DRAFT FOR DEVELOPMENT

DD IEC/PAS
62573:2008

Real-time Ethernet — Real-time Automation Protocol for Industrial Ethernet (RAPIEnet)

ICS 25.040.40; 33.040.40

National foreword

This Draft for Development is the UK implementation of IEC/PAS 62573:2008.

This publication is not to be regarded as a British Standard.

It is being issued in the Draft for Development series of publications and is of a provisional nature. It should be applied on this provisional basis, so that information and experience of its practical application can be obtained.

A PAS is a Technical Specification not fulfilling the requirements for a standard, but made available to the public and established in an organization operating under a given procedure.

A review of this Draft for Development will be carried out not later than three years after its publication.

Notification of the start of the review period, with a request for the submission of comments from users of this Draft for Development, will be made in an announcement in the appropriate issue of *Update Standards*. According to the replies received, the responsible BSI Committee will judge whether the validity of the PAS should be extended for a further three years or what other action should be taken and pass their comments on to the relevant international committee.

Observations which it is felt should receive attention before the official call for comments will be welcomed. These should be sent to the Secretary of the responsible BSI Technical Committee at British Standards House, 389 Chiswick High Road, London W4 4AL.

The UK participation in its preparation was entrusted to Technical Committee AMT/7, Industrial communications: process measurement and control, including fieldbus.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

This Draft for Development was published under the authority of the Standards Policy and Strategy Committee on 31 July 2008

ISBN 978 0 580 62623 4

Amendments/corrigenda issued since publication

Date	Comments

DD IEC/PAS 62573:2008

IEC/PAS 62573

Edition 1.0 2008-03

PUBLICLY AVAILABLE SPECIFICATION

PRE-STANDARD

Real-time Ethernet – Real-time Automation Protocol for Industrial Ethernet (RAPIEnet)

CONTENTS

INTRODUCTION

This Publicly Available Specification (PAS) describes a set of specifications essential for the ISO/IEC 8802-3-based “Real-time Automation Protocol for Industrial Ethernet (RAPIEnet),” Each specification in this PAS is to be classified into a separate part of the IEC 61158 series.

This specification meets the industrial automation market objective of identifying the RTE communication networks co-existing with the ISO/IEC 8802 series, providing more predictable, time-deterministic, and reliable real-time data transfer.

More specifically, these profiles help correctly state the compliance with the ISO/IEC 8802 series, and avoid the spread of divergent implementations that would limit its use, clarity, and understanding.

Additional profiles to address specific market concerns, such as functional safety or information security, may be addressed by future parts of the IEC 61784 series. This is not within the scope of this document. This PAS specifies the RAPIEnet communication profile portion of the protocol set.

This PAS specifies the essential part of the RAPIEnet profile, which is an extension of the ISO/IEC 8802-3-based data link layer, and the application layer exploiting the services of the data link layer immediately below. This document describes the specifications essential for the RAPIEnet profile, specifically for the data link layer and the application layer, in terms of the three-layer fieldbus Reference Model, which is based in part on the OSI Basic Reference Model. Other parts of RAPIEnet are outside the scope of this document.

Real-time Ethernet – Real-time Automation Protocol for Industrial Ethernet (RAPIEnet)

1 Scope

In industrial control systems, several types of field devices may be connected to control networks. These include drives, sensors and actuators, programmable controllers, distributed control systems, and human-machine interface devices. The process control data and the state data are transferred among these field devices in the system, and the communications between these field devices requires simplicity in application programming to be executed with adequate response time. In most industrial automation systems, the control network is required to provide time-deterministic and predictable response capability for applications. Plant production may be compromised due to errors that could be introduced into the control system if the network does not provide a time-deterministic response. Therefore, the following characteristics are required for a real-time Ethernet-based control network:

a) a time-deterministic response time between the control devices;

b) the ability to share process data seamlessly across the control system.

RAPIEnet is applicable to such an industrial automation environment in which time-deterministic real-time communications are a fundamental requirement.

This PAS specifies the protocol set necessary for RAPIEnet, specifically for the data link layer and the application layer, which is mapped on top of the data link layer, to exploit the services in accordance with the three-layer fieldbus reference model, which is based in part on the OSI Basic Reference Model. Both reference models subdivide the area of standardization for interconnection into a series of layers of manageable size. Throughout this PAS, the term “service” refers to the abstract capability provided by one layer of the OSI Basic Reference model to the layer immediately above.

This PAS consists of

a) physical layer specification;

b) data link layer service definitions;

c) data link layer protocol specification;

d) application layer service definitions;

e) application layer protocol specification.

The service of both the data link and the application layer in this PAS is a conceptual architectural service, independent of administrative and implementation details.

The data link layer describes the extension of the ISO/IEC 8802-3 data link layer for RAPIEnet, and the application layer describes the utilization of the upper layer functions over the RAPIEnet data link layer protocol.

2 Normative references

The following 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 document (including any amendments) applies.

IEC 61158-3 (all parts), *Industrial communication networks – Fieldbus specifications – Part 3: Data-link layer service definition*

IEC 61158-4 (all parts), *Industrial communication networks – Fieldbus specifications – Part 4: Dat- link layer protocol specification*

IEC 61158-5 (all parts), *Industrial communication networks – Fieldbus specifications – Part 5: Application layer service definition*

IEC 61158-6 (all parts), *Industrial communication networks – Fieldbus specifications – Part 6: Application layer protocol specification*

ISO/IEC 7498-1, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*

ISO/IEC 7498-3, *Information technology – Open Systems Interconnection – Basic Reference Model: Naming and addressing*

ISO/IEC 8802-3:2000, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specificationss*

ISO/IEC 8822:1994, *Information technology – Open Systems Interconnection – Presentation service definition*

ISO/IEC 8824-1:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*

ISO/IEC 8825-1:2002, *Information technology – ASN.1 encoding rules: Specifications of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*

ISO/IEC 8886:1996, *Information technology – Open Systems Interconnection – Data link service definition*

ISO/IEC 9545:1994, *Information technology – Open Systems Interconnection – Application Layer structure*

ISO/IEC 10731:1994, *Information technology – Open Systems Interconnection – Basic Reference Model – Conventions for the definition of OSI services*

3 Terms, definitions, and abbreviations

For the purposes of this document, some of the following terms and definitions have been compiled from the referenced documents. The terms and definitions of the ISO/IEC 7498-1, ISO/IEC 8802-3, and IEC 61588 series shall be fully valid, unless otherwise stated.

3.1 IEC 61158 definitions

For the purposes of this document, the following definitions of IEC 61158 apply.

3.1.1
application
function or data structure for which data are consumed or produced

[IEC 61158-5:2007]

3.1.2
application objects
multiple object classes that manage and provide a run-time exchange of messages across the network and within the network device

[IEC 61158-5:2007]

3.1.3
bit
unit of information consisting of a 1 or a 0. This is the smallest data unit that can be transmitted

[IEC 61158-4:2007]

3.1.4
client
1) object that uses the services of another (server) object to perform a task.

2) initiator of a message to which a server reacts

[IEC 61158-4:2007]

3.1.5
communication objects
components that manage and provide a run-time exchange of messages across the network [IEC 61158-5:2007]

3.1.6
connection
logical binding between two application objects within the same or different devices

[IEC 61158-4:2007]

3.1.7
connector
coupling device employed to connect the medium of one circuit or communication element with that of another circuit or communication element

[IEC 61158-2:2007]

3.1.8
cyclic
term used to describe events that repeat in a regular and repetitive manner

[IEC 61158-3:2007]

3.1.9
Cyclic Redundancy Check (CRC)
residual value computed from an array of data and used as a representative signature for the array

[IEC 61158-4:2007]

3.1.10
data
generic term used to refer to any information carried over a fieldbus

[IEC 61158-3:2007]

3.1.11
data consistency
means for coherent transmission and access of the input or output data object between and within client and server

[IEC 61158-5:2007]

3.1.12
device
physical entity connected to the fieldbus composed of at least one communication element (the network element) and which may have a control element and/or a final element (transducer, actuator, etc.)

[IEC 61158-2:2007]

3.1.13
device profile
collection of device-dependent information and functionality providing consistency between similar devices of the same device type

[IEC 61158-5:2007]

3.1.14
diagnosis information
all data available at the server for maintenance purposes

[IEC 61158-5:2007]

3.1.15
error
discrepancy between a computed, observed, or measured value or condition and the specified or theoretically correct value or condition

[IEC 61158-4:2007]

3.1.16
error class
general grouping for related error definitions and corresponding error codes

[IEC 61158-5:2007]

3.1.17
error code
identification of a specific type of error within an error class

[IEC 61158-5:2007]

3.1.18
event
an instance of a change of conditions

[IEC 61158-5:2007]

3.1.19
frame
denigrated synonym for data link protocol data unit

[IEC 61158-3:2007]

3.1.20
index
address of an object within an application process

[IEC 61158-5:2007]

3.1.21
interface
shared boundary between two functional units, defined by functional characteristics, signal characteristics, or other characteristics as appropriate

[IEC 61158-5:2007]

3.1.22
master
device that controls the data transfer on the network and initiates the media access of the slaves by sending messages, and that constitutes the interface to the control system

[IEC 61158-2:2007]

3.1.23
medium
cable, optical fibre, or other means by which communication signals are transmitted between two or more points

[IEC 61158-2:2007]

3.1.24
network
set of nodes connected by some type of communication medium, including any intervening repeaters, bridges, routers, and lower-layer gateways

[IEC 61158-6:2007]

3.1.25
node
endpoint of a link in a network, or a point at which two or more links meet

[IEC 61158-2, modified]

3.1.26
object
abstract representation of a particular component within a device

[IEC 61158-4:2007]

NOTE An object can be

a) an abstract representation of the capabilities of a device. Objects can be composed of any or all of the following components:

   1) data (information which changes with time);
   2) configuration (parameters for behaviour);
   3) methods (things that can be done using data and configuration);

b) a collection of related data (in the form of variables) and methods (procedures) for operating on that data that have clearly defined interface and behaviour.

3.1.27
server
object that provides services to another (client) object

[IEC 61158-4:2007]

3.1.28
service
operation or function that an object and/or object class performs upon request from another object and/or object class

[IEC 61158-5:2007]

3.1.29
slave
data link entity accessing the medium only after being initiated by the preceding slave or master

[IEC 61158-3:2007]

3.2 Definitions from other standards

The following definitions apply.

3.2.1
frame
unit of data transmission for ISO/IEC 8802-3 media access control that conveys a protocol data unit between media access control service users

[IEEE Std. 802.1Q:1998]

3.2.2
message
ordered series of octets intended to convey information

[derived from ISO 2382:16.02.01]

NOTE A message is normally used to convey information between peers at the application layer.

3.2.3
switch
media access control bridge as defined in IEEE 802.1D:1998

3.3 RAPIEnet terms and definitions

Terms and definitions of this PAS are described separately in 7.4 for the data link layer service definitions and protocol specifications, in 9.4 for the application layer service definition, and in 10.4 for the application layer protocol specification.

3.4 Symbols and abbreviations

Symbols and abbreviations of this PAS are described separately in 7.4 for the data link layer service definitions and protocol specifications, in 9.4 for the application layer service definition, and in 10.4 for the application layer protocol specification.

4 Technology overview

4.1 General Information

This section describes the basic operating principles and technical features of RAPIEnet. The aim of RAPIEnet is to maximize the use of full-duplex Ethernet bandwidth through the use of an internal hardware Ethernet switch. Therefore, RAPIEnet provides a collision-free transmission mechanism between two nodes. Every RAPIEnet device detects link failure and link establishment using media sensing technologies and shares the link information with each other so that fast connectivity recovery time is also guaranteed in a line or ring network.

4.2 Operating principles

From an Ethernet point of view, a RAPIEnet device is a dual-port switching device that receives and transmits standard ISO/IEC 8802-3 Ethernet frames. It is intelligent and can control directional frame forwarding between its dual ports according to the network status and device status.

4.2.1 Frame forwarding and receiving control

By using the internal full-duplex hardware switch, RAPIEnet provides a very efficient transmission mechanism that allows each RAPIEnet device on a network to transmit frames at any time without collision. Therefore, a RAPIEnet device transmits frames without restriction of medium access as soon as a frame appears in the transmit queue. Figure 1 shows the forwarding and receiving control of the RAPIEnet device

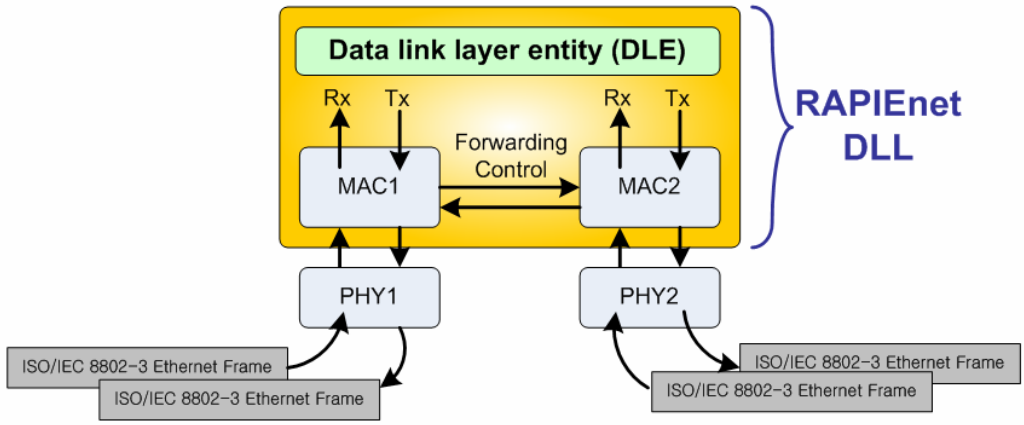


Figure 1 – Forwarding and receiving Ethernet frames

When an Ethernet frame is received at the media access control (MAC) layer through the physical interface transceiver (PHY), a RAPIEnet general device (GD, see 8.6.3.3) other than the ring network manager (RNM, see 8.6.3.3) or the line network manager (LNM, see 8.6.3.3), handles the received frame by taking one of the following actions depending on the destination MAC address in the received frame.

a) For a broadcast frame, accept and deliver the received frame to the DLE, and forward the received frame to the other port.

b) For a frame designated to the device itself, accept and deliver the received frame to the DLE without forwarding.

c) For a frame designated to the other device, do not accept the received frame, but forward the received frame to the other port.

This frame forwarding procedure is processed by the internal hardware switch so that it has little impact on the performance of the RAPIEnet protocol. In Figure 1, when the DLE has two concurrent frames to be transmitted through a common MAC port, such as a frame from the upper layer and a frame to be forwarded from the other port, the “round-robin” method is used to determine their transmission order.

As shown in Figure 2, the LNM disables both directional frame forward functions so that frames are not forwarded to the other port.

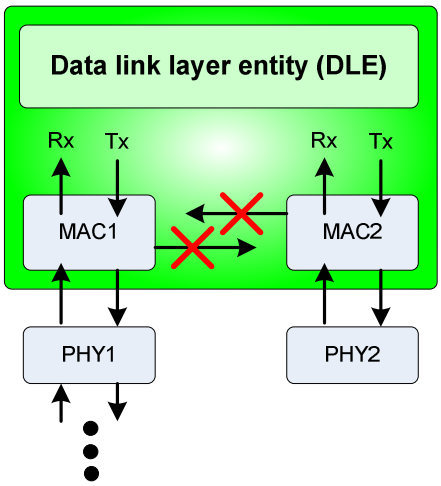


Figure 2 – Forward control of LNM

In a ring network, a frame can be continuously circulated when the designated device is not found or when the frame is broadcast on the network. In a RAPIEnet ring network, two RNMs are automatically selected, and, as shown in Figure 3, each RNM disables only one directional frame forward function to prevent infinite frame circulation. A primary RNM (RNMP) is selected first and then one of its neighbouring nodes is selected as a secondary RNM (RNMS).

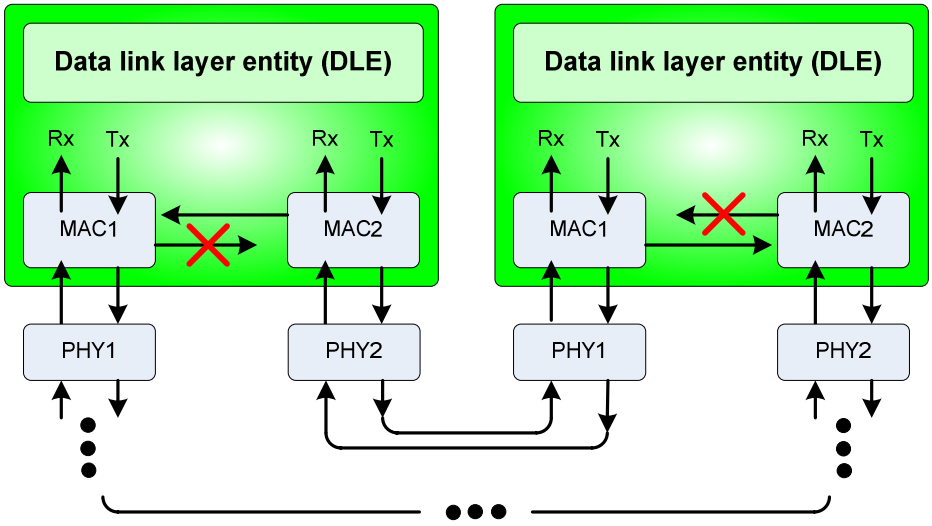


Figure 3 – Forward control of RNM

4.2.2 Link status monitoring

RAPIEnet provides an efficient mechanism for dynamic network topology management. When a link between two devices is established or released, it is automatically detected by both devices. This link status information is distributed and shared with every device on the network so that the network topology can be dynamically managed. The link status information is either “link active” or “link inactive” and the link status detection process is initiated by the hardware-triggered signal event (see 7.8.4.3). A status of “link active” means a RAPIEnet communication link is established between two devices and it is possible to send frames through the link. A status of “link inactive” means a RAPIEnet communication link is not established through an Ethernet MAC port and it is not possible to send frames through the port. By sharing all the link information on the network, every RAPIEnet device on the network knows the online network connectivity status (see 8.5.5). Figure 4 shows the intrinsic link status monitoring procedure of the RAPIEnet data link layer (DLL).

