical challenges such as task dynamics, measurement uncertainty, node mobility, and environmental adaptation ability.

The aims of CIP in sensor networks are to improve system efficiency, enhance quality of service, and guarantee system performance. It provides efficient mechanisms such as majority-voting fusion, decision template fusion and statistical methods for handling incomplete and/or inaccurate information. It also provides protocols to meet challenges from intrinsic environment complexity, large orders of magnitude network scaling and dynamic application requirements.

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

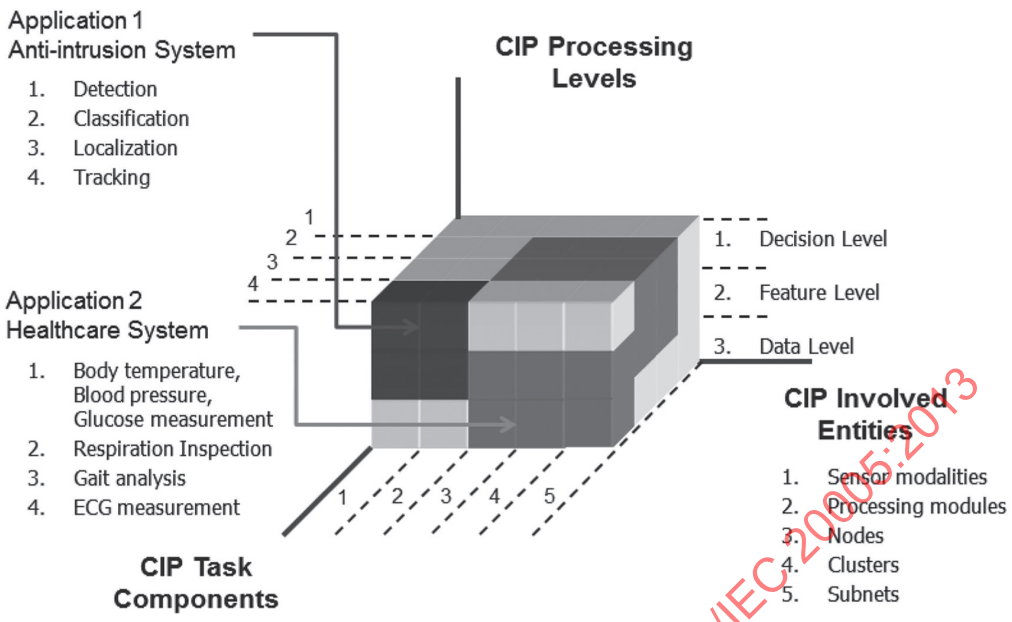

Figure 2 — Conceptual model of collaborative information processing

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 viewpoint. The task components in this viewpoint depend on the specific application scenarios of sensor networks. In anti-intrusion application system, the task components can be target detection, localization, classification, and tracking for security services. In healthcare system, task components may include blood pressure/temperature measurement, respiration inspection, and gait analysis. Figure 2 shows that tracking is one of the main CIP task components in Application 1. Decision-level and feature-level processing are applied, and both sensor modalities and processing modules are involved for CIP in the application. Specific selections and combinations using components 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 a functional entities point of views. In this model, CIP can be characterized by three distinct entities, which are named as capability declaration entity (CDE), collaborative strategy planning entity (CSPE) and communication requirement specification entity (CRSE).

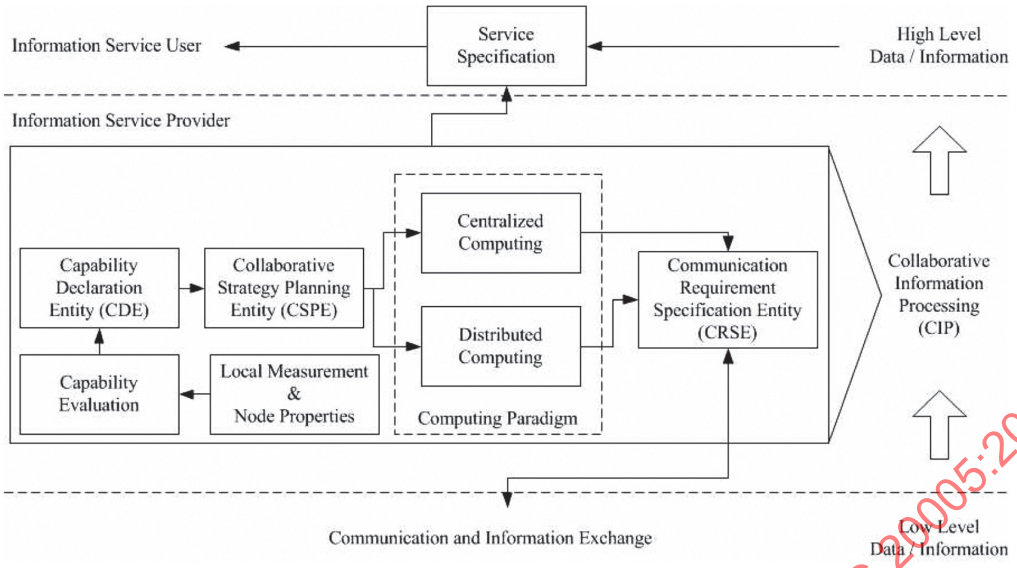

Figure 3 — Functional model of collaborative information processing

Capability Declaration Entity (CDE) declares capabilities of sensor nodes in a sensor network. Capabilities include not only individual node information such as sensing modality and its configuration, sensing range, remaining life on power, location, remaining storage capacity, and communication bandwidth, etc., but also certain characteristic information of sensory data collected by individual sensor node. One of the representative characteristics on sensory data are signal-to-noise ratio (SNR) value. Other characteristics include signal strength, estimated distance between a target (or targets) and sensor nodes, and state parameter prediction, etc. In other words, one sensor node should be qualified by the Capability Evaluation model to be a CIP participant before any actual CIP procedure is triggered. CDE requires a preliminary capability evaluation process which uses information of local measurement and node property.

Collaborative Strategy Planning Entity (CSPE) is probably the most important entity in CIP. CSPE uses available information provided by CDE and decides how collaborative information processing will be implemented. With certain cost functions or utility measures, CSPE tries to find a resource-efficient solution to collaborative strategy planning problem while the best information processing performance can also concurrently be achieved. 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. In distributed computing paradigm, there are several local computing/fusion centres. In centralized computing paradigm, only one central computing/fusion centre exists. The paradigm used is decided by the CSPE entity. Both the spatial and temporal correlation among different local centres, which may be dynamic, is also indicated by the CSPE entity.

Communication Requirement Specification Entity (CRSE) acts as interface between information service provider and Communication and Information Exchange. CRSE defines parameters and protocols to clearly describe requirements on communication and information exchange. For example requirements such as end-to-end delay, time jitter, bit error, and other QoS parameters should be specified.

5.5 Overview of services supporting CIP

Generic common services in the service layer could be divided into different subsets according to the types of service consumer entities in the application layer. This standard specifies a subset of generic common services which interface with the CIP entities in the application layer and support implementation of the 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 the CIP entities. Enhanced services are implemented through combining services, for example, integrating two or more core services or other generic common services provided in the service layer.

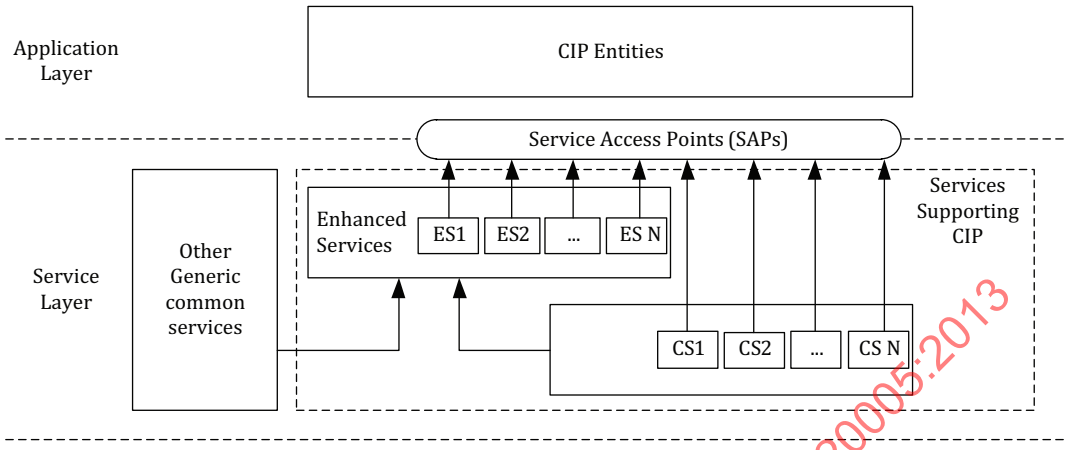

Figure 4 — Overview of services supporting CIP

5.5.1 Core services supporting CIP

The core services (CS) supporting CIP includes:

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

— **Logical grouping service:** This service implements functionalities concerning the establishment and management processes of logical group for the implementation of CIP in the application layer. The logical group is a logical set of sensor network entities involved in specific information processing tasks such as target detection, localization, recognition and target tracking. Logical grouping service provides mechanism for establishing collaborative relationship among the entities in intelligent sensor networks.

— **Data grouping service:** Data are generated by various sensors in different time intervals and scales. This service identifies or specifies a time interval common to all sensors participating in CIP, and groups all sensor data obtained during that time interval for processing. Data grouping service uses a time synchronization service to support CIP.

— **Data registration service:** Data generated from sensors in distributed sensor nodes may be described in different spatial reference coordinate systems. Data registration is a necessary process in order to transform or integrate different sets of data into one coordinate system. Based on reference coordinate system description, this service provides functionalities to keep the spatial reference coordinate system consistency among participators in CIP.

— **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 results from CIP processes.

— **Node-to-node inter-activation service:** This service provides mechanisms not only to initiate the execution of tasks in one sensor node commended by another sensor node, but also to trigger modules from one sensor node from another sensor 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 in deploying environment and application requirements.

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. The QoS management service uses the logical grouping service and the parameter adaptation service.

— **CIP-driven scheduling service:** This service provides functions to control and schedule node states upon the request of the 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 the event service, the logical grouping service, the parameter adaptation service and the 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 according to different event occurrence and different contexts in intelligent sensor networks. Adaptive sensing service can provide autonomous system maintenance and system adaptability in intelligent sensor network. Adaptive sensing service uses the event service, the information description service, and other generic common services including sensor configuration service.

6 Core services and interfaces specifications

6.1 Overview

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

Table 1 — Core services and the names of SAPs

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

6.2 Event service

Event service is provided through EVENT-SAP. The EVENT-SAP is the logical interface between the event service entity in the service layer and the 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 the primitive parameters.

Table 2 — EVENT-SAP primitive summary

Name	Request	Indication	Response	Confirm
EVENT-SUB	6.2.1	6.2.2		6.2.3
EVENT-REG		6.2.4		
EVENT-UNSUB	6.2.5			6.2.6

Table 3 — EVENT-SAP primitive parameters

Parameter Name	Description
EVSubSourceID	Source node ID of event subscription.
EVSubDestinationID	Destination node ID of event subscription.
EVSubModel	Event subscription models.
EVSubValue	Value for specific event subscription model.
EV_Time	A time indication for the event occurrence, as provided by the service layer. Return from destination node.
EVSubResultCode	Result code of event service operation.

6.2.1 EVENT-SUB.request

This primitive requests the process of event subscription from the application layer. The parameters of this primitive are:

EVENT-SUB.request {

EVSubSourceID,

EVSubDestinationID,

EVSubModel,

EVSubValue

}

Table 3 defines the parameters of this primitive.

This primitive is used by the CIP entity to subscribe the event service from the entity of the service layer. On receipt of this primitive, the entity providing the event service implements event subscription in the EVSubDestinationID node for the EVSubSourceID node.

6.2.2 EVENT-SUB.indication

This primitive indicates the event subscription from the service layer to CIP entity. The parameters of this primitive are:

EVENT-SUB.indication {

EVSubSourceID,

EVSubDestinationID,

EVSubModel,

EVSubValue

}

Table 3 defines the parameters of this primitive. This primitive is used when the service layer indicates an event subscription to the CIP entity. On receipt of this primitive, the CIP entity is indicated an event subscription.

6.2.3 EVENT-SUB.confirm

This primitive confirms an event subscription from the service layer to the CIP entity. The parameters of this primitive are:

EVENT-SUB.confirm {

EVSubSourceID,

EVSubDestinationID,

EVSubResultCode

}

Table 3 defines the parameters of this primitive. This primitive reports a result of the event subscription request. The EVSubResultCode parameter indicates a success if the event subscription is successful, otherwise an error is indicated.

6.2.4 EVENT-REG.indication

This primitive indicates the event occurrence from the service layer to the CIP entity. The parameters of this primitive are:

EVENT-REG.indication {

EVSubSourceID,

EVSubDestinationID,

EV_Time

}

Table 3 defines the parameters of this primitive. This primitive is used when the service layer indicates the event occurrences to the CIP entity. If one or more events occur or are detected, this primitive indication may be generated more than once. 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.