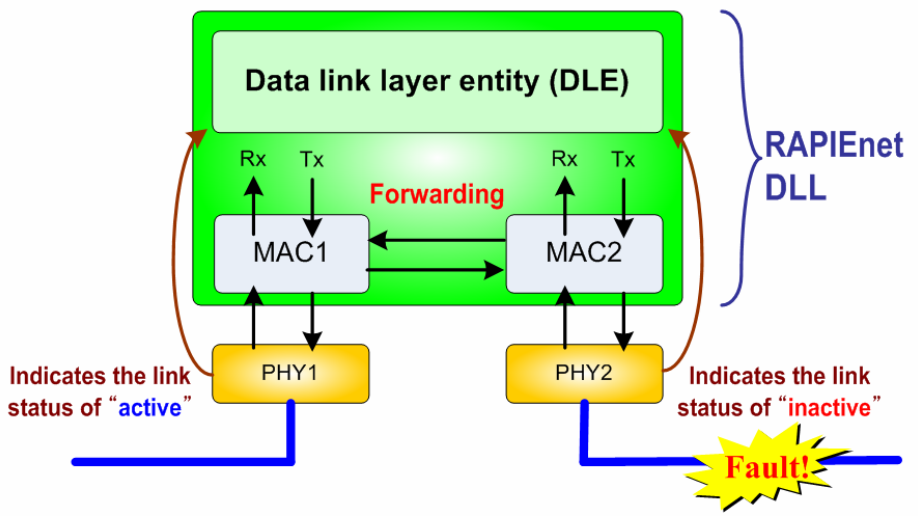


Figure 4 – Link status information

4.2.3 Error detection

A RAPIEnet device examines the frame check sequence (FCS) to determine whether the received frame is valid. When an invalid frame is received or an error is detected during the FCS, the RAPIEnet device discards the received frame, and the frame is not forwarded to the next device.

4.3 Topology

Network topology is one of the key features of automated control systems.

A line topology is more flexible, easier to configure, and more cost-effective than a star topology using Ethernet switches. In a RAPIEnet line topology, every device controls its dual Ethernet MAC port to form a daisy-chain line network. The link information between two nodes is shared with every device on the network, and thus every device updates its own path table with the link information it receives.

In a ring network, two paths are possible between two nodes: the clockwise path and the counter-clockwise path. Therefore, a RAPIEnet ring topology can maintain the network connectivity even in the case of cable break or link failure. To provide reliable and redundant network connectivity, RAPIEnet supports an online auto-configuration mechanism to form a network topology. When a link failure is detected in a ring network, the network is automatically reconfigured as a line network. RAPIEnet supports both line and ring network topologies, and also provides extremely fast recovery time from line-to-ring or from ring-to-line networks.

Figure 5 shows a basic example of RAPIEnet line network. A basic RAPIEnet line topology can be configured between two nodes with a single connection. A large-scale RAPIEnet line network can be expanded by extending the daisy-chained connection. Conversely, a large-scale RAPIEnet line network can be divided into two or more smaller scale line networks.

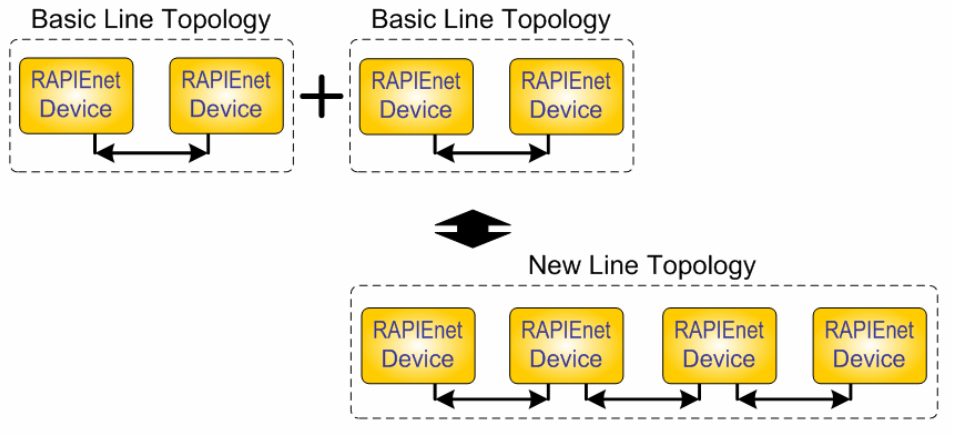


Figure 5 – Line topology

Figure 6 shows a basic example of a RAPIEnet ring network. When a new link is connected between two end nodes in a line network, the network is automatically reconfigured as a ring network. On the other hand, the network is automatically reconfigured as a line network when any link failure is detected in a ring network. Two RNMs are automatically selected to block the infinite circulation of any frame in a RAPIEnet ring network.

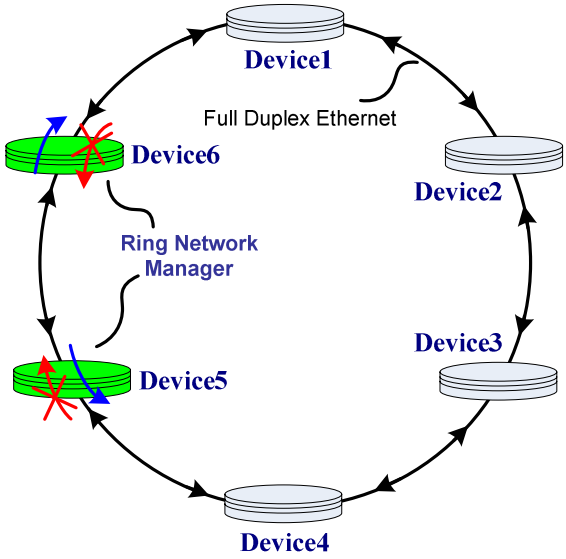


Figure 6 – Ring topology

4.4 Device reference model

This section describes the RAPIEnet device reference model using the principles, methodology, and model of ISO/IEC 7498-1. The OSI model provides a layered approach to communications standards, in which the layers can be developed and modified independently. This PAS defines functionality from the top to bottom of a full OSI stack and, potentially, some functions for the users of the stack. Functions of intermediate OSI layers 3 through 6 are consolidated into either the RAPIEnet data link layer or the RAPIEnet application layer. Likewise, features common to users of the fieldbus application layer may be provided by the RAPIEnet application layer to simplify user operation as shown in Figure 7.

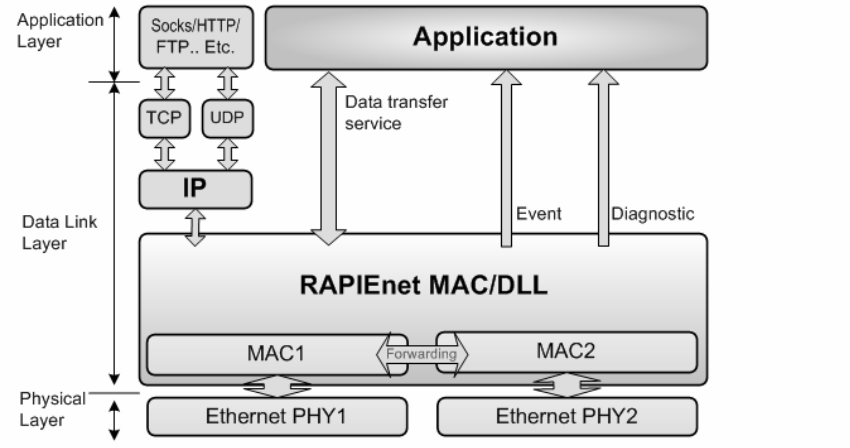


Figure 7 – OSI basic reference model

4.4.1 Physical layer

The RAPIEnet physical layer corresponds to the ISO/IEC 8802-3:2002 access method and physical layer specifications. The physical layer receives data from the upper data link layer, encodes the bits into signals, and transmits the resulting physical signals to the connected transmission medium. Signals are then received and decoded by the next device, and the data units are passed to the data link layer of the recipient device.

4.4.2 Data link layer (DLL)

The RAPIEnet DLL provides basic time-deterministic support for data communications among communication devices connected via RAPIEnet. The data link layer supports the timing demands typically required for high-performance automation applications. These do not change the basic principles of ISO/IEC 8802-3 but extend it towards RTE. Thus, it is possible to continue to use standard Ethernet hardware, infrastructure components, or test and measurement equipment, such as network analysers.

4.4.3 Application layer

The RAPIEnet application layer is designed to support the transfer of time-deterministic application requests and responses among communication devices in an automation environment. RAPIEnet allows for several optional services and protocol families to co-exist within the same communication device. In this way, generic Ethernet communication, such as TCP/UDP/IP-based protocols, may be implemented alongside real-time communications, file access protocols, and other generic protocols and services.

4.5 Data link layer overview

The RAPIEnet DLL has some unique technical features, such as extremely fast network recovery, plug and play, network management information base management, and automatic network configuration.

4.5.1 Extremely fast network recovery (EFR)

EFR is one of the most outstanding technical features of the RAPIEnet protocol. When a link fault is detected in a RAPIEnet ring network, the fault link information is broadcast to every device on the network. Every device updates its own path table and tries to find a new path around the fault. RAPIEnet provides extremely fast recovery time from topology changes to provide redundant network connectivity. The link status information is monitored by the hardware-triggered signal event (see 7.8.4), and the link status change information is broadcast to every device on the network.

4.5.2 Plug and play (PnP)

When a new device joins an existing network, the new link information is broadcast to every device on the network. The new device also collects existing link information from each device so that it can communicate to the other nodes on the network without manual configuration.

4.5.3 Network management information base (NMIB) management

A RAPIEnet device automatically collects and maintains its network management information base (NMIB), including network information and path table (see 8.3.6).

Every device on the same network shares and gathers link information on the network to update its network information and path table. Every device updates its network information and path table when it receives link status change information.

4.5.4 Automatic network configuration (ANC)

To support EFR functions in both ring and line networks, RAPIEnet does not restrict the dynamic change in network topology. RAPIEnet also supports automatic network configuration. When the network topology is changed, the changed network information is shared with every device on the network, and then every device updates its own network information and path table. RNMs or LNMs are automatically selected on the network according to the MAC address and data link address (see Annex A).

4.6 Application layer overview

Industrial automation and process control systems consist of primary automation devices (for example, sensors, actuators, local display devices, annunciators, programmable logic controllers, small single-loop controllers, and stand-alone field controls) as well as control and monitoring equipment.

The data transfer between these devices takes place in a peer-to-peer or multicast communication manner.

Figure 8 illustrates the interaction between the RAPIEnet fieldbus application layer (FAL) and the DLL. RAPIEnet supports cyclic and acyclic data transfer for its own application processes. RAPIEnet can also be used in parallel with TCP/IP or UDP communication. The use of other standard communication protocols is outside the scope of this PAS.

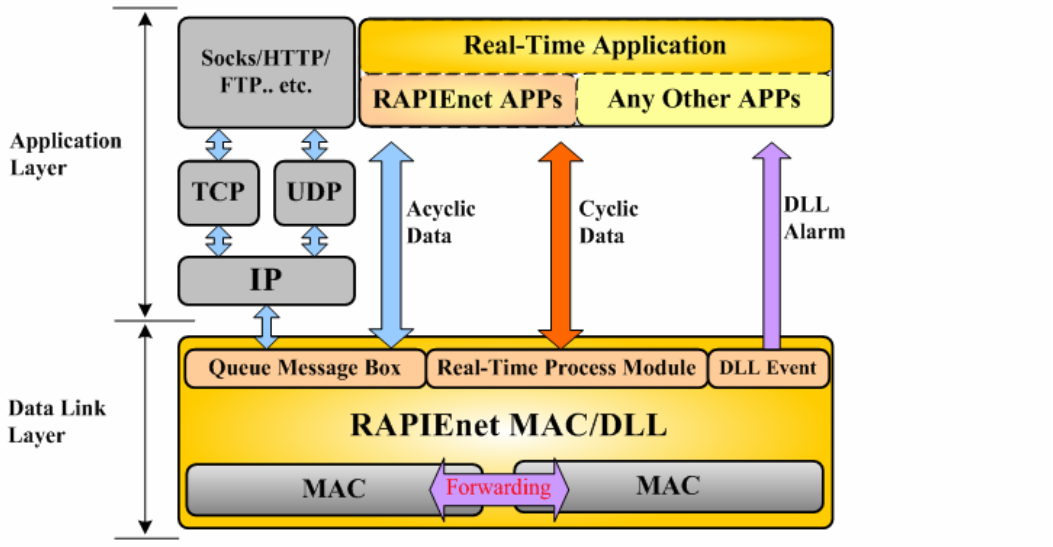


Figure 8 – Interaction between FAL and DLL

RAPIEnet supports the publisher-subscriber communication model for cyclic data sharing as shown in Figure 9. The publisher periodically multicasts preconfigured data, and subscribers receive that data. This cyclic data sharing is the most widely used model in industrial applications.

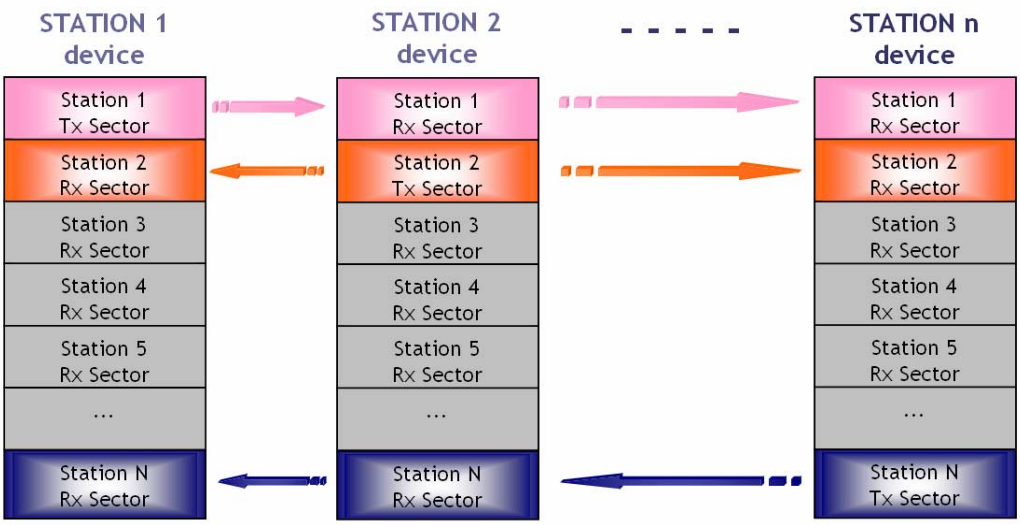


Figure 9 – Publisher-subscriber communication model

RAPIEnet supports the client-server communication model for event-triggered data transfer as shown in Figure 10. The client requests data invoked by the internal or external events. The server replies with the data requested. This can be used for event-triggered or user-triggered application processes.

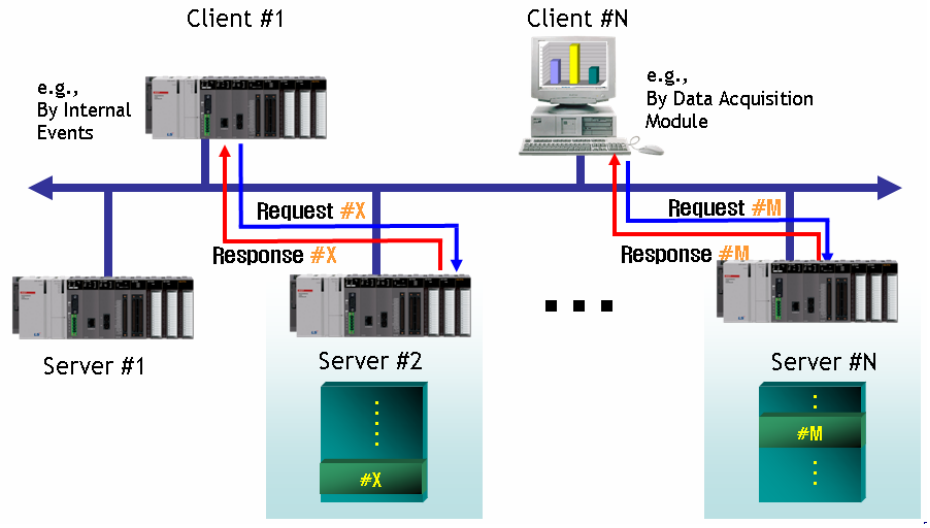


Figure 10 – Client-server communication model

5 Physical layer

5.1 Overview

The RAPIEnet physical layer corresponds to the ISO/IEC 8802-3:2002 access method and physical layer PASs. For RAPIEnet network systems, the following full-duplex physical layer technologies are used.

a) 100BASE-TX;

b) 100BASE-FX;

c) 1000BASE-T (IEEE 802.3:2005);

d) 1000BASE-X (IEEE 802.3:2005).

NOTE Recently proposed physical layer technologies, such as 10 Gigabit Ethernet or 100 Gigabit Ethernet, can also be used for the RAPIEnet physical layer because RAPIEnet is designed to be independent of the physical layer technology.

In a RAPIEnet network, any combination of these physical layer technologies may be used. Changeovers from one physical layer to another are supported.

5.2 100BASE-TX

100BASE-TX is an electrical physical layer system specified in ISO/IEC 8802-3. It uses two pairs of Category 5 balanced cabling as specified by ISO/IEC 11801.

5.3 100BASE-FX

100BASE-FX is an optical fibre physical layer system specified in ISO/IEC 8802-3. It uses two transmission technologies over fibre.

5.4 1000BASE-T

1000BASE-T is an electrical physical layer system specified in IEEE 802.3-2005. It uses four pairs of Category 5 balanced cabling as specified by ISO/IEC 11801. Category 6 cable may also be used.

5.5 1000BASE-X

1000BASE-X is an optical fibre physical layer system specified in IEEE 802.3-2005. It uses four transmission technologies over fibre.

6 Data link layer service definitions

6.1 Introduction

This section defines the RAPIEnet data link layer (DLL) services. This is to be one of the real-time Ethernet (RTE) specifications to facilitate the interconnection of automation system components. RAPIEnet data link services provide reliable and transparent data communication among RAPIEnet data link users through the logical interface between the ISO/IEC 8802-3 physical layer and the DLL.

The Data link Service is provided by the Data link Protocol that uses the services available from the physical layer. This section defines the Data link Service characteristics that the higher level protocol may use immediately. The Data link Protocol defines some procedures for sharing and maintaining the network information in a RAPIEnet network.

6.2 Scope

6.2.1 Overview

This subclause provides the common elements for basic time-critical messaging communications between devices in an automation environment. The term “time-critical” in this context means the prioritized full-duplex collision-free time-deterministic communication, of which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the required time risks the failure of the applications requesting the actions, with attendant risk to equipment, plant, and possibly human life.

This PAS defines in an abstract way the externally visible service provided by the RAPIEnet DLL in terms of

a) primitive actions and events of the service;

b) parameters associated with each primitive action and event, and the form that they take;

c) interrelationships between these actions and events, and their valid sequences.

The purpose of this PAS is to define the services provided to

a) the RAPIEnet application layer at the boundary between the application and DLLs of the fieldbus reference model;

b) systems management at the boundary between the DLL and the systems management of the Fieldbus reference model.

6.2.2 Specifications

The principal objective of this PAS is to specify the characteristics of conceptual DLL services suitable for time-critical communications, and to supplement the OSI Basic Reference Model in guiding the development of data link protocols for time-critical communications. A secondary objective is to provide migration paths from previously existing industrial communications protocols.

This PAS may be used as the basis for formal data link programming interfaces. Nevertheless, it is not a formal programming interface, and any such interface will need to address implementation issues not covered by this PAS, including

a) the sizes and octet ordering of various multi-octet service parameters;

b) the correlation of paired primitives for request and confirm, or indication and response.

6.2.3 Conformance

This PAS neither describes individual implementations of products, nor does it constrain the implementations of data link entities in industrial automation systems.

There is no conformance of equipment to this data link layer service definition specification. Instead, conformance is achieved through implementation of the corresponding data link protocol that fulfils the RAPIEnet DLL services defined in this PAS .

6.3 Normative references

The following 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, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*

ISO/IEC 7498-3, *Information technology – Open Systems Interconnection – Basic Reference Model: Naming and addressing*

ISO/IEC 8802-3, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer PASs*

ISO/IEC 10731:1994, *Information technology – Open Systems Interconnection – Basic Reference Model – Conventions for the definition of OSI services*

6.4 Terms, definitions, symbols, abbreviations, and conventions

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

6.4.1 Reference model terms and definitions

This PAS is based in part on the concepts developed in ISO/IEC 7498-1 and ISO/IEC 7498-3, and makes use of the following terms defined therein.

DL-address [7498-3]

DL-address-mapping [7498-1]

called-DL-address [7498-3]

calling-DL-address [7498-3]

centralized multi-end-point-connection [7498-1]

DL-connection [7498-1]

DL-connection-end-point [7498-1]

DL-connection-end-point-identifier [7498-1]

DL-connection-mode transmission [7498-1]

DL-connectionless-mode transmission [7498-1]

correspondent (N)-entities [7498-1]
correspondent DL-entities (N=2)
correspondent Ph-entities (N=1)

DL-duplex-transmission [7498-1]

(N)-entity [7498-1]
DL-entity (N=2)
Ph-entity (N=1)

DL-facility [7498-1]

flow control [7498-1]

(N)-layer [7498-1]
DL-layer (N=2)
Ph-layer (N=1)

layer-management [7498-1]

DL-local-view [7498-3]

DL-name [7498-3]

naming-(addressing)-domain [7498-3]

peer-entities [7498-1]

primitive name [7498-3]

DL-protocol [7498-1]

DL-protocol-connection-identifier [7498-1]

DL-protocol-data-unit [7498-1]

DL-relay [7498-1]

Reset [7498-1]

responding-DL-address [7498-3]

Routing	[7498-1]
Segmenting	[7498-1]
(N)-service DL-service (N=2) Ph-service (N=1)	[7498-1]
(N)-service-access-point DL-service-access-point (N=2) Ph-service-access-point (N=1)	[7498-1]
DL-service-access-point-address	[7498-3]
DL-service-connection-identifier	[7498-1]
DL-service-data-unit	[7498-1]
DL-simplex-transmission	[7498-1]
DL-subsystem	[7498-1]
systems-management	[7498-1]
DLS-user-data	[7498-1]

6.4.2 Service convention terms and definitions

This PAS also makes use of the following terms defined in ISO/IEC 10731 as they apply to the DLL.

acceptor

asymmetrical service

confirm (primitive);
requestor.deliver (primitive)

deliver (primitive)

DL-confirmed-facility

DL-facility

DL-local-view

DL-mandatory-facility

DL-non-confirmed-facility

DL-protocol-machine

DL-provider-initiated-facility

DL-provider-optional-facility

DL-service-primitive;
primitive

DL-service-provider

DL-service-user

DLS-user-optional-facility

indication (primitive);
acceptor.deliver (primitive)

multi-peer

request (primitive);
requestor.submit (primitive)

requestor

response (primitive);
acceptor.submit (primitive)

submit (primitive)

symmetrical service

6.4.3 Data link service terms and definitions

DL-segment, link, local link

Single data link (DL) subnetwork in which any of the connected data link entities (DLEs) may communicate direct, without any intervening data link relaying, whenever all of those DLEs that are participating in an instance of communication are simultaneously attentive to the DL-subnetwork during the period(s) of attempted communication.

Data link service access point (DLSAP)

Distinctive point at which DL-services are provided by a single DLE to a single higher layer entity.

NOTE This definition, derived from ISO/IEC 7498-1, is repeated here to facilitate understanding of the critical distinction between DLSAPs and their DL-addresses.

DLSAP address

Either an individual DLSAP address designating a single DLSAP of a single data link service (DLS) user (DLS-user), or a group DL-address potentially designating multiple DLSAPs, each of a single DLS-user.

NOTE This terminology was chosen because ISO/IEC 7498-3 does not permit the use of the term DLSAP-address to designate more than a single DLSAP at a single DLS-user.

(individual) DLSAP-address

DL-address that designates only one DLSAP within the extended link

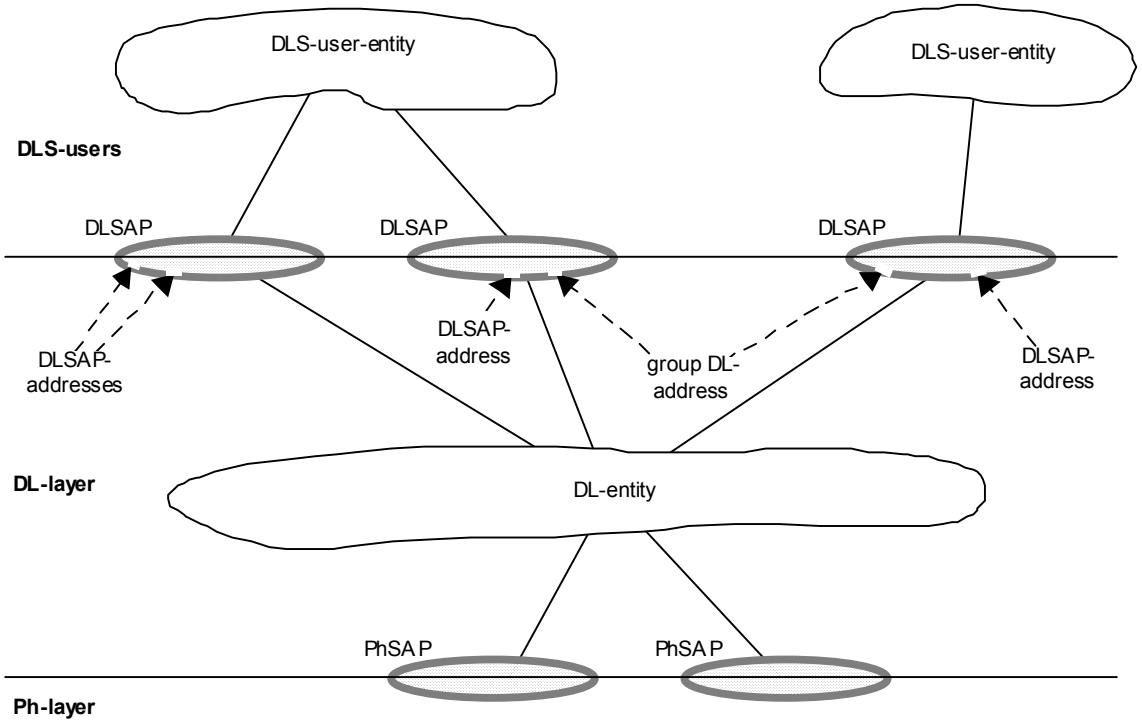
NOTE A single DL-entity may have multiple DLSAP-addresses associated with a single DLSAP.

Data link connection endpoint address (DLCEP-address)

DL-address that designates either

a) one peer DL-connection-end-point;

b) one multi-peer publisher DL-connection-end-point, and implicitly the corresponding set of subscriber DL-connection-end-points, where each DL-connection-end-point exists within a distinct DLSAP and is associated with a corresponding distinct DLSAP-address.



NOTE 1 DLSAPs and physical layer service access points (PhSAPs) are depicted as ovals spanning the boundary between two adjacent layers.

NOTE 2 DL-addresses are depicted as designating small gaps (points of access) in the DLL portion of a DLSAP.

NOTE 3 A single DLE may have multiple DLSAP-addresses and group DL-addresses associated with a single DLSAP.

Figure 11 – Relationships of DLSAPs, DLSAP-addresses, and group DL-addresses

Frame check sequence (FCS) error

Error that occurs when the computed frame check sequence value after reception of all the octets in a data link protocol data unit (DLPDU) does not match the expected residual.

frame

Synonym for DLPDU

network management

Management functions and services that perform network initialization, configuration, and error handling

protocol

Convention on the data formats, time sequences, and error correction for data exchange in communication systems.

receiving DLS-user

DL-service user that acts as a recipient of DLS-user data.

NOTE A DL-service user can be both a sending and receiving DLS-user concurrently.

sending DLS-user

DL-service user that acts as a source of DLS-user data.

device

single DLE as it appears on one local link.

DL-address

address that designates the (single) DLE associated with a single device on a specific local link. DL-address values are in the range 0-255. DL-addresses may be provided by hardware settings (for example, rotary switch) or set by software

device unique identification

unique 8-byte identification to identify a RAPIEnet device in a network. This ID is a combination of a 6-byte ISO/IEC 8802-3 MAC address and 2-byte DL-address

ring

active network where each node is connected in series to two other devices

NOTE A ring may also be referred to as a loop.

linear topology

topology where the devices are connected in series, with two devices each connected to only one other device, and all others each connected to two other devices, i.e., connected in a line

R-port

port in a communication device that is part of a ring structure

real-time

ability of a system to provide a required result in a bounded time

real-time communication

transfer of data in real-time

real-time Ethernet (RTE)

ISO/IEC 8802-3 based network that includes real-time communication

NOTE 1 Other communications can be supported, providing that the real-time communication is not compromised.

NOTE 2 This definition is based on, but not limited to, ISO/IEC 8802-3. It could be applicable to other IEEE 802 PASs, for example, IEEE802.11.

RTE end device

device with at least one active RTE port

RTE port

media access control (MAC) sublayer point where an RTE is attached to a local area network (LAN)

NOTE This definition is derived from that of bridge port in ISO/IEC 10038:1993, as applied to local MAC bridges.

switched network

network also containing switches.

NOTE Switched network means that the network is based on IEEE 802.1D and IEEE 802.1Q with MAC bridges and priority operations.

link

transmission path between two adjacent nodes [derived from ISO/IEC 11801].

6.4.4 Symbols and abbreviations

6.4.4.1 Common symbols and abbreviations

Term or abbreviation	Definition or meaning
DL	data link (used as a prefix or adjective)
DLC	data link connection
DLCEP	data link connection endpoint
DLE	data link entity (the local active instance of the DLL)
DLL	data link layer
DLPDU	data link protocol data unit
DLPM	data link protocol machine
DLM	data link management
DLME	data link management entity (the local active instance of DLM)
DLMS	data link management service
DLS	data link service
DLSAP	data link service-access-point
DLSDU	data link service-data-unit
FIFO	first-in, first-out (queuing method)
NMT	network management
OSI	Open Systems Interconnection
Ph-	physical layer (as a prefix)
PHY	physical interface transceiver
PhL	physical layer
RTE	Real-time Ethernet
IEC	International Electrotechnical Commission
IP	Internet protocol (see RFC 791)

ISO	International Organization for Standardization
MAC	media access control
NRT	non-real-time
PDU	protocol data unit
SAP	service access point
RT	real-time
TCP	Transmission Control Protocol (see RFC 793)
UDP	User Datagram Protocol (see RFC 768)

6.4.4.2 RAPIEnet: Additional symbols and abbreviations

Term or abbreviation	Definition or meaning
EFR	extremely fast recovery
GD	general device
LNM	line network manager
PO	power on
PnP	plug and play
RNM	ring network manager
RNMP	primary ring network manager
RNMS	secondary ring network manager
RNAC	ring network auto configuration
UID	device unique identification
RAPIEnet NMIB	RAPIEnet network management information base

6.4.5 Conventions

6.4.5.1 Common conventions

This PAS uses the descriptive conventions given in ISO/IEC 10731. The service model, service primitives, and time-sequence diagrams used are entirely abstract descriptions; they do not represent a specification for implementation. Service primitives, used to represent service user/provider interactions (see ISO/IEC 10731), convey parameters that indicate information available in the user/provider interaction.

This PAS uses a tabular format to describe the component parameters of the DLS primitives. The parameters that apply to each group of DLS primitives are set out in tables throughout the remainder of this specification. Each table consists of up to six columns, containing the name of the service parameter, and a column for each of those primitives and parameter-transfer directions used by the DLS, including

a) the request primitive's input parameters;

b) the request primitive's output parameters;

c) the indication primitive's output parameters;

d) the response primitive's input parameters;

e) the confirmation primitive's output parameters.

NOTE The request, indication, response and confirmation primitives are also known as requestor.submit, acceptor.deliver, acceptor.submit, and requestor.deliver primitives, respectively (see ISO/IEC 10731).

One parameter, or a portion of it, is listed in each row of each table. Under the appropriate service primitive columns, the following code is used to specify how the parameter is used, and its direction.

M parameter: mandatory for the primitive

U parameter: a user option that may or may not be provided depending on the dynamic use of the DLS-user. When not provided, a default value for the parameter is assumed

C parameter is conditional upon other parameters or upon the environment of the DLS-user

(Blank) parameter is never present.

Some entries are further qualified by items in parentheses. These may be one of

a) (=) a parameter-specific constraint indicating that the parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table;

b) (n) an indication that following note n contains additional information pertaining to the parameter and its use.

In any particular interface, not all parameters shall be stated explicitly. Some may be implicitly associated with the DLSAP at which the primitive is issued.

In the diagrams illustrating these interfaces, dashed lines indicate cause and effect or time sequence relationships, and wavy lines indicate that events occur at approximately the same time.

6.4.5.2 Additional conventions

In the diagrams illustrating the DLS and DLM interfaces, dashed lines indicate cause and effect or time sequence relationships between actions at different stations, while solid lines with arrows indicate cause and effect time sequence relationships that occur within the DLE provider at a single station.

The following notation, a shortened form of the primitive classes defined in 6.4.5.1, is used in the figures and tables.

req: request primitive

ind: indication primitive

cnf: confirmation primitive (confirmation)

res: response primitive

6.5 Data link service and concept

6.5.1 Overview

This PAS specifies the RAPIEnet data link services for an ISO/IEC 8802-3 based time-deterministic control network, which is one of the communication networks for RTE. The communication services support timing demands typical of high-performance automation

applications. They do not change the basic principles of ISO/IEC 8802-3, but extend it toward RTE. Thus, it is possible to continue to use standard Ethernet hardware, infrastructure components, or test and measurement equipment, such as network analysers.

The RAPIEnet DLL provides reliable and transparent data communication between two RAPIEnet end devices. The RAPIEnet DLL also guarantees abstract transparent data transfer between DL-users so that DLL provides flexible and convenient network connectivity to network users.

6.5.1.1 Overview of full duplex flow control

A RAPIEnet device is based on an integrated switch with two ports (ring ports) connected to the ring. Therefore, a RAPIEnet network system is made up of full-duplex, collision-free switching devices configured as a ring or a line network. Figure 12 shows the full-duplex flow control procedure in a RAPIEnet network system. RAPIEnet guarantees collision-free data transmission between two devices linked by a full-duplex Ethernet connection so that the RAPIEnet DLL provides reliable, transparent, and collision-free data transmission to the DLS-users.

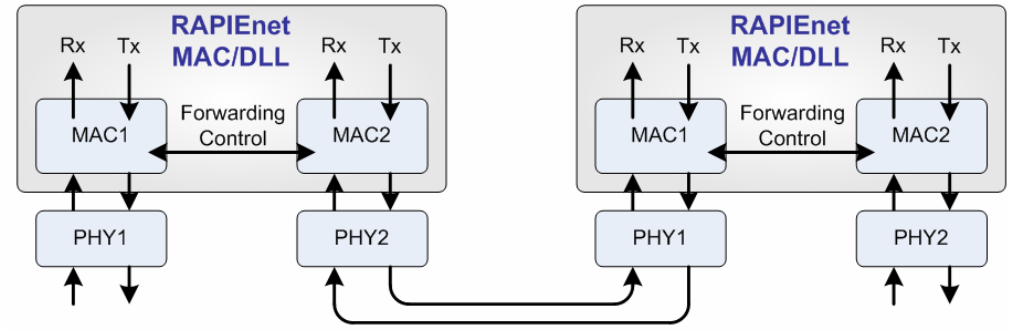


Figure 12 – Full-duplex flow control

6.5.1.2 Types and classes of DL-layer service

The DLS provides transparent and reliable data transmission between DLS-users over RAPIEnet. The DLS is based on services provided by the physical layer of ISO/IEC 8802-3 to the conceptual interface between the physical and data link layers.

Three types of data transmission services are provided.

Data service (DL-DATA)

Data service is used to transmit a RAPIEnet frame to a destination device or devices using the priority option. DL-DATA service is a queued service using the RT-queue.

Sporadic data service (DL-SPDATA)

Sporadic data service is used to transmit a common protocol frame, such as TCP/IP or UDP. RAPIEnet data link layer transmits without modification any received DLSDUs generated by a DLS-user. In this case, DLSDU is assumed to include DLPDU. DL-SPDATA is a queued service using the NRT-queue.

Network control message

Network-control-message service is used by the DL-management entity to share network-related information with the other devices in a RAPIEnet network segment.

6.5.1.2.1 Primitives of the data service`

The sequence of primitives for the data service is shown in Figure 13.

DL-DATA request and DL-DATA indication correspond to the MA-DATA request and MA-DATA indication defined by ISO/IEC8802-3, respectively.

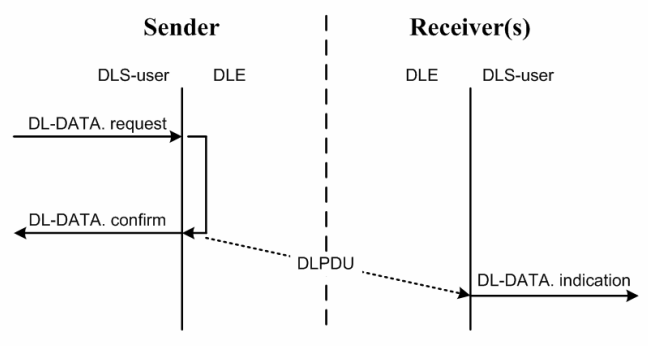


Figure 13 – Sequence diagram of DL-DATA service

The sender DLS-user prepares a DLSDU for a single receiver-side DLS-user, or for multiple DLS-users. The DLSDU is passed to the local DLE via the DLS interface by means of a DL-DATA request primitive. The DLE queues the service request, and the queued service request is transmitted by the DLPM to the receiver DLE or to multiple DLEs.

The receiving DLE(s) attempt to deliver the received DLSDU to the specified DLS-user(s).