6.2.5 EVENT-UNSUB.request

This primitive requests the cancellation of event subscription from the application layer. The parameters of this primitive are:

EVENT-UNSUB.request {

EVSubSourceID,

EVSubDestinationID,

EVSubmodel,

EVSubValue

}

Table 3 defines the parameters of this primitive. This primitive is used by the CIP entity to unsubscribe event service from the entity of the service layer. On receipt of this primitive, the entity providing the event service cancels event subscription in the EVSubDestinationID node for the EVSubSourceID node.

6.2.6 EVENT-UNSUB.confirm

This primitive confirms an event subscription cancellation from the service layer to the CIP entity. The parameters of this primitive are:

EVENT-UNSUB.confirm {

EVSubSourceID,

EVSubDestinationID,

EVSubResultCode

}

Table 3 defines the parameters of this primitive. This primitive confirms the cancellation of event subscription. The EVSubResultCode parameter indicates a success if the event subscription cancellation is successful, otherwise an error is indicated.

6.3 Logical grouping service

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	6.3.1	6.3.2		6.3.3
LG-MEMBERIN	6.3.4			6.3.5
LG-MEMBEROUT	6.3.6			6.3.7
LG-DISMISS	6.3.8	6.3.9		6.3.10
LG-QUERY	6.3.11			6.3.12
LG-SET	6.3.13	6.3.14		6.3.15

Table 5 — LG-SAP primitive parameters

Parameter Name	Description
LGRequestorID	Requestor node ID of logical grouping.
LGCoordinatorID	Coordinator node ID of logical group.
LGMaxNum	The maximum number of logical group members.
LGMemberINID	Member ID that requests joining in a logical group.
LGMemberOUTID	Member ID that requests quitting a logical group.
LGAttributeNum	The number of logical group attributes.
LGAttribute	Data structure of name and value of specific logical group attribute. Return from the coordinator node.
LGAttributeName	The name of specific logical group attributes.
LGAttributeValue	The value of specific logical group attributes.
LGResultCode	Result code of logical grouping service operation.

6.3.1 LG-ESTABLISH.request

This primitive requests the establishment of a logical group from the application layer. The parameters of this primitive are:

LG-ESTABLISH.request {

LGRequestorID,

LGCoordinatorID,

LGMaxNum

}

Table 5 defines the parameters of this primitive. This primitive is used by the CIP entity from the application layer to request the establishment of a logical group. 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.

6.3.2 LG-ESTABLISH.indication

This primitive indicates the logical group establishment from the service layer to the local CIP entity. The syntax of this primitive is:

LG-ESTABLISH.indication {

LGRequestorID,

LGCoordinatorID,

LGMaxNum

}

Table 5 defines the parameters of this primitive. This primitive is used when the service layer indicates an establishment of a logical group to the CIP entity. 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.

6.3.3 LG-ESTABLISH.confirm

This primitive confirms a logical group establishment from the service layer to the CIP entity in the application layer. The parameters of this primitive are:

LG-ESTABLISH.confirm {

LGRequestorID,

LGCoordinatorID,

LGResultCode

}

Table 5 defines the parameters of this primitive. This primitive reports a result of the logical group establishment request. The LGResultCode parameter indicates a success if a logical group coordinated by the LGCoordinatorID node is successful. Otherwise, an error is indicated to the LGRequestorID node.

6.3.4 LG-MEMBERIN.request

This primitive requests a membership of a logical group from the application layer. The parameters of this primitive are:

LG-MEMBERIN.request {

LGCoordinatorID,

LGMemberINID

}

Table 5 defines the parameters of this primitive. This primitive is used by the 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. 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, an LGResultCode value is generated to indicate an error that the maximum member number is reached.

6.3.5 LG-MEMBERIN.confirm

This primitive confirms a result of the request of a membership of a logical group to the CIP entity in the application layer. The parameters of this primitive are:

LG-MEMBERIN.confirm {

LGCoordinatorID,

LGMemberINID,

LGResultCode

}

Table 5 defines the parameters of this primitive. This primitive reports a result of the membership request to a logical group. The LGResultCode indicates a success if the LGMemberINID node successfully joins a logical group coordinated by the LGCoordinatorID node. Otherwise, an error is indicated.

6.3.6 LG-MEMBEROUT.request

This primitive requests a membership cancellation of a logical group from the application layer. The parameters of this primitive are:

LG-MEMBEROUT.request {

LGCoordinatorID,

LGMemberOUTID

}

Table 5 defines the parameters of this primitive. This primitive is used by the 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. 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.

6.3.7 LG-MEMBEROUT.confirm

This primitive confirms result of the request of a membership cancellation of a logical group to the CIP entity in the application layer. The parameters of this primitive are:

LG-MEMBERIN.confirm {

LGCoordinatorID,

LGMemberOUTID,

LGResultCode

}

Table 5 defines the parameters of this primitive. This primitive reports a result of membership cancellation request to a logical group. The LGResultCode parameter indicates a success if the LGMemberINID node successfully quits a logical group coordinated by the LGCoordinatorID node. Otherwise, an error is indicated.

6.3.8 LG-DISMISS.request

This primitive requests the dismissal of a logical group from the application layer. The parameters of this primitive are:

LG-DISMISS.request {

LGRequestorID,

LGCoordinatorID

}

Table 5 defines the parameters of this primitive. This primitive is used by the CIP entity from the application layer to request dismissal of a logical group in the LGCoordinatorID node. 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.

6.3.9 LG-DISMISS.indication

This primitive indicates the logical group dismissal from the service layer to the local CIP entity. The parameters of this primitive are:

LG-DISMISS.indication {

LGRequestorID,

LGCoordinatorID

}

Table 5 defines the parameters of this primitive. This primitive is used when the service layer indicates dismissal of a logical group to the CIP entity in the LGCoordinatorID node. On receipt of this primitive, the CIP entity in the LGCoordinatorID node is indicated the dismissal of a logical group.

6.3.10 LG-DISMISS.confirm

This primitive confirms a logical group dismissal from the service layer to the CIP entity in the LGRequestorID node. The parameters of this primitive are:

LG-DISMISS.confirm {

LGRequestorID,

LGCoordinatorID,

LGResultCode

}

Table 5 defines the parameters of this primitive. This primitive reports a result of logical group dismissal request. The LGResultCode parameter indicates a success if a logical group coordinated by the LGCoordinatorID node has now be successfully dismissed. Otherwise, an error is indicated to the LGRequestorID node.

6.3.11 LG-QUERY.request

This primitive queries logical group attributes by CIP entities in the application layer. The parameters of this primitive are:

LG-QUERY.request {

LGRequestorID,

LGCoordinatorID,

LGAttributeNum,

LGAttribute

}

Table 5 defines the parameters of this primitive. This primitive is used by the 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. On receipt of this primitive, the LGCoordinatorID node queries LGAttributeNum attributes of current logical group. The attribute names and values are structured by LGAttribute.

6.3.12 LG-QUERY.confirm

This primitive returns the results of logical group attributes to the CIP entity in the application layer. The parameters of this primitive are:

LG-QUERY.confirm {

LGRequestorID,

LGCoordinatorID,

LGAttributeNum,

LGAttribute,

LGResultCode

}

Table 5 defines the parameters of this primitive. This primitive returns the result of querying attribute of a logical group coordinated by the LGCoordinatorID node. The LGResultCode indicates a success if the LGAttributeNum attributes are successfully returned in the LGAttribute parameter. Otherwise, an error is confirmed.

6.3.13 LG-SET.request

This primitive requests to set specific attribute values of a logical group from the application layer. The parameters of this primitive are:

LG-SET.request {

LGRequestorID,

LGCoordinatorID,

LGAttributeName,

LGAttributeValue

}

Table 5 defines the parameters of this primitive. This primitive is used by the CIP entity from the application layer to set the value of specific attribute of a logical group. On receipt of this primitive, the entity of the service layer in the LGCoordinatorID node tries to retrieve the LGAttributeName attribute. If the LGAttributeName attribute is invalid, an error is generated. Otherwise, the LGCoordinatorID node tries to set the value of the LGAttributeName attribute with LGAttributeValue. Additional procedure may be applied to check if LGAttributeValue is valid for the LGAttributeName attribute. If not successfully, an error code is generated.

6.3.14 LG-SET.indication

This primitive indicates to the CIP entity in the application layer that the request of setting specific attribute value of a logical group has been received. The parameters of this primitive are:

LG-SET.indication {

LGRequestorID,

LGCoordinatorID,

LGAttributeName

}

Table 5 defines the parameters of this primitive. This primitive is used when the service layer indicates the process of setting attribute value of a logical group to the CIP entity in the LGCoordinatorID node. On receipt of this primitive, the CIP entity is indicated that the request to set LGAttributeName attribute has been received.

6.3.15 LG-SET.confirm

This primitive confirms attribute value changes of a logical group from the service layer to the CIP entity in the application layer. The parameters of this primitive are:

LG-SET.confirm {

LGRequestorID,

LGCoordinatorID,

LGAttributeName,

LGResultCode

}

Table 5 defines the parameters of this primitive. This primitive reports a result of logical group establishment request. The LGResultCode indicates a success if the value of LGAttributeName attribute is set successfully. If the value of LGAttributeName attribute is not set successfully, and an error is indicated to the LGRequestorID node.

6.4 Data grouping service

Data grouping service is provided through DG-SAP. The DG-SAP is the logical interface between data grouping service entity in the service layer and the 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 DG-SAP. Table 7 outlines primitive parameters.

Table 6 — DG-SAP primitive summary

Name	Request	Indication	Response	Confirm
DG-DGQUERY	6.4.1			6.4.2
DG-DGEXEC	6.4.3	6.4.4		6.4.5

Table 7 — DG-SAP primitive parameters

Parameter Name	Description
DGSrcID	Source node ID of data grouping service.
DGDstID	Destination node ID of data grouping service.
DGTimeRef	Value of time reference in data grouping process. Return from destination node.
DGExecVal	Value for data grouping process in destination node.
DGResultCode	Result code of data grouping service.

6.4.1 DG-DGQUERY.request

This primitive queries values of reference time for data grouping by CIP entities in the application layer. The parameters of this primitive are:

DG-DGQUERY.request {

DGSrcID,

DGDstID,

DGTimeRef

}

Table 7 defines the parameters of this primitive. This primitive is used by the CIP entity from the application layer to query value of reference time from the DGDstID node which is required by data grouping process in the DGSrcID node. On receipt of this primitive, the DGDstID node gets current reference time for generating sensory data, which is returned with DGTimeRef.

6.4.2 DG-DGQUERY.confirm

This primitive returns the values of reference time to the CIP entity in the application layer. The parameters of this primitive are:

DG-DGQUERY.confirm {

DGSrcID,

DGDstID,

DGTimeRef,

DGResultCode

}

Table 7 defines the parameters of this primitive. This primitive returns the result of querying values of reference time from the DGDstID node. The DGResultCode indicates a success if the value of reference time for generating sensory data are successfully returned in the DGTimeRef parameter. Otherwise, an error is indicated.

6.4.3 DG-DGEXEC.request

This primitive requests the execution of data grouping by the CIP entity in the application layer. The parameters of this primitive are:

DG-DGEXEC.request {

DGSrcID,

DGDstID,

DGExecVal

}

Table 7 defines the parameters of this primitive. This primitive is used by the CIP entity from the application layer in the DGSrcID node to request the execution of data grouping in the DGDstID node. On receipt of this primitive, the DGDstID node executes data grouping process locally, in which DGExecVal is used to for data grouping in the DGSrcID node.

6.4.4 DG-DGEXEC.indication

This primitive indicates the execution of data grouping process from the service layer to the local CIP entity. The parameters of this primitive are:

DG-DGEXEC.indication {

DGSrcID,

DGDstID,

DGExecVal

}

Table 7 defines the parameters of this primitive. This primitive is used when the service layer indicates the execution of data grouping process to the CIP entity. On receipt of this primitive, the CIP entity is indicated the execution of data grouping process.

6.4.5 DG-DGEXEC.confirm

This primitive confirms the execution of data grouping from the service layer to the CIP entity. The parameters of this primitive are:

DG-DGEXEC.confirm {

DGSrcID,

DGDstID,

DGResultCode

}

Table 7 defines the parameters of this primitive. This primitive reports execution result of data grouping process to the CIP entity. The DGResultCode indicates a success if the reference time for generating sensory data in the DGDstID node is successfully synchronized with the reference time in the DGSrcID node. Otherwise, an error is indicated to the DGSrcID node.

6.5 Data registration service

Data registration service is provided through REG-SAP. The REG-SAP is the logical interface between data registration entity in the service layer and the 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 REG-SAP. Table 9 outlines primitive parameters.