There is no confirmation of correct receipt at the remote DLEs or of delivery to the intended DLS-user(s); acknowledgements do not occur. When the DLSDU is transmitted, it reaches all receiver-side DLEs at about the same time, ignoring signal propagation delays. Each DLE addressed by the DLSDU that has received the data error-free, passes the DLSDU and associated addressing information to the local DLS-user by means of a DL-DATA indication primitive.

6.5.1.2.2 Primitives of the sporadic data service

The sequence of primitives for the sporadic data service is shown in Figure 14. DL-SPDATA request and DL-SPDATA indication correspond to the MA-DATA request and MA-DATA indication defined by ISO/IEC8802-3, respectively.

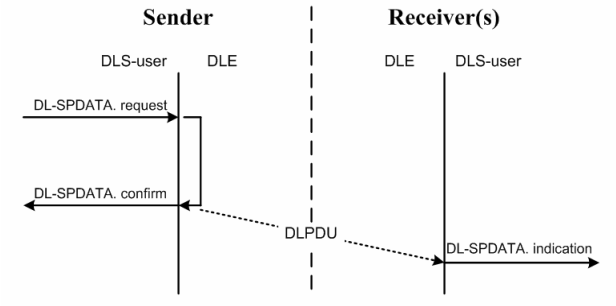


Figure 14 – Sequence diagram of DL-SPDATA service

DL-SPDATA service is used to transmit other protocol frames, such as TCP/IP or UDP. DL-SPDATA service is transmitted through both R-ports using the non-real-time (NRT) queue without referring to the path table and without modification of the received DLSDU.

6.5.1.2.3 Primitives of the network control message service

The sequence of primitives for the network control message service is shown in Figure 15. DL-NCM_SND request and DL-NCM_SND indication correspond to the MA-DATA request and MA-DATA indication defined by ISO/IEC 8802-3, respectively.

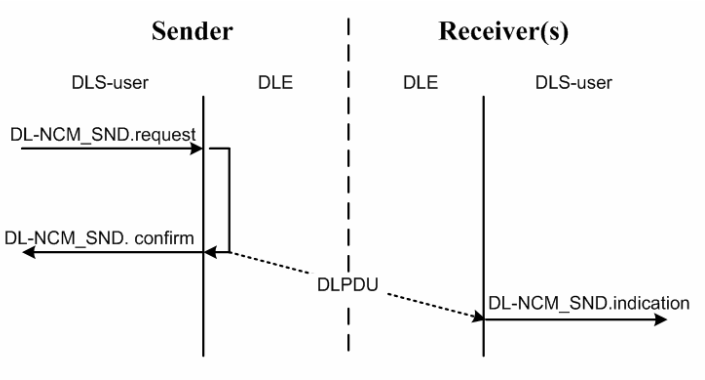


Figure 15 – Sequence diagram of NCM service primitive

The DL-NCM_SND service is used to transmit a network control message. The DL-NCM_SND service is transmitted to one or both R-ports through the real-time (RT) queue.

6.5.1.3 Addressing

Each DLE on the link is designated by a DL-address. The range of individual DL-addresses is limited, from 0 to a maximum of 255. Table 1 shows the DL-address assignment (see 8.4.3.3.2).

The DL-address 0xFFFF is used for broadcast message. This DL-address is either configured by the application process or is set on the device (for example, using address switches).

Table 1 – Destination DL-address

Field Name	Position	Value/Description
Destination DL-address	b15 – b0	0xFFFF: broadcast address 0xFFFE: network control address (C_NCM_ADDR) 0xFFFD – 0xFFDE: user-defined multicast address 0xFFDD: invalid address 0x0100 – 0xFFDC: reserved 0x0000 – 0x00FF: regular RAPIEnet address

6.5.1.3.1 Broadcast address

If the destination DL-address is 0xFFFF, the destination MAC address field contains the ISO/IEC 8802-3 broadcast MAC address (see 8.4.3.3.2.1).

6.5.1.3.2 Network control address

If the destination DL-address is C_NCM_ADDR, the destination MAC address field contains C_NCM_MAC_ADDR. (see 8.4.3.3.2.2) However, NCM_LINK_ACTV and NCM_ADV_THIS messages are transmitted using C_NCM_ADDR as destination address (see 8.4.3.3.2.2).

NOTE C_NCM_ADDR cannot be accessed by the DLS-user.

6.5.1.3.3 User-defined multicast address

A user-defined multicast address may be used to indicate multiple recipients. This PAS does not restrict the use of the user-defined multicast address and it is not a mandatory feature in RAPIEnet (see 8.4.3.3.2.3).

6.5.2 Detailed description of the data service

6.5.2.1 General

DL-DATA request and DL-DATA indication correspond to the MA-DATA request and MA-DATA indication defined by ISO/IEC8802-3, respectively.

DL-DATA service provides 1:1 or 1:N data transmission in a RAPIEnet segment. DL-DATA service is used by the DLS-user to send a DLSDU to a single peer-end device or multiple peer-end devices. The DL-DATA service is processed by the priority option indicated in the DL-DATA request primitive. Figure 16 shows the DL-DATA service procedure.

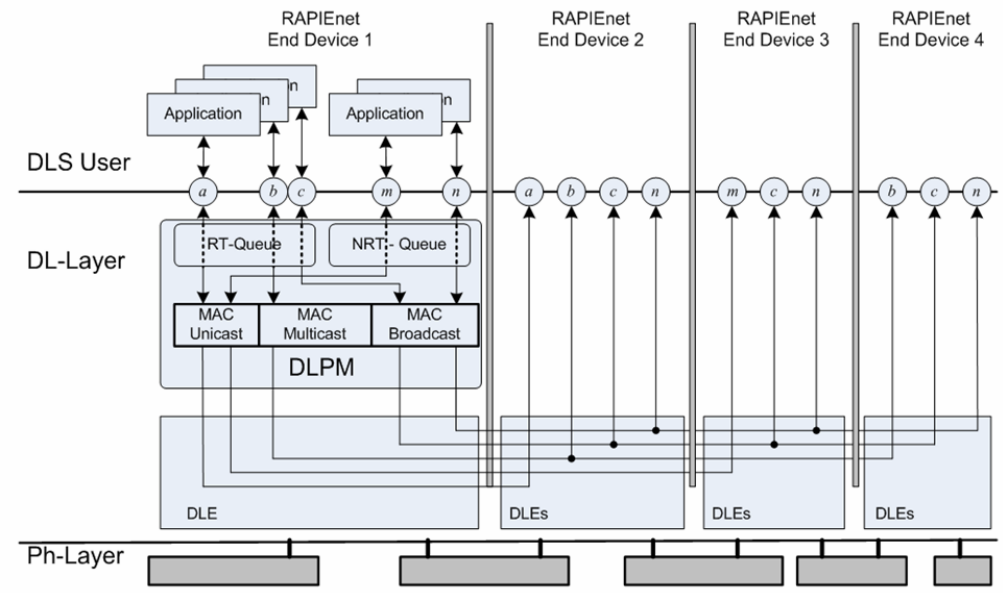


Figure 16 – DL-DATA service

6.5.2.2 Sequence of primitives

The data service primitives and their associated parameters are summarized in Table 2 and the primitive sequence is shown in Figure 13.

Table 2 – Primitives and parameters used in DL-DATA service

Function	Location	Primitive	Direction	Parameters
Send data	Sender	DL-DATA request	To DLE	DST_addr Priority DSAP SSAP Group mask Group mask length DLSDU DLSDU length

Function	Location	Primitive	Direction	Parameters
Send data confirmation to calling DLS-user	Sender	DL-DATA confirm	From DLE	Status
Receive data	Receiver(s)	DL-DATA indication	From DLE	DST_addr SRC_addr SSAP DLSDU DLSDU length

6.5.2.3 Transmit/receive data

6.5.2.3.1 Function

DL-DATA service primitives allow the DLS-user to transfer message data to a single peer DLS-user or multiple peer DLS-users at remote devices.

6.5.2.3.2 Types of primitives and the parameters

Table 3 indicates the parameters of DL-DATA service.

Table 3 – DL-DATA primitives and parameters

DL-DATA	Request	Indication	Confirm
Parameter	Input	Output	Output
DST_addr	M	M(=)	-
SRC_addr	-	M	-
Priority	M	M(=)	-
DSAP	M	-	-
SSAP	U	U(=)	-
Group mask	U	-	-
Group mask length	U	-	-
DLSDU	M	M(=)	-
DLSDU length	M	M(=)	-
Status	-	-	M

6.5.2.3.2.1 DST_addr

This parameter indicates the destination DL-address of the DLE(s) for which the DLPDU is intended. It may be an individual or multicast (including broadcast) DL-address. It should be noted that this is the DL-address, not the MAC address (see Table 1).

6.5.2.3.2.2 SRC_addr

This parameter indicates the source DL-address of an individual DLE, which sends the DLPDU.

6.5.2.3.2.3 Destination service access point (DSAP)

This parameter indicates the destination service access point of the DLE for which the DLPDU is intended. The DSAP address is not reserved for any particular application.

6.5.2.3.2.4 Priority

This parameter indicates the message priority of the DL-DATA service request and the priority of the frame in the RT-queue. A DLS-user can indicate the message priority of a DL-DATA service request according to the application.

6.5.2.3.2.5 Source service access point (SSAP)

This parameter indicates the source service access point of the DLE from which the DLPDU is being sent. The source SAP address is not reserved for any particular application.

6.5.2.3.2.6 Data link service data unit (DLSDU)

This parameter specifies the information that is transferred from local DLS-user to the remote DLS-user.

6.5.2.3.2.7 DLSDU length

This parameter indicates the length of DLSDU in bytes.

6.5.2.3.2.8 Status

This parameter allows the DLS-user to determine whether the requested service was provided successfully. If it failed, the reason is specified. The value of this parameter can be one of

a) “success – successfully completed”;

b) “failure – invalid requested parameter”;

c) “failure – RT-queue is full”;

d) “failure – unavailable destination”

6.5.3 Detailed description of the sporadic data service

6.5.3.1 General

This DLS provides unacknowledged data transmission between single DLSAPs or unacknowledged data transmission from a single DLSAP to a group of DLSAPs on the extended link. DL-SPDATA request and DL-SPDATA indication correspond to the MA-DATA request and MA-DATA indication defined by ISO/IEC 8802-3, respectively. The DL-SPDATA service allows the DLS-user to transfer non-RAPIEnet message data to a single-peer DLS-user or multiple-peer DLS-users at remote devices. The DL-SPDATA service is processed through the NRT-queue. A DLSDU from a DLS-user is not modified and is transmitted to both R-ports without referring to the path table.

6.5.3.2 Sequence of primitive

The sporadic message data service primitives and the parameters are summarized in Table 4 and the primitive sequence is shown in Figure 14.

Table 4 – Primitives and parameters used in DL-SPDATA service

Function	Location	Primitive	Direction	Parameters
Send Sporadic Data	Sender	DL-SPDATA request	To DLE	DLSDU DLSDU length
Confirmation to calling DLS-user	Sender	DL-SPDATA confirm	From DLE	Status
Receive Sporadic Data	Receiver(s)	DL-SPDATA indication	From DLE	DLSDU DLSDU length

6.5.3.3 Transmit/receive sporadic data

6.5.3.3.1 Function

DL-SPDATA service primitives allow the DLS-user to transfer message data to a single-peer DLS-user or multiple-peer DLS-users at remote nodes through the two R-ports.

6.5.3.3.2 Type of primitives and the parameters

Table 5 indicates the parameters of sporadic data service.

Table 5 – DL-SPDATA primitives and parameters

DL-SPDATA	Request	Indication	Confirm
Parameter	Input	Output	Output
DLSDU	M	M(=)	-
DLSDU length	M	M(=)	-
Status	-		M

6.5.3.3.2.1 Data link service data unit (DLSDU)

This parameter indicates the DLSDU generated by the DLS-user.

6.5.3.3.2.2 DLSDU length

This parameter indicates the length of the DLSDU.

6.5.3.3.2.3 Status

This parameter allows the DLS-user to determine whether or not the requested service was provided successfully. If it failed, the reason is specified. The value of this parameter can be one of

a) “success – successfully completed”;

b) “failure – invalid requested parameter”;

c) “failure – the NRT-queue is full”.

6.5.4 Detailed description of network control message service

6.5.4.1 General

The network control message service is used by the DLM to share network-related information with the other devices in a RAPIEnet network segment. The network control message service is a local device information transfer service initiated by the DLM.

6.5.4.2 Sequence of primitives

The network control message data service primitives and the parameters are summarized in Table 6, and the primitive sequence is shown in Figure 15.

Table 6 – Primitives and parameters used on DL-NCM_SND service

Function	Location	Primitive	Direction	Parameters
Send Network Control Message	DLM	DL-NCM_SND request	To DLPM	DST_addr NCMT DLMDU

				DLMDU length R-port
Confirmation to calling DLM	DLPM	DL-NCM_SND confirm	To DLM	Status
Receive Network Control Message	DLPM	DL-NCM_SND indication	To DLM	DST_addr SRC_addr NCMT DLMDU DLMDU length R-port

6.5.4.3 Transmit/receive network control message

6.5.4.3.1 Function

DL-NCM_SND service primitives allow the DLS-user to transfer network control messages to a single peer DLS-user or multiple peer DLS-users at remote nodes through the R-port.

6.5.4.3.2 Type of primitives and parameters

Table 7 indicates the parameters of the network control message service.

Table 7 – DL-NCM_SND primitives and parameters

DL-NCM_SND	Request	Indication	Confirm
Parameter	input	output	output
DST_addr	M	M(=)	-
SRC_addr		M	-
NCMT	M	M(=)	-
DLMDU	M	M(=)	-
DLMDU length	M	M(=)	-
R-port	M	M	-
Status	-		M

6.5.4.3.2.1 DST_addr

This parameter indicates the destination DL-address of the DLE(s) for which the DLPDU is intended. When the network control message (NCM) type is NCM_LINK_ACTV or NCM_ADV_THIS, the destination DL-address is set to C_NCM_ADDR (see 8.4.3.3.2.2). When the network control message type is NCM_LINE_START or NCM_RING_START, the destination DL-address is set to the broadcast address (see 8.4.3.3.2.1).

6.5.4.3.2.2 SRC_addr

This parameter indicates the source DL-address of an individual DLE from which the data link management data unit (DLMDU) is being sent.

6.5.4.3.2.3 Network control message type (NCMT)

This parameter indicates the type of network control message. Table 8 describes the NCMT.

Table 8 – Summary of Network Control Message Type

Network Control Message Type	Description
NCM_LINK_ACTV	This message type indicates that a new RAPIEnet link has been established through an R-port. This network control message is transmitted through the newly activated R-port DLM detects the status of each R-port using the Ph-LINK_STATUS_CHANGE and Ph-GET_LINK_STATUS services. When an R-port's status is changed from link inactive to link active, local device information is transmitted through the newly activated R-port
NCM_ADV_THIS	NCM_ADV_THIS message is used to transmit the recipient's local device information when the recipient receives the NCM_LINK_ACTV message from the newly linked device. This message is transmitted through the R-port, which is used to receive the NCM_LINK_ACTV message
NCM_LINE_START	This message is used to broadcast that the network topology has been automatically configured as a line network. This message is initiated by the DLM whose state has changed to LNM when the existing line network is divided into two line networks or when a link failure is detected in a ring network, and the network is reconfigured as a line network
NCM_RING_START	This message is used to broadcast that the network topology has been automatically configured as a ring network. This message is initiated and transmitted through both R-ports by the DLM whose state has changed to RNMP
NCM_ACK_RNMS	This message is used by the RNMS device to advise that the RNMS has been successfully selected. NCM_ACK_RNMS message is unicast from the RNMS to the RNMP

6.5.4.3.2.4 R-port

This parameter indicates the RAPIEnet MAC port to send or receive a frame.

6.5.4.3.2.5 Data link Management Data Unit (DLMDU)

This parameter indicates the DLMDU. Local device information is used as the DLMDU in network control messages.

6.5.4.3.2.6 DLMDU length

This parameter indicates the length of the DLMDU in bytes.

6.5.4.3.2.7 Status

This parameter allows the DLMS-user to determine whether the requested service was provided successfully. If it failed, the reason is specified. The value of this parameter can be one of

a) "ok – success – network control message is successfully completed";

b) "failure – invalid parameters in the request".

6.6 Data link management services

6.6.1 General

This section describes the interface between a DLE and a data link management system user (DLMS-user). The services of this interface are required for the protocol that implements the DLS specified in 7.5

6.6.2 Data link management service (DLMS) facilities

The DLMS provides the facilities for the initialization, configuration, event and error handling between the DLMS-user and the logical functions in the DLE. The following functions are provided to the DLMS-user.

a) Reset the local DLE.
b) Request for and modification of the actual operating parameters and of the counters of the local DLE.
c) Notification of unexpected events, errors, and status changes, both local and remote.
d) Request for identification of the local DLE, and its DLSAP configuration.

6.6.3 Data link management service (DLMS)

6.6.3.1 Overview

The DLMS provides the following services to the DLMS-user.

a) Reset
b) Set-value
c) Get-value
d) SAP-allocation
e) SAP-deallocation
f) Get-SAP-information
g) Get-diagnostic-information
h) Event
i) Get-path

The services Reset, Set-value, Get-value, SAP-allocation, SAP-deallocation, Get-diagnostic-information, Event, and Get-path are considered mandatory. The others are optional.

6.6.3.2 Reset

The DLMS-user uses this service to cause the DLM to reset the DLE. Reset is equivalent to power on. The DLMS-user receives a confirmation of the reset.

6.6.3.3 Set-value

The DLMS-user uses this service to assign new values to the DLE variables. The DLMS-user receives a confirmation of whether or not the specified variables have been set to the new values.

6.6.3.4 Get-value

This service enables the DLM to read DLE variables. The response includes the actual value of the specified variables.

6.6.3.5 SAP-allocation

The SAP-allocation service is used by the DLMS-user to obtain a SAP assignment from the DLM. To receive a service request from the peer device and deliver the received DLSDU to the corresponding DLS-user, the destination DLS-user should obtain a SAP assignment before the service transaction. However, the SSAP does not have to be assigned beforehand because the default SSAP value of 0 is used for normal DL services. An SSAP might be assigned for special application functions.

6.6.3.6 SAP-deallocation

The SAP-deallocation service is used by the DLMS-user to release and return the allocated SAP to DLM. The deallocated SAP can be used again using the SAP-allocation service.

6.6.3.7 Get-SAP information

The Get-SAP information service is used by the DLMS-user to obtain information from the DLM about the allocated SAP.

6.6.3.8 Get-diagnostic-information

The Get-diagnostic information service is used by the DLMS-user to obtain current diagnostic information about the local device, remote devices, and the network.

6.6.3.9 Event

DLM employs this service to inform the DLMS-user about certain events or errors in the DLL.

6.6.3.10 Get-path

The Get-path service is used by the DLS-user to obtain path information from the DLM to determine which is the preferred R-port.

6.6.4 Overview of interactions

The DLMS and their primitives are summarized in Table 9.

Table 9 – Summary of DL-management primitives and parameters

Service	Primitive	Parameter
Reset	DLM-RESET request	<none>
	DLM-RESET confirm	Status
Set-value	DLM-SET_VALUE request	Variable name Desired value
	DLM-SET_VALUE confirm	Status
Get-value	DLM-GET_VALUE request	Variable name
	DLM-GET_VALUE confirm	Status Current value
SAP-allocation	DLM-SAP_ALLOC request	SAP DLS-user ID
	DLM-SAP_ALLOC confirm	Status
SAP-deallocation	DLM-SAP_DEALLOC request	SAP
	DLM-SAP_DEALLOC confirm	Status
Get-SAP-information	DLM-GET_SAP_INFO request	SAP
	DLM-GET_SAP_INFO confirm	Status DLS-user ID
Get-diagnostic-information	DLM-GET_DIAG request	Diag-type
	DLM-GET_DIAG confirm	Status, Diagnostic information
Event	DLM-EVENT indication	DLM event ID

Service	Primitive	Parameter
Get-path	DLM-GET_PATH request	DST_addr
	DLM-GET_PATH confirm	status R-port MAC-address

The sequences of the DLM primitives are shown in Figures 17 and 18.

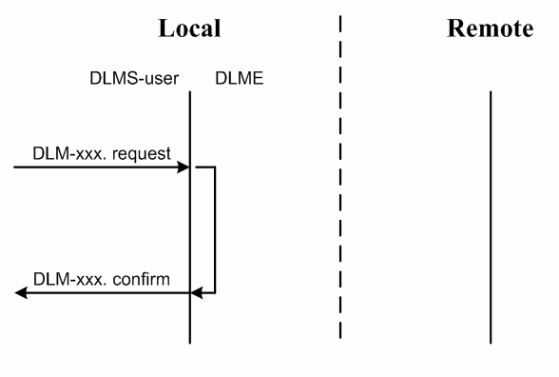


Figure 17 – Sequence diagram of Reset, Set-value, Get-value, SAP-allocation, SAP-deallocation, Get-SAP information and Get-diagnostic information service primitives

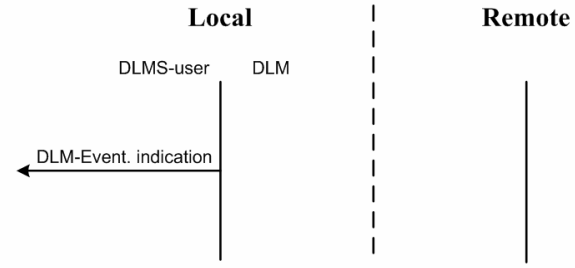


Figure 18 – Sequence diagram of event service primitive

6.6.5 Detailed specification of service and interactions

6.6.5.1 Reset

6.6.5.1.1 Function

The DLM-Reset request primitive allows the DLM to reset the DLE. Reset is equivalent to power on. When the DLE receives a reset request from the DLM, the DLE sets its status to “offline” and all DLE variables are cleared. The DLMS-user receives the DLM-Reset confirmation primitive with the success or failure status of the result.

6.6.5.1.2 Types of primitives and the parameters

6.6.5.1.2.1 General

Table 10 indicates the primitives and parameters of the Reset service.

Table 10 – DLM-RESET primitives and parameters

DLM-RESET	Request	Confirm
Parameter	Input	Output
Status	-	M
NOTE Establishing a method by which a confirmed primitive is correlated with its corresponding preceding request primitive is a local responsibility.		

6.6.5.1.2.2 Status

This parameter allows the DLMS-user to determine whether the requested service was provided successfully. If it failed, the reason is specified. The value of this parameter can be one of

a) “ok – successfully completed”;

b) “failure – terminated before completion”.

6.6.5.2 Set-value

6.6.5.2.1 Function

This service is used to assign new values to the variables of the DLE. The DLMS-user receives confirmation that the specified variables have been set to the new values.

6.6.5.2.2 Types of primitives and the parameters

6.6.5.2.2.1 General

Table 11 indicates the primitives and parameters of the Set-value service.

Table 11 – DLM-SET_VALUE primitives and parameters

DLM-SET_VALUE	Request	Confirm
Parameter	Input	Output
Variable name	M	-
Desired value	M	-
Status	-	M
NOTE Establishing a method by which a confirmed primitive is correlated with its corresponding preceding request primitive is a local responsibility.		

6.6.5.2.2.2 Variable name

This parameter specifies the variable in the DLE whose value is to be set.

NOTE The selectable variables and their permitted values or value ranges are defined in Clause 8.

6.6.5.2.2.3 Desired value

This parameter specifies the desired value for the selected variable.

6.6.5.2.2.4 Status

This parameter allows the DLMS-user to determine whether the requested service was provided successfully. If it failed, the reason is specified. The value of this parameter can be one of:

a) “ok – success – the variable could be updated”;

b) “failure – the variable does not exist or could not assume the new value”;

c) “failure – invalid parameters in the request”.

6.6.5.3 Get-value

6.6.5.3.1 Function

This service is used by the DLMS-user to read the value of a DLE variable. The response of the DLMS returns the actual value of the specified variable.

6.6.5.3.2 Type of primitives and parameters of DLM-GET_VALUE

6.6.5.3.2.1 General

Table 12 indicates the primitives and parameters of the Get-value service.

Table 12 – DLM-GET_VALUE primitives and parameters

DLM-GET_VALUE	Request	Confirm
Parameter	Input	Output
Variable name	M	
Current value	-	M
Status	-	M
NOTE Establishing a method by which a confirmed primitive is correlated with its corresponding preceding request primitive is a local responsibility.		

6.6.5.3.2.2 Variable name

This parameter specifies the variable in the DLE whose value is being requested. The selectable variables are defined in the corresponding part of Clause 8.

6.6.5.3.2.3 Current value

This parameter is present when the status parameter indicates that the requested service was performed successfully. This parameter specifies the current value of the selected variable.

NOTE The observable variables and their permitted value ranges are defined in Clause 8.

6.6.5.3.2.4 Status

This parameter allows the DLMS-user to determine whether or not the requested service was provided successfully. If it failed, the reason is specified. The value of this parameter can be one of:

a) “ok – success – the variable could be read”;

b) “failure – the variable does not exist or could not be read”;

c) “failure – invalid parameters in the request”.

6.6.5.4 SAP-allocation

6.6.5.4.1 Function

SAP-allocation service is used by the DLMS-user to obtain a SAP assignment from the DLM. The DLMS-user receives the DLM-SAP_ALLOC confirmation primitive indicating success or failure.

When a frame is received, the DLE examines the destination SAP address field in the frame and delivers the received frame to the corresponding DLS-user.

NOTE The allocation of SSAP is not restricted in this PAS. The default SSAP value of 0 is used for normal DL services. However, SSAP may be assigned a specific value for special application functions.

NOTE The method to allocate a SAP address is not restricted in this PAS so that a DLS-user can obtain a SAP assignment from the DLM without restriction. Therefore, network users need to check if the indicated SAP address is not duplicated with SAP addresses already allocated.

6.6.5.4.2 Types of primitives and the parameters

6.6.5.4.2.1 General

Table 13 indicates the primitives and parameters of the SAP-allocation service.

Table 13 – DLM-SAP_ALLOC primitives and parameters

DLM-SAP_ALLOC	Request	Confirm
Parameter	Input	Output
SAP	M	-
DLS-user ID	M	-
Status	-	M
NOTE Establishing a method by which a confirmed primitive is correlated with its corresponding preceding request primitive is a local responsibility.		

6.6.5.4.2.2 SAP

This parameter indicates the SAP for which the DLMS-user is attempting to obtain an assignment from the DLM.

NOTE The method to allocate an SSAP is not restricted in this PAS.

6.6.5.4.2.3 DLMS-user ID

This parameter indicates the numeric identification of the DLMS-user. This identification is unique to a local RAPIEnet device.

6.6.5.4.2.4 Status

This parameter allows the DLMS-user to determine whether or not the requested service was provided successfully. If it failed, the reason is specified. The value of this parameter can be one of:

a) “ok – success – SAP successfully allocated to DLS-user”;

b) “failure – indicated that the designated SAP is already used”;

c) “failure – invalid parameters in the request”.

6.6.5.5 SAP-deallocation

6.6.5.5.1 Function

The SAP-deallocation service is used by the DLMS-user to release and return an allocated SAP to the DLM. The DLMS-user receives the DLM-SAP_DEALLOC confirmation primitive indicating success or failure. The deallocated SAP can be reassigned to the DLS-user using the SAP-allocation service.

6.6.5.5.2 Types of primitives and the parameter

6.6.5.5.2.1 General

Table 14 indicates the primitives and parameters of the SAP-deallocation service.

Table 14 – DLM-SAP_DEALLOC primitives and parameters

DLM-SAP_DEALLOC	Request	Confirm
Parameter	Input	Output
SAP	M	-
Status	-	M
NOTE Establishing a method by which a confirmed primitive is correlated with its corresponding preceding request primitive is a local responsibility.		

6.6.5.5.2.2 SAP

This parameter indicates the SAP that the DLMS-user is attempting to release and return the allocation to the DLM.

NOTE The deallocation of an SSAP is not restricted in this PAS.

6.6.5.5.2.3 Status

This parameter allows the DLMS-user to determine whether or not the requested service was provided successfully. If it failed, the reason is specified. The value of this parameter can be one of:

a) “ok – success – the variable could be read”;

b) “failure – indicated SAP is not allocated to any DLS-user”;

c) “failure – invalid parameters in the request”.

6.6.5.6 Get-SAP-information

6.6.5.6.1 Function

GET-SAP-information is used by the DLMS-user to obtain SAP information from the DLM. The DLMS-user receives the DLM-GET_SAP_INFO confirmation primitive with the allocated DLS-user ID and SAP status.

6.6.5.6.2 Types of primitives and the parameters

6.6.5.6.2.1 General

Table 15 indicates the primitives and parameters of the Get-SAP-information service.

Table 15 – DLM-GET_SAP_INFO primitives and parameters

DLM-GET_SAP_INFO	Request	Confirm
Parameter	Input	Output
SAP	M	-
DLS-user ID	-	M
Status	-	M
NOTE Establishing a method by which a confirmed primitive is correlated with its corresponding preceding request primitive is a local responsibility.		

6.6.5.6.2.2 SAP

This parameter indicates the SAP about which the DLMS-user is attempting to obtain information from the DLM.

NOTE The method of obtaining information about an SSAP is not specified in this PAS.

6.6.5.6.2.3 DLS-user ID

This parameter indicates the numeric identification of the DLS-user to which the SAP is allocated.

6.6.5.6.2.4 Status

This parameter allows the DLMS-user to determine whether or not the requested service was provided successfully. If it failed, the reason is specified. The value of this parameter can be one of:

a) “ok – success – the variable could be read”;

b) “failure – indicated SAP is not allocated”;

c) “failure – invalid parameters in the request”.

6.6.5.7 Get-diagnostic-information

6.6.5.7.1 Function

The Get-diagnostic-information service is used by the DLMS-user to obtain diagnostic information, including local device information, remote device information, network status information, and the path table.

6.6.5.7.2 Types of primitives and the parameters

6.6.5.7.2.1 General

Table 16 indicates the primitives and parameters of the Get-diagnostic-information service.

Table 16 – DLM-GET_DIAG primitives and parameters

DLM-GET_DIAG	Request	Confirm
Parameter name	Input	Output
Diag-type	M	-
Local device Information	-	C
Network information	-	C
Path table information	-	C

DLM-GET_DIAG	Request	Confirm
DL-address	C	-
Status	-	M
NOTE Establishing a method by which a confirmed primitive is correlated with its corresponding preceding request primitive is a local responsibility.		

6.6.5.7.2.2 Diag-type

This parameter indicates the type of diagnostic information required by the DLMS-user. This parameter has one of three values: local device information, network information, or path table information (see 8.6.3.2).

6.6.5.7.2.3 Local device information

If the Diag-type is specified as local device information, local device information is returned to the DLMS-user using DLM-GET_DIAG confirmation (see 8.3.6.4).

6.6.5.7.2.4 Network information

If the Diag-type is specified as network information, network-related information is returned to the DLMS-user using DLM-GET_DIAG confirmation (see 8.3.6.5).

6.6.5.7.2.5 Path table information

If the Diag-type is specified as path table information, path table information is returned to DLMS-user using DLM-GET_DIAG confirmation (see 8.3.6.6).

6.6.5.7.2.6 DL-address

This parameter is used when the Diag-type is specified as local device information. This parameter indicates the device's DL-address.

6.6.5.7.2.7 Status

This parameter allows the DLMS-user to determine whether or not the requested service was provided successfully. If it failed, the reason is specified. The value of this parameter can be one of:

a) "ok – success – the diagnostic information could be read";

b) "failure – invalid parameters in the request".

6.6.5.8 Event

6.6.5.8.1 Function

This service is used to notify the DLMS-user that certain events or errors have occurred in the DLL.

6.6.5.8.2 Types of primitives and the parameters

6.6.5.8.2.1 General

Table 17 indicates the primitives and parameters of the event service.

Table 17 – DLM-EVENT primitives and parameters

DLM-EVENT	Indication
Parameter	Output
DLM event ID	M

6.6.5.8.2.2 DLM event ID

This parameter specifies the primitive or composite event in the DLE the occurrence of which is being announced. The possible values are defined in the corresponding subclauses of Clause 8.

Table 18 indicates the DLM event identifier.

Table 18 – DLM event identifier

DLM-event-identifier	Description
EVENT_NET_TPG_CHG	This event identifier indicates that the network topology has changed
EVENT_DEV_STATE_CHG	This event identifier indicates that the local device's DLM state has changed
EVENT_THIS_ADDR_COLLISION	This event identifier indicates that the local device's DL-address is duplicated to other device on the network
EVENT_THIS_ADDR_COLLISION_CLEAR	This event identifier indicates that the local device's DL-address collision has been cleared
EVENT_NET_ADDR_COLLISION	This event identifier indicates that there exist at least two devices on the network that have the same DL-address
EVENT_NET_ADDR_COLLISION_CLEAR	This event identifier indicates that the remote device's DL-address collision has been cleared
EVENT_IN_DEVICE	This event identifier indicates that a new device has joined the network
EVENT_OUT_DEVICE	This event identifier indicates that a device has left the network

6.6.5.9 Get-path

6.6.5.9.1 Function

The Get-path service is used by the DLMS-user to obtain the path table entry and preferred R-port information about the designated device from the DLM before sending a DLPDU to the destination device.

6.6.5.9.2 Types of primitives and the parameters

6.6.5.9.2.1 General

Table 19 indicates the primitives and parameters of the Get-path information service.

Table 19 – DLM-GET_PATH primitives and parameters

DLM-GET_PATH	Request	Confirm
Parameter	Input	Output
DST_addr	M	-
R-port	-	M
MAC address	-	M
Status	-	M
NOTE Establishing a method by which a confirmed primitive is correlated with its corresponding preceding request primitive is a local responsibility.		

6.6.5.9.2.2 DST_addr

This parameter indicates the destination DL-address of the DLE to which the DLPDU is to be delivered.

6.6.5.9.2.3 R-port

This parameter indicates the preferred R-port in the transmitting device that is to be used to send the DLPDU.

6.6.5.9.2.4 MAC address

This parameter indicates the ISO/IEC 8802-3 Ethernet MAC address of the destination device.

6.6.5.9.2.5 Status

This parameter allows the DLMS-user to determine whether the requested service was provided successfully. If it failed, the reason is specified. The value of this parameter can be one of:

a) “ok – success – the path information could be read”;

b) “failure – invalid parameter in the request”;

c) “failure – DL-address collision for the designated node”;

d) “failure – designated node is not found in the path table”.

6.7 MAC control service

6.7.1 General

This subclause describes the interface between a DLE and a MAC control service (MACS) user (MACS-user). The services of this interface are needed for the DLM state machine specified in 8.6.3.3.

6.7.2 MAC control service

6.7.2.1 Overview

MAC control functions are provided by the following services:

a) MAC-reset;

b) MAC-forward-control.

The MAC-reset and MAC-forward-control services are considered mandatory.

6.7.2.2 MAC-reset

The MACS-user uses this service to reset the MAC. Reset is equivalent to power on. The MACS-user receives a confirmation that this has taken place.

6.7.2.3 MAC-forward-control

Every RAPIEnet device has dual MAC Ethernet ports. The MAC-forward-control service is used by a MACS-user to control the frame relay function between two R-ports. The MACS-user receives a confirmation that this has taken place.

6.7.3 Overview of interactions

The MAC control services and their primitives are summarized in Table 20.

Table 20 – Summary of MAC control primitives and parameters

Service	Primitive	Parameter
MAC-Reset	MAC-RESET request	<none>
	MAC-RESET confirm	Status
MAC-forward-control	MAC-FW_CTRL request	R-port Forward enable
	MAC-FW_CTRL confirm	Status

The sequences of the MAC control primitives are shown in Figure 19.

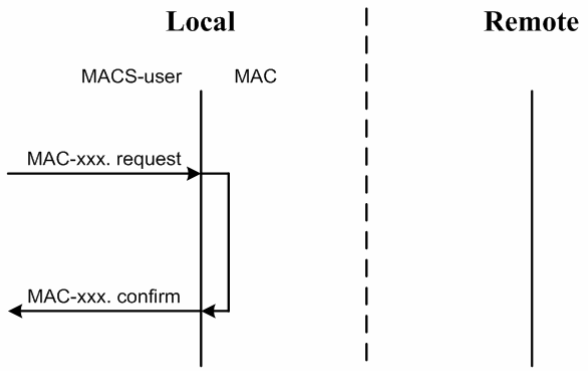


Figure 19 – Sequence diagram of MAC-reset and MAC-forward-control service primitive

6.7.4 Detailed specification of service and interactions

6.7.4.1 MAC-reset

6.7.4.1.1 Function

The MAC-Reset request primitive is used to reset the MAC. The MACS-user receives the MAC-Reset confirmation primitive with the success or failure of the result.

6.7.4.1.2 Types of primitives and the parameters

6.7.4.1.2.1 General

Table 21 indicates the primitives and parameters of the MAC-Reset service.

Table 21 – MAC-RESET primitives and parameters

MAC-RESET	Request	Confirm
Parameter	Input	Output
Status	-	M
NOTE Establishing a method by which a confirmed primitive is correlated with its corresponding preceding request primitive is a local responsibility.		

6.7.4.1.2.2 Status

This parameter allows the MACS-user to determine whether or not the requested service was provided successfully. If it failed, the reason is specified. The value of this parameter can be one of:

a) “ok – successfully completed”;

b) “failure – terminated before completion”.

6.7.4.2 MAC-forward-control

6.7.4.2.1 Function

The MAC-forward-control service allows a MACS-user to control the frame relay function between two RAPIEnet R-ports.