Table 8 — REG-SAP primitive summary

Name	Request	Indication	Response	Confirm
REG-REGQUERY	6.5.1			6.5.2
REG-REGEXEC	6.5.3	6.5.4		6.5.5

Table 9 — REG-SAP primitive parameters

Parameter Name	Description
REGSrcID	Source node ID of data registration service.
REGDstID	Destination node ID of data registration service.
REGRef	Data structure of name and value of specific reference attribute in data registration process. Return from destination node.
REGRefDimension	The dimensions of REGRef data for data registration process.
REGExecVal	Multi-dimensional data for data registration process in destination node.
REGResultCode	Result code of data registration service.

6.5.1 REG-REGQUERY.request

This primitive queries reference attributes for data registration process by CIP entities in the application layer. The parameters of this primitive are:

REG-REGQUERY.request {

REGSrcID,

REGDstID,

REGRef

}

Table 9 defines the parameters of this primitive. This primitive is used by the CIP entity from the application layer to query value of the REGRef attribute from the REGDstID node which is required by data registration process in the REGSrcID node. On receipt of this primitive, the REGSrcID node requests the value of the REGRef attribute in the REGDstID node.

6.5.2 REG-REGQUERY.confirm

This primitive returns a result of querying value of the REGRef attribute to the CIP entity in the application layer. The parameters of this primitive are:

REG-REGQUERY.confirm {

REGSrcID,

REGDstID,

REGRef,

REGResultCode

}

Table 9 defines the parameters of this primitive. This primitive returns the result of querying REGRef attribute value in the REGDstID node. The REGResultCode parameter indicates a success if the value of the REGRef attribute is successfully returned in the REGRef parameter. Otherwise, an error is indicated.

6.5.3 REG-REGEXEC.request

This primitive requests the execution of data registration by the CIP entity in the application layer. The parameters of this primitive are:

REG-REGEXEC.request {

REGSrcID,

REGDstID,

REGRefDimension,

REGExecVal

}

Table 9 defines the parameters of this primitive. This primitive is used by the CIP entity from the application layer in the REGSrcID node to request the execution of data registration in the REGDstID node. On receipt of this primitive, the REGDstID node executes data registration process locally, in which REGExecVal is used to perform data registration. The dimension of REGExecVal is specified by REGRefDimension.

6.5.4 REG-REGEXEC.indication

This primitive indicates the execution of data registration process from the service layer to the local CIP entity. The parameters of this primitive are:

REG-REGEXEC.indication {

REGSrcID,

REGDstID

}

Table 9 defines the parameters of this primitive. This primitive is used when the service layer indicates the execution of data registration process to the CIP entity in the REGDstID node. On receipt of this primitive, the CIP entity in the REGDstID node is indicated the execution of data registration process.

6.5.5 REG-REGEXEC.confirm

This primitive confirms the execution of data registration from the service layer to the CIP entity. The parameters of this primitive are:

REG-REGEXEC.confirm {

REGSrcID,

REGDstID,

REGResultCode

}

Table 9 defines the parameters of this primitive. This primitive reports execution result of data registration process to the CIP entity in the REGSrcID node. The REGResultCode indicates a success if the data registration process is successfully executed within the REGDstID node. Otherwise, an error is indicated to the REGSrcID node.

6.6 Information description service

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 the 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 INFO-SAP. Table 11 outlines primitive parameters.

Table 10 — INFO-SAP primitive summary

Name	Request	Indication	Response	Confirm
INFO-LEVELGET	6.6.1			6.6.2
INFO-LEVELSET	6.6.3	6.6.4		6.6.5
INFO-DATA	6.6.6	6.6.7		6.6.8

Table 11 — INFO-SAP primitive parameters

Parameter Name	Description
InfoSrcID	Source node ID of information description service.
InfoDstID	Destination node ID of information description service.
LevelVal	Difference information description levels.
InfoDataDimension	The dimensions of InfoData.
InfoData	Multi-dimensional data which are transferred between source node and destination node.
InfoResultCode	Result code of information description primitives.

6.6.1 INFO-LEVELGET.request

This primitive requests information description levels by CIP entities in the application layer. The parameters of this primitive are:

INFO-LEVELGET.request {

InfoSrcID,

InfoDstID,

LevelVal

}

Table 11 defines the parameters of this primitive. This primitive is used by the CIP entity from the application layer to query information description level of the InfoDstID node which is required by the InfoSrcID node. On receipt of this primitive, the InfoDstID node gets current information description level.

6.6.2 INFO-LEVELGET.confirm

This primitive returns information description level to the CIP entity in the application layer. The parameters of this primitive are:

INFO-LEVELGET.confirm {

InfoSrcID,

InfoDstID,

LevelVal,

InfoResultCode

}

Table 11 defines the parameters of this primitive. This primitive returns the result of querying information description level to the InfoSrcID node. The InfoResultCode parameter indicates a success if the information description level of the InfoDstID node is successfully returned in the LevelVal parameter. Otherwise, an error is indicated.

6.6.3 INFO-LEVELSET.request

This primitive sets the information description level by the CIP entity in the application layer. The parameters of this primitive are:

INFO-LEVELSET.request {

InfoSrcID,

InfoDstID,

LevelVal

}

Table 11 defines the parameters of this primitive. This primitive is used by the CIP entity from the application layer in the InfoSrcID node to set the information description level of the InfoDstID node to LevelVal. 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. 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.

6.6.4 INFO-LEVELSET.indication

This primitive indicates the set operation of information description levels from the service layer to the local CIP entity. The parameters of this primitive are:

INFO-LEVELSET.indication {

InfoSrcID,

InfoDstID,

LevelVal

}

Table 11 defines the parameters of this primitive. This primitive is used when the service layer indicates the set operation of information description levels to the CIP entity in the InfoDstID node. 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.

6.6.5 INFO-LEVELSET.confirm

This primitive confirms the set of information description levels from the service layer to the CIP entity. The parameters of this primitive are:

INFO-LEVELSET.confirm {

InfoSrcID,

InfoDstID,

LevelVal,

InfoResultCode

}

Table 11 defines the parameters of this primitive. This primitive reports the result of setting information description levels to the CIP entity in the InfoSrcID node. The InfoResultCode indicates a success if the information description level in the InfoDstID node is successfully set to LevelVal. Otherwise, an error is indicated to the InfoSrcID node.

6.6.6 INFO-DATA.request

This primitive requests the transferring of information of specific information description levels by the CIP entity in the application layer. The parameters of this primitive are:

INFO-DATA.request {

InfoSrcID,

InfoDstID,

LevelVal,

InfoDataDimension,

InfoData

}

Table 11 defines the parameters of this primitive. This primitive is used by the 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. 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.