6.7.4.2.2 Types of primitives and parameters

Table 22 indicates the primitives and parameters of the MAC-forward-control service.

Table 22 – MAC-FW_CTRL primitives and parameters

MAC-FW_CTRL	Request	Confirm
Parameter	Input	Output
R-port	M	-
Forward enable	M	-
Status	-	M
NOTE Establishing a method by which a confirmed primitive is correlated with its corresponding preceding request primitive is a local responsibility.		

6.7.4.2.2.1 R-port

This parameter indicates the R-port whose frame relay function is to be controlled. When this parameter is indicated as R-port1, the frame relay function from R-port1 to R-port2 is to be controlled. On the other hand, when this parameter is indicated as R-port2, the frame relay function from R-port2 to R-port1 is to be controlled.

6.7.4.2.2.2 Forward enable

This parameter allows the MACS-user to enable or disable the frame relay function of the designated R-port.

6.7.4.2.2.3 Status

This parameter allows the MACS-user to determine whether or not the requested service was provided successfully. If it failed, the reason is specified. The value of this parameter can be one of:

a) “ok – successfully controlled”;

b) “failure – invalid parameters in the request”.

6.8 Ph-control service

6.8.1 General

This subclause describes the interface between the PhE and Ph-control service user (PhS-user). The services of this interface are required for the DLM state machine specified in 8.6.3.3.

6.8.2 Ph-control service

6.8.2.1 Overview

Ph-control provides the following services to the PhS-user:

a) Ph-reset;

b) Ph-get-link-status;

c) Ph-link-status-change.

The services Ph-reset, Ph-get-link-status, and Ph-link-status-change are considered mandatory.

6.8.2.2 Ph-reset

The PhS-user uses this service to reset the PhEs. Reset is equivalent to power on. The PhS-user receives a confirmation that this has taken place.

6.8.2.3 Ph-get-link-status

The Ph-get-link-status is used by the PhS-user to obtain the link status information: i.e., “link active” or “link inactive.”

6.8.2.4 Ph-link-status-change

The Ph-link-status-change service is used to notify the PhS-user of link status change information. This service is initiated by the hardware-triggered signal event in the physical layer.

6.8.3 Overview of interactions

The Ph-control services and their primitives are summarized in Table 23.

Table 23 – Summary of Ph-control primitives and parameters

Service	Primitive	Parameter
Ph-reset	Ph-RESET request	<none>
	Ph-RESET confirm	Status
Ph-get-link-status	Ph-GET_LINK_STATUS request	R-port
	Ph-GET_LINK_STATUS confirm	R-port link status Status
Ph-link-status-change	Ph-LINK_STATUS_CHANGE indication	R-port

The sequences of the Ph-control primitives are shown in Figures 20 and 21.

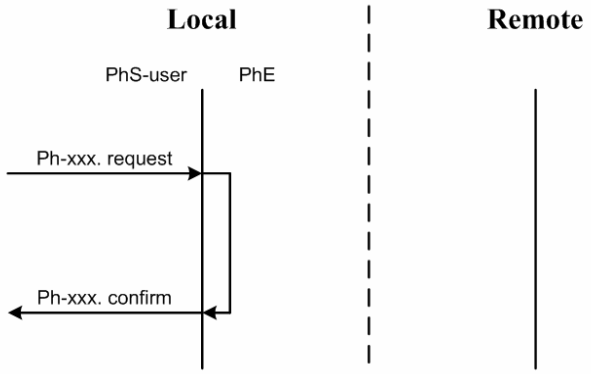


Figure 20 – Sequence diagram of Ph-reset and Ph-get-link-status service primitive

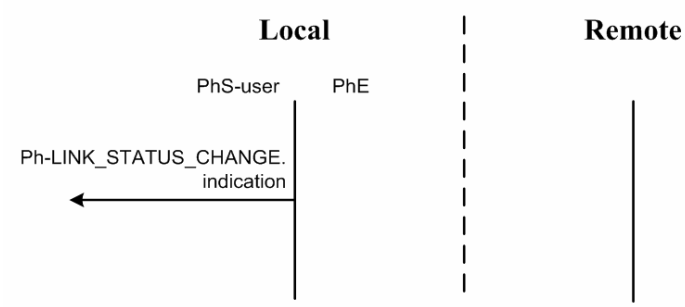


Figure 21 – Sequence diagram of Ph-link-status-change service primitive

6.8.4 Detailed specification of service and interactions

6.8.4.1 Ph-reset

6.8.4.1.1 Function

The Ph-reset request primitive is used to reset the PhE. The PhS-user receives the Ph-reset confirmation primitive with an indication of success or failure.

6.8.4.1.2 Types of primitives and the parameters

6.8.4.1.2.1 General

Table 24 indicates the primitives and parameters of the Ph-reset service.

Table 24 – Ph-RESET primitives and parameters

Ph-RESET	Request	Confirm
Parameter	Input	Output
Status	-	M
NOTE Establishing a method by which a confirmed primitive is correlated with its corresponding preceding request primitive is a local responsibility.		

6.8.4.1.2.2 Status

This parameter allows the PhS-user to determine whether or not the requested service was provided successfully. If it failed, the reason is specified. The value of this parameter can be one of:

a) “ok – successfully completed”;

b) “failure – terminated before completion”.

6.8.4.2 Ph-get-link-status

6.8.4.2.1 Function

The Ph-get-link-status service allows the PhS-user to obtain link status information from the physical layer.

6.8.4.2.2 Types of primitives and the parameters

6.8.4.2.2.1 General

Table 25 indicates the primitives and parameters of the Ph-get-link-status service.

Table 25 – Ph-GET_LINK_STATUS primitives and parameters

Ph-GET_LINK_STATUS	Request	Confirm
Parameter	Input	Output
R-port	M	-
link status	-	M
Status	-	M
NOTE The method by which a confirmation primitive is correlated with its corresponding preceding request primitive is a local matter.		

6.8.4.2.2.2 R-port

This parameter indicates the designated R-port to read the link status information.

6.8.4.2.2.3 Link status

This parameter indicates the returned link status information. The returned link status value is one of:

a) “Link Active – the link is activate and the transmission path between two adjacent nodes is available”;

b) “Link Inactive – the link is inactivate and the transmission path between two adjacent nodes is not available”.

6.8.4.2.2.4 Status

This parameter allows the PhS-user to determine whether or not the requested service was provided successfully. If it failed, the reason is specified. The value of this parameter can be one of:

a) “ok – success – the R-port link status could be read”;

b) “failure – invalid parameters in the request”.

6.8.4.3 Ph-link-status-change

6.8.4.3.1 Function

The Ph-link-status-change service is used to notify the PhS-user of link status change information. This service is initiated by the hardware-triggered signal event.

6.8.4.3.2 Types of primitives and the parameters

6.8.4.3.2.1 General

Table 26 indicates the primitives and parameters of the Ph-link-status-change service.

Table 26 – Ph-LINK_STATUS _CHANGE primitives and parameters

Ph-LINK_STATUS_CHANGE	Indication
Parameter	Output
R-port	M

6.8.4.3.2.2 R-port

This parameter indicates the R-port whose link status is changed.

7 Data link layer protocol specification

7.1 Introduction

This clause describes the RAPIEnet data link protocol. This section of the data link protocol PAS is one of the Real-time Ethernet (RTE) specifications to facilitate the interconnection of automation system components. RAPIEnet data link services provide reliable and transparent data communication between RAPIEnet data link users through the logical interface between the ISO/IEC 8802-3 physical layer and the data link layer (DLL).

This clause of the data link protocol specifies the detailed descriptions of data link services specified in Clause 7. This clause defines the data link protocol functions and the characteristics for the interactions between the peer devices.

7.2 Scope

7.2.1 General

The DLL provides basic time-critical data communications between devices in an automated environment. RAPIEnet provides priority-based cyclic and acyclic data communication using an internal collision-free, full-duplex dual-port Ethernet switch technology. For wide application in various automation applications, RAPIEnet does not restrict the cyclic/acyclic scheduling policy in the DLL.

7.2.2 Specifications

This PAS describes

a) procedures for the timely transfer of data and control information from one data link user entity to a peer user entity and among the data link entities forming the distributed data link service provider;

b) procedures for giving communication opportunities based on PAS ISO/IEC 8802-3 MAC, with provisions for nodes to be added or removed during normal operation;

c) structure of the fieldbus data link protocol data units (DLPDUs) used for the transfer of data and control information by the protocol of this PAS, and their representation as physical interface data units.

7.2.3 Procedures

The procedures are defined in terms of

a) interactions between peer data link entities (DLEs) through the exchange of fieldbus DLPDUs;

b) interactions between a data link service (DLS) provider and a DLS-user in the same system through the exchange of DLS primitives;

c) interactions between a DLS-provider and a physical layer service provider in the same system through the exchange of Ph-service primitives.

7.2.4 Applicability

These procedures are applicable to instances of communication between systems that support time-critical communications services in the data link layer of the OSI or Fieldbus reference models and that require the ability to interconnect in an open systems interconnection environment. Profiles provide a simple multi-attribute means of summarizing an implementation’s capabilities and, thus, its applicability to various time-deterministic communications needs.

7.2.5 Conformance

This PAS also specifies conformance requirements for systems implementing these procedures. This PAS does not contain tests to demonstrate compliance with such requirements.

7.3 Overview of the data link protocol

7.3.1 General

RAPIEnet extends Ethernet according to the ISO/IEC 8802-3 specification with mechanisms to transfer data with predictable timing demands typical of high-performance automation. It does not change the basic principles of the Ethernet PAS ISO/IEC 8802-3 but extends it toward RTE. Thus, it is possible to continue to use standard Ethernet hardware, infrastructure components, or test and measurement equipment, such as network analysers.

7.3.2 Overview of medium access control

A RAPIEnet device requires an integrated switch with two ports (ring ports) connected to the ring. A RAPIEnet network system is constructed with full-duplex, collision-free Ethernet switching devices as a ring or a line network. RAPIEnet guarantees collision-free data transmission between two devices linked by a full-duplex Ethernet connection. Thus, the RAPIEnet data link layer provides reliable, transparent and collision-free data transmission among DLS-users.

A RAPIEnet connection provides collision-free, full-duplex data communication between two devices. Therefore, a RAPIEnet device can transmit a frame at any time without restriction of media access rights. RAPIEnet data link layer does not restrict the scheduling method to use for data link resources, but each device can be individually scheduled by the application program so that RAPIEnet can be applied flexibly in various applications.

A RAPIEnet frame is delivered to the destination device in one of the following two ways.

a) A frame is initiated by a source device and directly transmitted to the neighbouring destination device.
b) A frame is initiated by a source device and forwarded to the destination device by intermediate devices.

When a frame is forwarded by the intermediate device, the frame forwarding procedure is processed by the internal hardware switch to minimize the processing time, and it does not affect the performance of RAPIEnet DLL.

7.3.3 Service assumed from the physical layer

This subclause describes the assumed physical layer service (PhS) and the constraints used by the DLE. The physical service is assumed to provide the following service primitives specified by ISO/IEC 8802-3, Clause 2.

The assumed primitives of PhS are MA-DATA.request and MA-DATA.indication.

The temporal relationship of the primitives is shown in Figure 22.

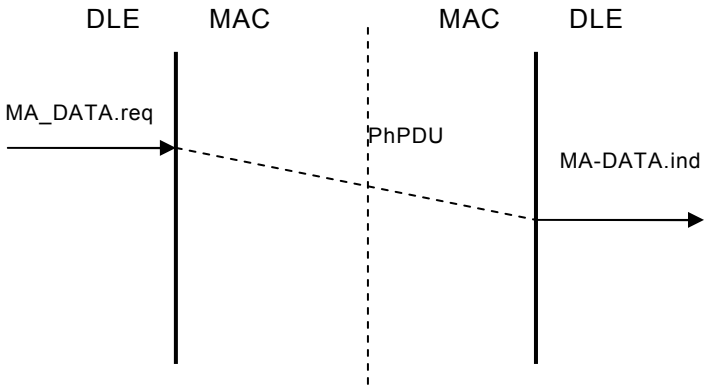


Figure 22 – Interaction of PhS primitives with DLE

The MA-DATA request primitive defines the transfer of data from a MAC client entity to a single peer entity or multiple peer entities in the case of group addresses.

The MA-DATA indication primitive defines the transfer of data from the MAC sublayer entity (through the optional MAC control sublayer, if implemented) to the MAC client entity or entities in the case of group addresses.

7.3.4 DLL architecture

The RAPIEnet DLL provides reliable and efficient higher-level data transfer service using a full-duplex ring or line topology without any specific access control scheme. The RAPIEnet

DLL is modeled as a combination of control components of the data link protocol machine (DLPM) and the DLL management interface.

The DLPM transmits the data link service data unit (DLSDU) generated by the local DLS-user to the appropriate R-port according to the data link service policy. The DLPM also examines the received frame and delivers the received DLPDU to the appropriate DLS-user.

The DLL management interface provides DLL management functions to maintain the network management information base (NMIB). The NMIB includes local device information, network information and the path table. The DLL is comprised of the components listed in Table 27.

Table 27 – DLL components

Components	Description
DL-protocol machine (DLPM)	Transmits the DLSDU received from local DLS-user through the real-time queue (RT-queue) or non-real-time queue (NRT-queue). Examines the received frame and delivers the DLPDU to the appropriate DLS-user
DLL management interface (DLM-interface)	Holds the station management variables that belong to the DLL, and manages synchronized changes of the link parameters

The internal arrangement of these components, and their interfaces, are shown in Figure 23. The arrowheads illustrate the primary direction of the flow of data and control.

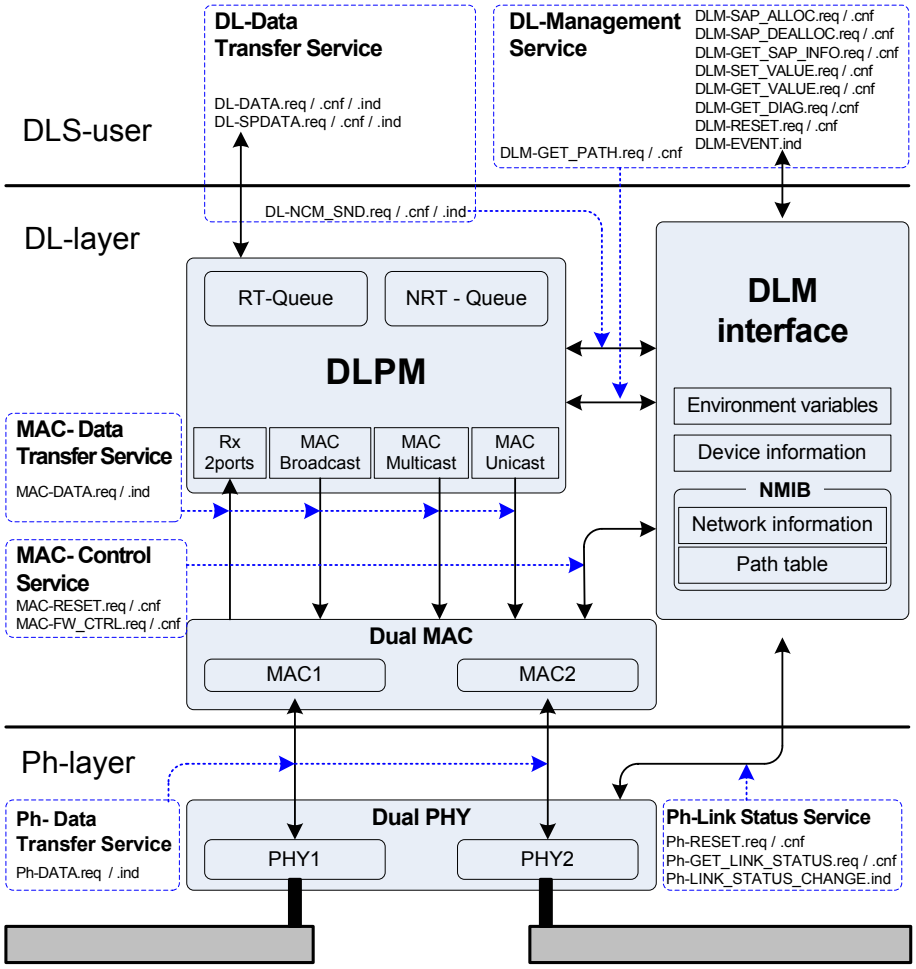


Figure 23 – Data link layer architecture

7.3.4.1 DLL management (DLM) interface support function

DLM is one of the key RAPIEnet features that enables the Plug and Play (PnP), Network Auto Configuration (NAC), and Extremely Fast Recovery (EFR) functions. DLM spontaneously maintains and shares local device information and network-related information with every device on the network.

a) RAPIEnet NMIB management

   The DLM interface maintains the NMIB including the local device information that indicates the physical and logical status of the device, the network information that indicates the current network configuration status, and the path table that indicates the path information to the other devices and their profiles.

b) Plug and Play (PnP)

   When a device joins the existing network, it is automatically detected and configured without extra manual configuration of parameters so that the new device can communicate directly with the other devices on the network.

c) Network auto configuration (NAC)

   RAPIEnet supports ring and line network topologies. When the network is changed from ring-to-line or from line-to-ring topology, the change is automatically detected and broadcast to every device on the network, and then the network information and path table entries are automatically updated.

d) Extremely fast recovery (EFR)

   When the network is changed from ring-to-line topology, network information and path information are automatically updated within 10 ms.

e) System diagnostic services

   The DLM interface also provides system diagnostic service.

7.3.5 Data type

This subclause describes the basic RAPIEnet data types. The data types described in this section are only the normative references to represent RAPIEnet data type formats. The implementation issues are not covered in this PAS. The encoding rule of each data type is described in 8.6.

7.3.5.1 Boolean

Data of basic data type BOOLEAN can have the values TRUE or FALSE. The values are represented as bit sequences of length 1. The value TRUE is represented by a 1 and the value FALSE is represented by a 0.

7.3.5.2 Unsigned integer

Data of basic data type UNSIGNEDn have values of non-negative integers. The value range is 0,…, 2^{n-1}. The data are represented as bit sequences of length n. The bit sequence

$$b = b_{n-1} \sim b_0$$

is assigned the value

$$\text{UNSIGNEDn} = b_{n-1}2^{n-1} + .. + b_1 2^1 + b_0 2^0$$

The bit sequence starts on the right with the least significant byte as shown in Table 28.

Table 28 – UNSIGNEDn data type

Octet number	1	2	3	4	5	6	7	8
UNSIGNED8	$b_7..b_0$							
UNSIGNED16	$b_7..b_0$	$b_{15}..b_8$						
UNSIGNED24	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$					
UNSIGNED32	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$				
UNSIGNED40	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$			
UNSIGNED48	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$		
UNSIGNED56	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	
UNSIGNED64	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	$b_{63}..b_{56}$

7.3.5.3 Signed integer

Data of basic data type INTEGERn have values of integers. The value range is from -2^{n-1} to 2^{n-1}. The data are represented as bit sequences of length n. The bit sequence

$$b = b_{n-1} \sim b_0$$

is assigned the value

INTEGERn = $b_{n-2}2^{n-2} + .. + b_1 2^1 + b_0 2^0$, when b_{n-1} is 0, and

INTEGERn = -Complement(b) – 1, when b_{n-1} is 0,

The bit sequence starts on the right with the least significant byte as shown in Table 29.

Table 29 – INTEGERn data type

Octet number	1	2	3	4	5	6	7	8
INTEGER8	$b_7..b_0$							
INTEGER16	$b_7..b_0$	$b_{15}..b_8$						
INTEGER24	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$					
INTEGER32	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$				
INTEGER40	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$			
INTEGER48	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$		
INTEGER56	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	
INTEGER64	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	$b_{63}..b_{56}$

7.3.5.4 Octet String

The data type OCTET_STRINGn is defined below. The n represents the byte length of the octet string.

ARRAY[n] of UNSINGED8 OCTET_STRINGn

7.3.5.5 Visible String

The data type VISIBLE_STRINGn is defined below. The admissible values of data of type VISIBLE_CHAR are 0x00 and the range 0x20–0x7E. The data are interpreted as 7-bit coded characters. The n indicates the byte length of the visible string.

UNSINGED8 VISIBLE_CHAR

ARRAY[n] of VISIBLE_CHAR VISIBLE_STRINGn

There is no 0x00 necessary to terminate the string.

7.3.5.6 Time of day

The data type TIMEOFDAY is defined below. TIMEOFDAY consists of UNSIGNED16 date counting and UNSIGNED32 millisecond fields.

TIMEOFDAY

UNSIGNED32 milliseconds: the count from time 00:00 in milliseconds.

UNSIGNED16 date count: the number of days from January 1, 1984.

7.3.6 Local parameters and variables

This PAS uses DLS-user request parameters P(…) and local variables V(…) as a means of clarifying the effects of certain actions and the conditions under which those actions are valid. The specification also uses local timers T(…) as a means of monitoring the actions of the distributed DLS-provider and of ensuring a local DLE response to the absence of those actions. The specification uses local counters C(…) for performing rate measurement functions. It also uses local queues Q(…) as a means of ordering certain activities, of clarifying the effects of certain actions, and of clarifying the conditions under which those activities are valid.

Unless otherwise specified at the moment of their creation or of DLE activation:

a) all variables shall be initialized to their default value, or to their minimum permitted value if no default is specified;

b) all counters shall be initialized to zero;

c) all timers shall be initialized to inactive.

DLM may change the values of configuration variables.

Local parameters and variables include DLE configuration parameters, local device information, and NMIB variables. The DLE operational condition is stored in DLE configuration parameters and the DLE configuration parameters are configured locally or remotely according to the application. Local data link information, such as DL-address and the status of each R-port are stored in the local device information. NMIB includes network information and the path table. Network topology and the network-related variables are stored in the network

information, and the device profile and path information of the other devices on the network are summarized in the path table.

7.3.6.1 DLE configuration parameters

These parameters are required to configure the local DLE operation, and they are managed by DLM. Every device on the same network segment has the same DLE configuration parameters. DLM-GET_VALUE service and DLM-SET_VALUE service are used to read or write the DLE configuration parameters. Table 30 shows the list of DLE configuration parameters.

Table 30 – DLE configuration parameters

Parameter	Data type	Default value	Description
Max DL-address	UNSIGNED16	255	Maximum DL-address 256–65535: Reserved
DLPM scheduling policy	UNSIGNED8	0	0: First-In, First-Out 1: Fixed priority 2–255: Reserved

7.3.6.1.1 P(MAX_ADDR): Maximum DL-address

This variable holds the maximum device address and is set by the DLM. The range of this variable is 1–255. The default value of this variable is 255. This variable also indicates the maximum number of path table entries in the NMIB. The values in the range 256–65535 are reserved.

7.3.6.1.2 P(DLPMSP): data link protocol machine scheduling policy

This variable holds the scheduling policy of the local DLPM, which dictates how to serve the concurrent DLSDUs in the RT-queue. The NRT-queue is not scheduled by the DLPMSP. DLPMSP can have one of the following values.

a) 0: first-in, first-out. The first received frame is served first. In this case, the message priority in the DLSDU is ignored.

b) 1: Fixed priority. The DLSDU with high priority is served first. If many DLSDUs are stacked with the same priority, the frame received first is served first.

c) 2–255: reserved.

7.3.6.2 Queues to support data transfer

When a DLS-user generates a DLSDU to be transmitted by the DLPM, the DLSDU is encapsulated by the DLPDU. The DLPDU is then stored in the RT-queue or the NRT-queue according to its application and service type. Table 31 shows the queue list for RAPIEnet data transfer.

Table 31 – Queues to support data transfer

Parameter	Data type	Default value	Description
RT-queue	-	-	Transmit queue to store the real-time data
NRT-queue	-	-	Transmit queue to store the non-real-time data

7.3.6.2.1 Q(RTQ): RT-queue

The RT-queue stores the real-time DLPDUs generated by a DLS-user to be transmitted by the DLPM. The size of the RT-queue is not restricted in this PAS and it is considered a detail of the local implementation. The RT-queue is scheduled by the DLPM.

7.3.6.2.2 Q(NRTQ): NRT-queue

The NRT-queue stores non-real-time DLPDUs generated by a DLS-user. The size of the NRT-queue is not restricted in this PAS and it is considered a detail of the local implementation. Messages in the RT-queue are transmitted first. Messages in the NRT-queue are transmitted only when the RT-queue is empty.

7.3.6.3 Variables to support SAP management

Allocation and de-allocation of the data link service access point (DLSAP) is managed by the DLM. When a frame is received, the DLPM examines the destination service access point (DSAP) in the DLSDU. If the DSAP is already allocated to a DLS-user, the DLM returns the appropriate DLS-user ID equivalent for the received DSAP address, and the DLPM delivers the received DLSDU to the DLS-user. If the service access point (SAP) is not allocated and no appropriate DLS-user ID is found in the DLM, the received DLSDU is discarded by the DLPM. Therefore, to receive a DLSDU from a certain peer DLS-user, a DLS-user must first obtain a SAP allocation using the DLM-SAP_ALLOC service. Once the SAP is allocated to a DLS-user, it is used to send and receive data until the SAP is deallocated and returned to the DLM. The deallocated SAP can be used again after reallocation. The SAP address and its appropriate DLS-user ID are stored together and maintained by the DLM. The maximum number of SAP management items is 65535 but the method to allocate and de-allocate the SAP address is not restricted in this PAS. Table 32 shows the list of SAP management variables.

Table 32 – Variables to support SAP management

Parameter	Data type	Default value	Description
SAP	UNSIGNED16	-	Service access point
DLS-user ID	UNSIGNED32	-	Numeric identification of the DLS-user that owns the SAP allocation

7.3.6.3.1 V(SAP): SAP

This variable holds the SAP. The value for this variable is in the range 0–65535.

7.3.6.3.2 V(DLS_USER_ID): DLS-user ID

This variable holds the 4-byte numeric identification of the DLS-user who owns the SAP allocation.

7.3.6.4 Variables to support local device information management

To maintain the network topology, every device manages a device database including the local device information and the other device information. Table 33 shows the list of device information management variables.

Table 33 – Variables to support device information management

Parameter	Data type	Default value	Description
DL-address	UNSIGNED16	INVALID_ADDR	Local DL-address
Device flags	UNSIGNED64	0	Local device flags

Device state	UNSIGNED8	0	DLM state
Device UID	UNSIGNED64	INVALID_UID	Local device unique ID
Device UID for R-port1	UNSIGNED64	INVALID_UID	Device unique ID connected through R-port1
Device UID for R-port2	UNSIGNED64	INVALID_UID	Device unique ID connected through R-port2
MAC address	UNSIGNED48	0	Local device MAC address
Port information	UNSIGNED16	0	Local device port information
Protocol version	UNSIGNED8	0	Local device protocol version
Device type	UNSIGNED16	0	Local device type
Device description	VISIBLE_STRING[16]	" "	Device description string
Hop count	UNSIGNED16	0	Hop count

7.3.6.4.1 V(DL_ADDR): DL-address

This variable holds the DL-address that designates the (single) DL-entity associated with a single device on a specific local link whose value is constrained to the range 0–255. The DL-address may be provided by hardware settings (for example, rotary switch) or set by software. The DL-address is defined as Table 34.

Table 34 – DL-address

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED16:8	Read/Write	Read	0–255: DL-address
UNSIGNED16:8	Read/Write	Read	256–65535: reserved

7.3.6.4.2 V(DEV_FLAG): Device flag

This variable holds the flags for events that occurred in a local device. When the local DL-address collision flag is set, EVENT_THIS_ADDR_COLLISION event is generated by the DLM, and then the local DL-address collision flag is cleared. When the DLM state change flag is set, EVENT_DEV_STATE_CHG is generated by the DLM, and then the DLM state change flag is cleared. RAPIEnet device flags and event flags are listed in Table 35.

Table 35 – Device flags

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED64:1	Read/Write	Read	Local DL-address collision flag 0x00: Normal 0x01: Collision
UNSIGNED64:1	Read/Write	Read	DLM state change flag 0x00: Normal 0x01: changed
UNSIGNED64:62	Read/Write	Read	Reserved

7.3.6.4.3 V(DLM_STATE): DLM state

This variable holds the DLM state. DLM state is defined as shown in Table 36.

Table 36 – DLM state

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED8	Read/Write	Read	0x00: INVALID_DLM_STATE 0x01: standalone state (SA) 0x02: line network manager state (LNM) 0x03: general Device state (GD) 0x04: primary ring network manager state (RNMP) 0x05: secondary ring network manager state (RNMS) 6–255: Reserved

7.3.6.4.4 V(DEV_UID): Device UID

This variable holds the unique 8-byte identification that identifies a RAPIEnet device in a network. It is a combination of the 6-byte ISO/IEC 8802-3 MAC address and the 2-byte DL-address. The device UID is defined as shown in Table 37.

Table 37 – Device unique identification

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED64:16	Read/Write	Read	0–255: DL-address 256–65535: Reserved
UNSIGNED64:48	Read/Write	Read	ISO/IEC 8802-3 MAC address

7.3.6.4.5 V(DEV_UID_RP1): Device UID for R-port1

This variable holds the UID of the device that is linked through the R-port1. The device UID for R-port1 is defined as shown in Table 38.

Table 38 – Unique identification of device connected to R-port1

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED64:16	Read/Write	Read	0–255: DL-address 256–65535: Reserved
UNSIGNED64:48	Read/Write	Read	ISO/IEC 8802-3 MAC Address

7.3.6.4.6 V(DEV_UID_RP2): Device UID for R-port2

This variable holds the UID of the device that is linked through the R-port2. The Device UID for R-port2 is defined as shown in Table 39.

Table 39 – Unique identification of device connected to R-port2

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED64:16	Read/Write	Read	0–255: DL-address 256–65535: Reserved
UNSIGNED64:48	Read/Write	Read	ISO/IEC 8802-3 MAC Address

7.3.6.4.7 V(MAC_ADDR): MAC address

This variable holds 6-byte ISO/IEC 8802-3 Ethernet MAC address of local device. As a RAPIEnet device has two Ethernet MAC ports, both MAC addresses should be identical. The MAC address is defined as shown in Table 40.

Table 40 – MAC address

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED48	Read/Write	Read	ISO/IEC 8802-3 MAC address

7.3.6.4.8 V(PORT_INFO): Port information

This variable holds the port information for each R-port, including such elements as link status, linked device type, and frame forwarding control. It is defined as shown in Table 41.

Table 41 – Port information

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED16:1	Read/Write	Read	Port1 link status 0x00:R-port1 link inactive 0x01:R-port1 link active
UNSIGNED16:1	Read/Write	Read	Port2 link status 0x00: R-port2 link inactive 0x01:R-port2 link active
UNSIGNED16:1	Read/Write	Read	Port1 linked device type 0x00: R-port1 heterogeneous device 0x01: R-port1 homogeneous device
UNSIGNED16:1	Read/Write	Read	Port2 linked device type 0x00: R-port2 heterogeneous device 0x01: R-port2 homogeneous device
UNSIGNED16:1	Read/Write	Read	Frame forwarding function from R-port1 to R-port2 0x00: Forward disabled 0x01: Forward enabled
UNSIGNED16:1	Read/Write	Read	Frame Forwarding function from R-port2 to R-port1 0x00: Forward disabled 0x01: Forward enabled
UNSIGNED16:10	Read/Write	Read	Reserved

7.3.6.4.9 V(PROTOCOL_VER): Protocol version

This variable holds the protocol version of local device. It is defined as shown in Table 42.

Table 42 – Protocol version

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED8:2	Read	Read	0x00: major version 1 0x01: major version 2 0x02: major version 3 0x03: major version 4
UNSIGNED8:3	Read	Read	0x00: minor version 0 0x01: minor version 1 0x02: minor version 2 0x03: minor version 3 0x04: minor version 4 0x05: minor version 5 0x06: minor version 6 0x07: minor version 7
UNSIGNED8:3	-	-	Reserved

7.3.6.4.10 V(DEV_TYPE): Device type

This variable holds the local device type that represents the general function of the device. The value of this variable is defined as shown in Table 43.

Table 43 – Device type

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED16:8	Read	Read/Write	0–255: general device type 0: invalid device type 1: programmable logic controller (PLC) 2: motion controller 3: human-machine interface (HMI) 4: industrial personal computer 5: inverter 6: simple I/O 7–255: reserved
UNSIGNED16:8	Read	Read/Write	0–255: application specific device type

7.3.6.4.11 V(DEV_DESC): Device description

This variable contains a description of the local device. The maximum length of this variable is 16 bytes. It is defined as shown in Table 44.

Table 44 – Device description

Data type	Access type DLM	Access type DLS-user	Value/Description
VISIBLE_STRING[16]	-	Read/Write	Any string defined by a DLS-user set using the set DLL configuration service

7.3.6.4.12 V(HOP_CNT): Hop count

This variable holds the count of the number of devices between two devices. When the DLM receives NCM_ADV_THIS or NCM_LA DLPDU, the DLM saves the received hop count value in this variable, then increments the hop counts in the received frame by 1 and transmits the frame through the other R-port. In this way, each device builds its own path table with a hop count for R-port1 and a hop count for R-port2. This variable is defined as shown in Table 45.

Table 45 – Hop count

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED16	Read/Write	-	0–255: Hop count 256–65535: reserved

7.3.6.5 Variables and counter to support network information management

Network information is managed automatically by the DLM. Network information variables and counters are summarized in Table 46.

Table 46 – Variables to support managing network information

Parameter	Data type	Default value	Description
Topology	UNSIGNED8	NET_TPG_SA	Network topology
Collision count	UNSIGNED8	0	DL-address collision counter between

			remote devices
Device count	UNSIGNED16	1	Device counter for the network segment
Topology change count	UNSIGNED16	0	Network topology change counter
Last topology change time	TIMEOFDAY	0	Date and time when the network topology last was changed.
RNMP device UID	UNSIGNED64	INVALID_UID	UID of the RNMP device
RNMS device UID	UNSIGNED64	INVALID_UID	UID of the RNMS device
LNM device UID for R-port1	UNSIGNED64	INVALID_UID	UID of the LNM device in R-port1 direction
LNM device UID for R-port2	UNSIGNED64	INVALID_UID	UID of the LNM device in R-port2 direction
Network flags	UNSIGNED64	0	Network event flags

7.3.6.5.1 V(TPG): Topology

This variable holds the type of network topology. The value of this variable is defined as shown in Table 47.

Table 47 – Topology

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED8	Read/Write	Read	Network topology 0x01: standalone 0x02: line topology 0x03: ring topology

7.3.6.5.2 C(COLL_CNT): Collision count

This variable holds the DL-address collision count for remote devices. The value is incremented by the DLM when a remote DL-address collision is detected and the value is decremented when the collision is cleared. This variable is defined as Table 48

Table 48 – Collision count

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED8	Read/Write	Read	0–255: remote DL-address collision count

7.3.6.5.3 C(DEV_CNT): Device Count

This variable holds the total number of devices on the network to a maximum of 256. It is defined as shown in Table 49.

Table 49 – Device count

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED16	Read/Write	Read	0: not used 1–256: device count 257–65535: reserved

7.3.6.5.4 C(TPG_CHG_CNT): Topology change count

This variable holds the topology change count. The value is incremented by the DLM when the network is changed from ring-to-line or from line-to-ring topology. It is defined as shown in Table 50.

Table 50 – Topology change count

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED16	Read/Write	Read	0–65535: Topology change count

7.3.6.5.5 V(TPG_CHG_TIME): Last topology change time

This variable holds the date and time when the network topology was last changed. It is defined as shown in Table 51.

Table 51 – Last topology change time

Data type	Access type DLM	Access type DLS-user	Value/Description
TIMEOFDAY	Read/Write	Read	The date and time when the network topology was last changed

7.3.6.5.6 V(UID_RNMP): RNMP device UID

This variable holds the device UID selected as the RNMP on the network. It is defined as shown in Table 52.

Table 52 – RNMP device UID

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED64:16	Read/Write	Read	0–255: DL-address 256–65535: Reserved
UNSIGNED64:48	Read/Write	Read	ISO/IEC 8802-3 MAC address

7.3.6.5.7 V(UID_RNMS): RNMS device UID

This variable holds the UID of the device selected as the RNMS on the network. It is defined as shown in Table 53.

Table 53 – RNMS device UID

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED64:16	Read/Write	Read	0–255: DL-address 256–65535: Reserved
UNSIGNED64:48	Read/Write	Read	ISO/IEC 8802-3 MAC address

7.3.6.5.8 V(UID_LNM_RP1): LNM device UID for R-port1

In a RAPIEnet line network, the two end devices are automatically selected as the LNMs. This variable holds the UID of the device selected as the LNM in the R-port1 direction. It is defined as shown in Table 54.

Table 54 – LNM device UID for R-port1

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED64:16	Read/Write	Read	0–255: DL-address 256–65535: Reserved
UNSIGNED64:48	Read/Write	Read	ISO/IEC 8802-3 MAC address

7.3.6.5.9 V(UID_LNM_RP2): LNM device UID for R-port2

In a RAPIEnet line network, two end devices are automatically selected as the LNMs. This variable holds the UID of the device selected as the LNM in the R-port2 direction. It is defined as shown in Table 55.