6.6.7 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. The parameters of this primitive are:

INFO-DATA.indication {

InfoSrcID,

InfoDstID,

LevelVal,

InfoDataDimension,

InfoData

}

Table 11 defines the parameters of this primitive. This primitive is used when the service layer indicates the reception of data unit of specific information description level to the CIP entity in the InfoDstID node. 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.

6.6.8 INFO-DATA.confirm

This primitive confirms the transfer of data unit of specific information description level from the service layer to the CIP entity. The parameters of this primitive are:

INFO-DATA.confirm {

InfoSrcID,

InfoDstID,

LevelVal,

InfoResultCode

}

Table 11 defines the parameters of this primitive. This primitive reports the result of transferring data unit of the LevelVal information description level to the CIP entity in the InfoSrcID node. The InfoResultCode

indicates a success if a data unit of the LevelVal information description level is successfully transferred from the InfoSrcID node to the InfoDstID node. Otherwise, an error is indicated to the InfoSrcID node.

6.7 Node-to-node inter-activation service

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 the 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 N2NACT-SAP. Table 13 outlines primitive parameters.

Table 12 — N2NACT-SAP primitive summary

Name	Request	Indication	Response	Confirm
N2NACT	6.7.1			6.7.2

Table 13 — N2NACT-SAP primitive parameters

Parameter Name	Description
N2NSrcID	Source node ID of node-to-node inter-activation service.
N2NDstID	Destination node ID of node-to-node inter-activation service.
N2NACTMode	Difference node-to-node inter-activation modes.
N2NACTDataAttributeNum	The number of attributes used in node-to-node inter-activation process.
N2NACTDataAttribute	Data structure of attribute values for node-to-node inter-activation process.
N2NResultCode	Result code of node-to-node inter-activation process.

6.7.1 N2NACT.request

This primitive implements the inter-activation process between two nodes which can be requested by the CIP entity in the application layer. The parameters of this primitive are:

N2NACT.request {

N2NSrcID,

N2NDstID,

N2NACTMode,

N2NACTDataAttributeNum,

N2NACTDataAttribute

}

Table 13 defines the parameters of this primitive. This primitive is used by the CIP entity from the application layer to request activation in the N2NDstID node from the N2NSrcID node. 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. Different modes could be supported in

this service. The context data for activation process is given by N2NACTDataAttribute in the primitive. N2NACTDataAttributeNum specifies the number of attribute in N2NACTDataAttribute.

6.7.2 N2NACT.confirm

This primitive confirms the result of node-to-node inter-activation process from the service layer to the CIP entity. The parameters of this primitive are:

N2NACT.confirm {

N2NSrcID,

N2NDstID,

N2NACTMode,

N2NResultCode

}

Table 13 defines the parameters of this primitive. This primitive reports the result of node-to-node inter-activation process to the CIP entity in the N2NSrcID node. The N2NResultCode indicates a success if the N2NDstID node is successfully activated by the N2NSrcID node with the N2NACTMode mode. Otherwise, an error is indicated.

6.8 Parameter adaptation service

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 the 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 14 lists the primitives supported by the PAR-SAP. Table 15 outlines primitive parameters.

Table 14 — PAR-SAP primitive summary

Name	Request	Indication	Response	Confirm
PAR	6.8.1	6.8.2		6.8.3

Table 15 — PAR-SAP primitive parameters

Parameter Name	Description
PARSrcID	Source node ID of parameter adaptation service.
PARDstID	Destination node ID of parameter adaptation service.
PARParameterName	Parameter name in adaptation or query process.
PARParameterLength	Length of PARParameter in parameter adaptation process.
PARParameter	Value for PARParameterName in parameter adaptation process.
PARResultCode	Result code of parameter adaptation service.

6.8.1 PAR.request

This primitive requests the parameter adaptation process by the CIP entity in the application layer. The parameters of this primitive are:

PAR.request {

PARSrcID,

PARDstID,

PARParameterName,

PARParameterLength,

PARParameter

}

Table 15 defines the parameters of this primitive. This primitive is used by the CIP entity from the application layer in the PARSrcID node to request the parameter adaptation in the PARDstID node. 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.

6.8.2 PAR.indication

This primitive indicates to the CIP entity in the application layer that the request of parameter adaptation has been received. The parameters of this primitive are:

PAR.indication {

PARSrcID,

PARDstID,

PARParameterName,

}

Table 15 defines the parameters of this primitive. This primitive is used when the service layer indicates the parameter adaptation process to the CIP entity in the PARDstID node. On receipt of this primitive, the CIP entity is indicated that the request of PARParameterName parameter adaptation has been received.

6.8.3 PAR.confirm

This primitive confirms the result of parameter adaptation from the service layer to the CIP entity. The parameters of this primitive are:

PAR.confirm {

PARSrcID,

PARDstID,

PARParameterName,

PARResultCode

}

Table 15 defines the parameters of this primitive. This primitive reports the result of parameter adaptation process to the CIP entity in the PARSrcID node. The PARResultCode indicates a success if the PARParameterName parameter in the PARDstID node is successfully updated. Otherwise, an error is indicated to the InfoSrcID node.

7 Enhanced services and interfaces specifications

7.1 Overview

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 16, the names of service access points (SAPs) through which specific service is provided are listed.

Table 16 — 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

7.2 QoS management service

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.

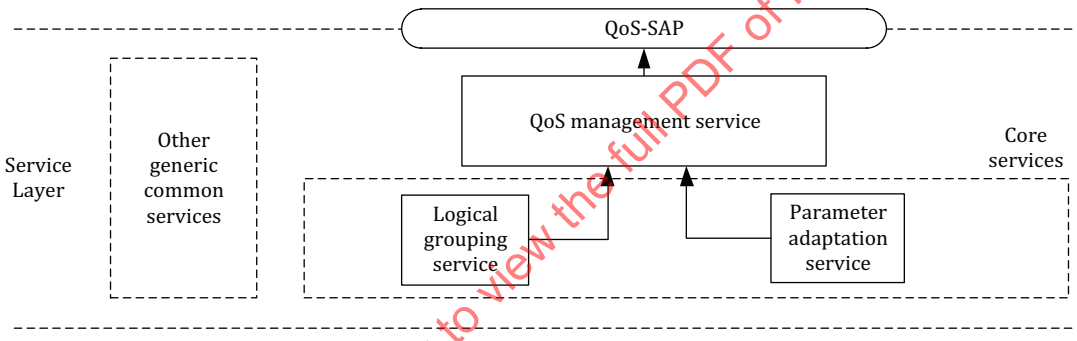

Figure 5 — — Relationship among QoS management service and other services

QoS management service is provided through QoS-SAP. The QoS-SAP is the logical interface between QoS management entity in the service layer and the 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 QoS-SAP. Table 18 outlines primitive parameters.

Table 17 — QoS-SAP primitive summary

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

Table 18 — QoS-SAP primitive parameters

Parameter Name	Description
QoSRequestorID	Node ID of QoS management service requestor.
QoSProfileManagerID	Node ID of QoS profile manager.
QoSProfileID	Profile ID for QoS management service.
QoSProfileUpdateMode	The profile updating mode for QoS management service.
QoSProfileLGlist	Node ID list of coordinator for Logical groups in a QoS profile.
QoSProfilePARlist	Parameter name and value list for logical groups in a QoS profile.
QoSResultCode	Result code of QoS management service.

7.2.1 QoS-PROFILE-ESTABLISH.request

This primitive requests the establishment of a QoS Profile for QoS management from the application layer. The parameters of this primitive are:

QoS-PROFILE-ESTABLISH.request {

QoSRequestorID,

QoSProfileManagerID,

QoSProfileID

}

Table 18 defines the parameters of this primitive. This primitive is used by the CIP entity from the application layer in the QoSRequestorID node to request the establishment of a QoS profile in the QoSProfileManagerID node. 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 the establishment of a new QoS profile. The new QoS profile will be identified by QoSProfileID.

7.2.2 QoS-PROFILE-ESTABLISH.indication

This primitive indicates the establishment of a QoS profile from the service layer to the application layer. The parameters of this primitive are:

QoS-PROFILE-ESTABLISH.indication {

QoSRequestorID,

QoSProfileManagerID,

QoSProfileID

}

Table 18 defines the parameters of this primitive. This primitive is used by the QoS management service entity to indicate an establishment of a QoS profile to the CIP entity in the QoSProfileManagerID node. On receipt of this primitive, the CIP entity in QoSProfileManagerID node is indicated the establishment of a new QoS profile identified by QoSProfileID.

7.2.3 QoS-PROFILE-ESTABLISH.confirm

This primitive confirms a new QoS profile establishment from the service layer to the CIP entity in the application layer. The parameters of this primitive are:

QoS-PROFILE-ESTABLISH.confirm {

QoSRequestorID,

QoSProfileManagerID,

QoSProfileID,

QoSResultCode

}

Table 18 defines the parameters of this primitive. This primitive reports a result of a QoS profile establishment request. The QoSResultCode indicates a success if a new QoS profile identified by QoSProfileID in the QoSProfileManagerID node is successfully established. Otherwise, an error is indicated to the QoSRequestorID node.

7.2.4 QoS-PROFILE-UPDATE.request

This primitive requests to update a QoS Profile for QoS management from the application layer. The parameters of this primitive are:

QoS-PROFILE-UPDATE.request {

QoSRequestorID,

QoSProfileManagerID,

QoSProfileID,

QoSProfileUpdateMode,

QoSProfileLGlist,

QoSProfilePARlist

}

Table 18 defines the parameters of this primitive. This primitive is used by the CIP entity from the application layer in the QoSRequestorID node to request to update a QoS profile identified by QoSProfileID in the QoSProfileManagerID node. 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.

7.2.5 QoS-PROFILE-UPDATE.confirm

This primitive confirms the result of updating a QoS Profile to the application layer. The parameters of this primitive are:

QoS-PROFILE-UPDATE.confirm {

QoSRequestorID,

QoSProfileManagerID,

QoSProfileID,

QoSProfileUpdateMode,

QoSResultCode

}

Table 18 defines the parameters of this primitive. This primitive is used 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. On receipt of this primitive, the QoSRequestorID node is confirmed the result of updating QoSProfileID profile in the QoSProfileManagerID node. The QoSResultCode indicates if the profile updating is successful. Otherwise, an error is indicated to the QoSRequestorID node.

7.2.6 QoS-PROFILE-APPLY.request

This primitive requests to apply a QoS Profile for QoS management from the application layer. The parameters of this primitive are:

QoS-PROFILE-APPLY.request {

QoSRequestorID,

QoSProfileManagerID,

QoSProfileID

}

Table 18 defines the parameters of this primitive. This primitive is used by the CIP entity from the application layer in the QoSRequestorID node to request to apply a QoS profile identified by QoSProfileID. 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 triggers a series of processes using logical grouping service and parameter adaptation service. All logical groups defined by QoSProfileLGlist updates defined parameters with defined values specified by QoSProfilePARlist.

7.2.7 QoS-PROFILE-APPLY.confirm

This primitive confirms the result of applying a QoS Profile to the application layer. The parameters of this primitive are:

QoS-PROFILE-APPLY.confirm {

QoSRequestorID,

QoSProfileManagerID,

QoSProfileID,

QoSResultCode

}

Table 18 defines the parameters of this primitive. This primitive is used 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. On receipt of this primitive, the QoSRequestorID node is confirmed the result of applying QoSProfileID profile. The QoSResultCode parameter indicates a success if the profile is applied successfully. Otherwise, an error is indicated by QoS management service entities in the service layer.

7.2.8 QoS-PROFILE-DELETE.request

This primitive requests to delete a QoS Profile for QoS management from the application layer. The parameters of this primitive are:

QoS-PROFILE-DELETE.request {

QoSRequestorID,

QoSProfileManagerID,

QoSProfileID

}

Table 18 defines the parameters of this primitive. This primitive is used by the CIP entity from the application layer in the QoSRequestorID node to request deletion of a QoS profile in the QoSProfileManagerID node. 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.

7.2.9 QoS-PROFILE-DELETE.indication

This primitive indicates the deletion of a QoS profile from the service layer to the application layer. The parameters of this primitive are:

QoS-PROFILE-DELETE.indication {

QoSRequestorID,

QoSProfileManagerID,

QoSProfileID

}

Table 18 defines the parameters of this primitive. This primitive is used by the QoS management service entity to indicate a deletion of a QoS profile to the CIP entity in the QoSProfileManagerID node. On receipt of this primitive, the CIP entity in QoSProfileManagerID node is indicated the deletion of a QoS profile identified by QoSProfileID.

7.2.10 QoS-PROFILE-DELETE.confirm

This primitive confirms a QoS profile deletion from the service layer to the CIP entity in the application layer. The parameters of this primitive are:

QoS-PROFILE-DELETE.confirm {

QoSRequestorID,

QoSProfileManagerID,

QoSProfileID,

QoSResultCode

}

Table 18 defines the parameters of this primitive. This primitive reports a result of a QoS profile deletion request. The QoSResultCode parameter indicates a success if the QoS profile identified by QoSProfileID in the QoSProfileManagerID node is successfully deleted. Otherwise, an error is indicated to the QoSRequestorID node.

7.3 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. 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.

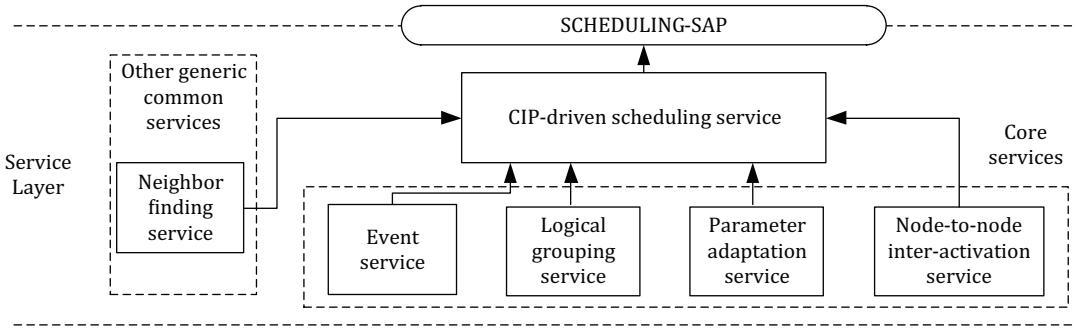

Figure 6 — Relationship among CIP-driven scheduling service and other services

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 the 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 SCHEDULING-SAP. Table 20 outlines primitive parameters.

Table 19 — SCHEDULING-SAP primitive summary

Name	Request	Indication	Response	Confirm
SCHEDULING-SCHEME-ESTABLISH	7.3.1	7.3.2		7.3.3
SCHEDULING-SCHEME-UPDATE	7.3.4			7.3.5
SCHEDULING-SCHEME-APPLY	7.3.6			7.3.7
SCHEDULING-SCHEME-DELETE	7.3.8	7.3.9		7.3.10

Table 20 — SCHEDULING-SAP primitive parameters

Parameter Name	Description
SCHEDULINGRequestorID	Node ID of CIP-driven scheduling service requestor.
SCHEDULINGSchemeManagerID	Node ID of scheme manager.
SCHEDULINGSchemeID	Scheme ID for CIP-driven scheduling service.
SCHEDULINGMode	The scheduling mode for CIP-driven scheduling service.
SchemeEVSubNodelist	Event subscription node ID list for CIP-driven scheduling service.
SchemeEVSubTypelist	Event type list for each event subscription nodes.
SchemePARlist	Parameter name and value list for CIP-driven scheduling service.
SCHEDULINGResultCode	Result code of CIP-driven scheduling service.

7.3.1 SCHEDULING-SCHEME-ESTABLISH.request

This primitive requests the establishment of a scheme for CIP-driven scheduling service from the application layer. The parameters of this primitive are:

SCHEDULING-SCHEME-ESTABLISH.request {

SCHEDULINGRequestorID,

SCHEDULINGSchemeManagerID,

SCHEDULINGSchemeID

}

Table 20 defines the parameters of this primitive. This primitive is used by the CIP entity from the application layer in the SCHEDULINGRequestorID node to request an establishment of a scheme in the SCHEDULINGSchemeManagerID node. 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 the establishment of a new scheduling scheme. The new scheme will be identified by SCHEDULINGSchemeID.

7.3.2 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. The parameters of this primitive are:

SCHEDULING-SCHEME-ESTABLISH.indication {

SCHEDULINGRequestorID,

SCHEDULINGSchemeManagerID,

SCHEDULINGSchemeID

}

Table 20 defines the parameters of this primitive. This primitive is used by the CIP-driven scheduling service entity to indicate an establishment of a scheme to the CIP entity in the SCHEDULINGSchemeManagerID node. On receipt of this primitive, the CIP entity in the SCHEDULINGSchemeManagerID node is indicated the establishment of a new scheme identified by SCHEDULINGSchemeID.

7.3.3 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. The parameters of this primitive are:

SCHEDULING_SCHEME-ESTABLISH.confirm {

SCHEDULINGRequestorID,

SCHEDULINGSchemeManagerID,

SCHEDULINGSchemeID,

SCHEDULINGResultCode

}

Table 20 defines the parameters of this primitive. This primitive reports a result of a scheme establishment request. The SCHEDULINGResultCode parameter indicates a success if a new scheme identified by SCHEDULINGSchemeID in the SCHEDULINGSchemeManagerID node is successfully established. Otherwise, an error is indicated to the SCHEDULINGRequestorID node.

7.3.4 SCHEDULING-SCHEME-UPDATE.request

This primitive requests to update a scheme for CIP-driven scheduling service from the application layer. The parameters of this primitive are:

SCHEDULING-SCHEME-UPDATE.request {

SCHEDULINGRequestorID,

SCHEDULINGSchemeManagerID,

SCHEDULINGSchemeID,

SCHEDULINGMode,

SchemeEVSubNodelist,

SchemeEVSubTypelist,

SchemePARlist

}

Table 20 defines the parameters of this primitive. This primitive is used by the CIP entity from the application layer in the SCHEDULINGRequestorID node to request to update a scheme identified by SCHEDULINGSchemeID in the SCHEDULINGSchemeManagerID node. 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. SchemeEVSubNodelist, SchemeEVSubTypelist and SchemePARlist define event subscription node list, event type list and parameter list in the scheme.

7.3.5 SCHEDULING-SCHEME-UPDATE.confirm

This primitive confirms the result of updating a scheme to the application layer. The parameters of this primitive are:

SCHEDULING-SCHEME-UPDATE.confirm {

SCHEDULINGRequestorID,

SCHEDULINGSchemeManagerID,

SCHEDULINGSchemeID,

SCHEDULINGResultCode

}

Table 20 defines the parameters of this primitive. This primitive is used 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. On receipt of this primitive, the SCHEDULINGRequestorID node is confirmed the result of updating SCHEDULINGSchemeID scheme in the SCHEDULINGSchemeManagerID node. The SCHEDULINGResultCode parameter indicates a success if the scheme is updated successfully. Otherwise, an error is indicated to the SCHEDULINGRequestorID node.

7.3.6 SCHEDULING-SCHEME-APPLY.request

This primitive requests to apply a scheme for CIP-driven scheduling service from the application layer. The parameters of this primitive are:

SCHEDULING-SCHEME-APPLY.request {

SCHEDULINGRequestorID,

SCHEDULINGSchemeManagerID,

SCHEDULINGSchemeID

}