Table 55 – LNM device UID for R-port2

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED64:16	Read/Write	Read	0–255: DL-address 256–65535: Reserved
UNSIGNED64:48	Read/Write	Read	ISO/IEC 8802-3 MAC address

7.3.6.5.10 V(NET_FLAG): Network flags

This variable holds the event flags to notify the DLMS-user of network events. When any bit in this variable is set, the DLM generates a DLM-event indication service primitive to notify the DLMS-user of the event. After the DLM-event service has completed successfully, the designated bit in this variable is cleared. Network flags are defined in Table 56.

Table 56 – Network flags

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED64:1	Read/Write	Read	Network topology change status 0x00: Normal 0x01: Network topology has changed
UNSIGNED64:1	Read/Write	Read	Network DL-address collision status 0x00: Normal 0x01: Network DL-address collision detected
UNSIGNED64:1	Read/Write	Read	In device status 0x00: Normal 0x01: New device has joined the network
UNSIGNED64:1	Read/Write	Read	Out device status 0x00: Normal 0x01: Device has left the network

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED64:60	-	-	Reserved

The bit fields in the network flags are defined as follows.

a) _DEVICE network topology change status

This bit is set to TRUE by the DLM when it detects that the network topology has changed. This bit is reset to FALSE when the DLMS-user has been notified using the EVENT_NET_TPG_CHG service.

b) Network DL-address collision status

This bit is set to TRUE by the DLM when it detects a DL-address collision on the network. This bit is reset to FALSE when the DL-address collision is cleared.

c) In device status

This bit is set to TRUE by the DLM when it detects that a new device has joined the network. This bit is reset to FALSE when the DLMS-user has been notified using the EVENT_IN_DEVICE service.

d) Out device status

This bit is set to TRUE by the DLM when it detects that a device has left the network. This bit is reset to FALSE when the DLMS-user has been notified using the EVENT_OUTservice.

7.3.6.6 Variables and counter to support a device path information management

The path table is managed by the DLM and is made up of table items related to the other RAPIEnet devices on the network. The variables and counters for a table item are defined in Table 57.

Table 57 – Variables and counter to support managing path information

Parameter	Data type	Default value	Description
DL-address	UNSIGNED16	INVALID_ADDR	DL-address
Hop count for R-port1	UNSIGNED16	INVALID_HOP_CNT	Hop count in R-port1 direction
Hop count for R-port2	UNSIGNED16	INVALID_HOP_CNT	Hop count in R-port2 direction
Preferred R-port	UNSIGNED8	INVALID_R_PORT	R-port with the smaller hop count value to the peer device.
Destination R-port	UNSIGNED8	INVALID_R_PORT	Selected R-port for sending a frame to the destination DL-address.
Device state	UNSIGNED8	0	Peer device's DLM state
MAC address	UNSIGNED48	0	Peer device's ISO/IEC 8802-3 MAC address
Port information	UNSIGNED16	0	Peer device's local R-port information
Protocol version	UNSIGNED8	0	Peer device's protocol version
Device type	UNSIGNED16	0	Peer device's application device type
Device description	VISIBLE_STRING[16]	" "	Peer device's description
Device UID	UNSIGNED64	INVALID_UID	Peer device's device UID
Device UID for R-port1	UNSIGNED64	INVALID_UID	Device UID of the device connected through the peer device's R-port1
Device UID for R-port2	UNSIGNED64	INVALID_UID	Device UID of the device connected through the peer device's R-port2

Parameter	Data type	Default value	Description
In net count	UNSIGNED16	0	The number of times that the peer device has joined in the network
In net time	TIMEOFDAY	0	The date and time when the peer device last joined the network
Out net count	UNSIGNED16	0	The number of times that the peer device has been disconnected from the network
Out net time	TIMEOFDAY	0	The date and time when the peer device was last disconnected from the network

7.3.6.6.1 V(path-DL_ADDR): DL-address

See 8.3.6.4.1

7.3.6.6.2 C(path-HOP_CNT_RP1): Hop count for R-port1

This variable indicates the frame forwarding counts for sending a frame from the local device to the peer device through the R-port1. It is defined as shown in Table 58.

Table 58 – Hop count for R-port1 direction

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED16	Read/Write	Read	0–255: Hop count for R-port1 256–65535: Reserved

7.3.6.6.3 C(path-HOP_CNT_RP2): Hop count for R-port2

This variable indicates the frame forwarding counts for sending a frame from the local device to the peer device through the R-port2. It is defined as shown in Table 59.

Table 59 – Hop count for R-port2 direction

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED16	Read/Write	Read	0–255: hop count for R-port2 256–65535: Reserved

7.3.6.6.4 V(path-PREFER_RP): Preferred R-port

This variable holds the preferred R-port for sending a frame from the local device to the peer device without regard for the RNMP or RNMS. This variable is determined as the R-port that has the smaller hop count value for the peer device. If the R-port1 hop count and R-port2 hop count have the same value, R-port1 is selected as the preferred R-port. This variable is defined as shown Table 60.

Table 60 – Preferred R-port

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED8	Read/Write	Read	0: Invalid 1: R-port1 2: R-port2 3–255: Reserved

7.3.6.6.5 V(path-DST_RP): Destination R-port

This variable holds the destination R-port for sending a frame from the local device to the peer device. In a line network, this variable has the same value as the Preferred R-port. However, in a ring network, this variable is determined based on the RNMP and RNMS position, because the preferred path may be blocked by the RNMP or RNMS. In this case, the destination R-port is selected as the other R-port. It is defined as shown in Table 61.

Table 61 – Destination R-port

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED8	Read/Write	Read	0: Invalid 1: R-port1 2: R-port2 3–255: Reserved

7.3.6.6.6 V(path-DEV_STATE): Device state

See 8.3.6.4.3.

7.3.6.6.7 V(path-MAC_ADDR): MAC address

See 8.3.6.4.7.

7.3.6.6.8 V(path-PORT_INFO): Port information

See 8.3.6.4.8.

7.3.6.6.9 V(path-PROTOCOL_VER): Protocol version

See 8.3.6.4.9.

7.3.6.6.10 V(path-DEV_TYPE): Device type

See 8.3.6.4.10.

7.3.6.6.11 V(path-DEV_DESC): Device description

See 8.3.6.4.11.

7.3.6.6.12 V(path-DEV_UID): Device UID

See 8.3.6.4.4.

7.3.6.6.13 V(path-DEV_UID_RP1): Device UID for R-port1

See 8.3.6.4.5.

7.3.6.6.14 V(path-DEV_UID_RP2): Device UID for R-port2

See 8.3.6.4.6.

7.3.6.6.15 C(path-IN_NET_CNT): In net count

This variable holds the number of times that the peer device has joined the network. When a line network is merged into an existing network, the variables for the newly joined devices are incremented together. This variable is defined as shown in Table 62.

Table 62 – In net count

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED16	Read/Write	Read	0–65535: the number of times that the peer device has joined the network

7.3.6.6.16 V(path-IN_NET_TIME): In net time

This variable holds the date and time when the peer device last joined in the network. This variable is defined as shown in Table 63.

Table 63 – In net time

Data type	Access type DLM	Access type DLS-user	Value/Description
TIMEOFDAY	Read/Write	Read	The date and time when the device last joined in the network

7.3.6.6.17 C(path-OUT_NET_CNT): Out net count

This variable holds the number of times that the peer device has been disconnected from the network. When a device or a group of devices are disconnected from the network, the variables for the disconnected devices are incremented together. This variable is defined as shown in Table 64.

Table 64 – Out net count

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED16	Read/Write	Read	0–65535: the number of times that the device has been disconnected from the network

7.3.6.6.18 V(path-OUT_NET_TIME): Out net time

This variable holds the date and time when the device was last disconnected from the network. This variable is defined as shown in Table 65.

Table 65 – Out net time

Data type	Access type DLM	Access type DLS-user	Value/Description
TIMEOFDAY	Read/Write	Read	The date and time when the device was last disconnected from the network

7.3.6.7 Variables, counters, timers, and queues to support path table management

7.3.6.7.1 Path table

The path table is composed of items related to the other devices on the network. It is managed by the DLM in the form of an array table filled with the path information of each device (see 8.3.6.6). The maximum size of the path table is a function of MAX_ADDR (see 8.3.6.1.1) as follows.

Path table: Array[n] of device's path information, n = MAX_ADDR + 1

7.4 General structure and encoding

7.4.1 Overview

The DLL and its procedures are necessary to provide services to the DLS-user using the services available from the physical layer. This clause describes the structure and semantics of the management application protocol data unit (MAPDU), the DLPDU, and the procedure commonly used in this specification. This portion is identical to, and fully compliant with, the ISO/IEC 8802-3 specification.

NOTE In this clause, any reference to bit k of an octet is a reference to the bit whose weight in a one-octet unsigned integer is 2^k. This is sometimes referred to as "little-endian" bit numbering.

7.4.2 MAPDU structure and encoding

The local MAC sublayer uses the service primitives provided by the physical layer service (PLS) sublayer specified by ISO/IEC 8802-3, Clause 2. The following service primitives provided by the PLS sublayer are mandatory.

a) MA-DATA request

b) MA-DATA indication

7.4.3 Common MAC frame structure, encoding and elements of procedure

7.4.3.1 MAC frame structure

7.4.3.1.1 MAC frame format for the RAPIEnet DLPDU

The DLPDU for the RAPIEnet is encapsulated in the data field of a MAC frame as specified by ISO/IEC 8802-3, Clause 3. The value of the length/type field is $88FE_H$, which is authorized and registered as the protocol identification number by the IEEE Registration Authority, to identify a RAPIEnet fieldbus frame. Figure 24 shows the RAPIEnet DLPDU structure.

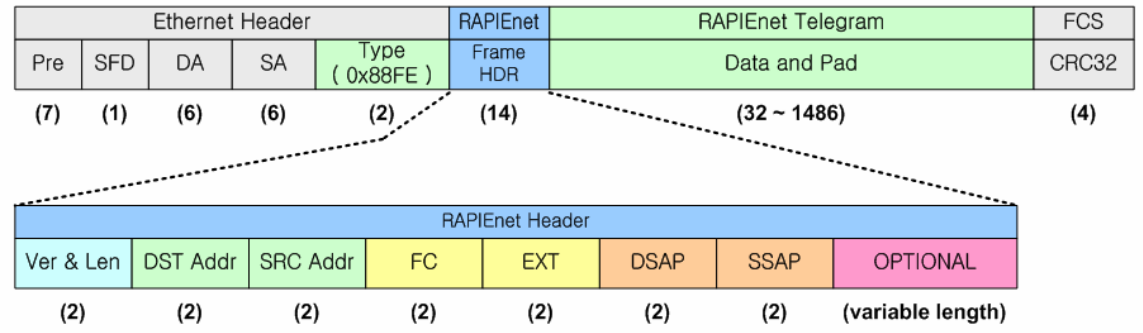


Figure 24 – Common MAC frame format for RAPIEnet DLPDU

7.4.3.1.2 MAC frame format for RAPIEnet fieldbus sporadic DLPDU

The MAC frame format used for RAPIEnet fieldbus sporadic data transmission is identical to the frame format of Ethernet V2.0 specified by ISO/IEC 8802.3, Clause 3, and the value of the length/type field is anything other than 0x88FE. Figure 25 shows the frame format for a RAPIEnet fieldbus sporadic DLPDU.

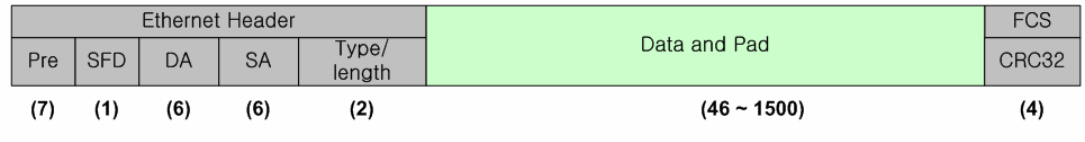


Figure 25 – MAC frame format for other protocols

7.4.3.2 Elements of the MAC frame

7.4.3.2.1 General

The elements of the MAC frame are the preamble, the start frame delimiter, the destination MAC address, the source MAC address, the length/type code, and the frame check sequence (FCS), all as specified by ISO/IEC 8802-3, Clause 3.

7.4.3.2.2 Preamble field

The preamble of MAC frame is identical to ISO/IEC 8802-3, Clause 3. This is a 7-octet field that is used to allow the physical signalling portion of the circuitry to reach its steady-state synchronization with the receiving frame timing. The preamble pattern is:

"10101010 10101010 10101010 10101010 10101010 10101010 10101010"

The bits are transmitted in order from left to right. The nature of the pattern is such that for Manchester encoding, it appears as a periodic waveform on the medium that enables bit synchronization. It should be noted that the preamble ends with a "0."

7.4.3.2.3 Start frame delimiter

The start frame delimiter (SFD) is identical to ISO/IEC 8802-3, Clause 3. The SFD field is the bit pattern sequence "10101011." It immediately follows the preamble pattern and indicates the start of a frame.

7.4.3.2.4 Address field

The address fields (both destination MAC address and source MAC address) are identical in structure and semantics to the address field of the basic MAC frame, described in ISO/IEC 8802-3, Clause 3. Each address field is 48 bits in length.

7.4.3.2.5 Destination MAC address field

The destination MAC address field is identical to ISO/IEC 8802-3, Clause 3. It specifies the device(s) for which the frame is intended and may be an individual or multicast (including broadcast) address. The destination MAC address is set to the corresponding DL-address by the DLM. RAPIEnet also defines a special MAC address, 00-E0-91-02-05-99 (NCM_MAC_ADDR) for sharing network management information using the DLM services. Every message received through the NCM_MAC_ADDR is delivered to the DLM to update the network management information. The message is not forwarded by the MAC layer but the message is examined and forwarded by the DLM.

7.4.3.2.6 Source MAC address field

The source MAC Address field is identical to ISO/IEC 8802-3, Clause 3. This field specifies the device sending the frame and is not interpreted by the DLE or the CSMA/CD MAC sublayer.

7.4.3.2.7 Length/type field

The length/type field is identical to ISO/IEC 8802-3, Clause 3. To be identified as a RAPIEnet frame, the value of the length/type field is set to 0x88FE, which is authorized and registered as the protocol identification number for RTE-RAPIEnet by the IEEE Registration Authority. Every frame with a value other than 0x88FE is identical to the frame in ISO/IEC 8802-3, Clause 3, and is processed as a RAPIEnet fieldbus sporadic data frame.

7.4.3.3 Elements of the RAPIEnet DLPDU

7.4.3.3.1 Version and length

This field stores the protocol version and the length of a RAPIEnet telegram or data field. This version and length field is specified in Figure 26. The version is represented by 2 bits for the major version and 3 bits for the minor version, and the length is given by 11 bits.

Version and Length															
Version					Length										
15	14	13	21	11	10	9	8	7	6	5	4	3	2	1	0

Figure 26 – Version and length field

The parts of this field and permissible values are described in Table 66.

Table 66 – Version and length

Field Name		Position	Value/Description
Version	Major	b15–b14	RAPIEnet Protocol major version 0x00: major version 1 0x01: major version 2 0x02: major version 3 0x03: major version 4
	Minor	b13–b11	RAPIEnet Protocol minor version 0x00: minor version 0 0x01: minor version 1 0x02: minor version 2 0x03: minor version 3 0x04: minor version 4 0x05: minor version 5 0x06: minor version 6 0x07: minor version 7
Length		b10–b0	Frame length including FCS field

7.4.3.3.2 DST_addr

This field indicates the destination DL-address of the node to which the frame is sent. This value is represented as shown in Figure 27.

Destination DL−address															
DST_Addr															
15	14	13	21	11	10	9	8	7	6	5	4	3	2	1	0

Figure 27 – DST_addr field

The separate field and its permissible values are described in Table 67.

Table 67 – Destination DL-address

Field Name	Position	Value/Description
DST_addr	b15–b0	0xFFFF: broadcast address 0xFFFE: network control address (C_NCM_ADDR) 0xFFFD–0xFFDE: user-defined multicast address 0xFFDD: invalid address 0x0100–0xFFDC: reserved 0x0000–0x00FF: regular RAPIEnet DL-address

7.4.3.3.2.1 Broadcast address

If the destination DL-address is 0xFFFF, the destination MAC address field contains the ISO/IEC 8802-3 MAC address.

7.4.3.3.2.2 Network control address

If the destination DL-address is 0xFFFE (C_NCM_ADDR), the destination MAC address field contains C_NCM_MAC_ADDR. However, NCM_LINK_ACTV and NCM_ADV_THIS messages are transmitted using C_NCM_ADDR as the destination DL-address.

NOTE C_NCM_ADDR cannot be accessed by the DLS-user.

7.4.3.3.2.3 User-defined multicast address

A user-defined multicast address is used to indicate multiple recipients. However, user-defined multicast addressing is not a mandatory feature in this specification. It is designed for use in a special application system that requires multicast communication. Therefore, user-defined multicast addressing is not interoperable between heterogeneous devices. The destination DL-address range from 0xFFFD–0xFFDE is used to specify the user-defined multicast address. However, the method of using the user-defined multicast address is not specified in this PAS and is considered a local responsibility. This specification does not restrict the use of user-defined multicast addresses, nor is it a mandatory feature.

7.4.3.3.3 SRC_addr

This field indicates the source DL-address of the node from which the frame is generated. This value is represented as shown in Figure 28.

Source DL-address															
SRC_Addr															
15	14	13	21	11	10	9	8	7	6	5	4	3	2	1	0

Figure 28 – SRC_addr field

The separate field and its permissible values are described in Table 68.

Table 68 – Source DL-address

Field Name	Position	Value/Description
SRC_addr	b15–b0	Source DL-address

7.4.3.3.4 Frame control (FC)

The frame control field indicates the frame control information. This value is represented as shown in Figure 29.

Frame Control															
VoE	RES	PRI		ToS				NCMT							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Where
VoE: Validation of Extension code
RES: Reserved
PRI : Priority
ToS: Type of Service
NCMT: Network control message type

Figure 29 – Frame control field

The separate field and its permissible values are described in Table 69.

Table 69 – Frame control

Field name	Position	Value/Description
Validation of extension code (VoE)	b15	0x00: EXT Code is invalid 0x01: EXT Code is valid
Reserved	b14	Reserved
Priority (PRI)	b13–b12	0x00: lowest priority … 0x03: highest priority
Type of service (ToS)	b11–b8	0x00: Network Control Message (NCM) 0x01: unconfirmed service request 0x02–0x0F: Reserved
Network Control Message Type (NCMT)	b7 – b0	0x00: reserved 0x01: NCM_LINK_ACTV 0x02: NCM_ADV_THIS 0x03: NCM_LINE_START 0x04: NCM_RING_START 0x05–0xFF:reserved

7.4.3.3.4.1 Validation of extension code (VoE)

If the frame has the extension field, VoE is set to TRUE; otherwise VoE is set to FALSE.

7.4.3.3.4.2 Priority

This field indicates the frame priority. This field contains the value of the message priority parameter for the DL service. The highest priority is 0x03 and the lowest is 0x00.

7.4.3.3.4.3 Type of service (ToS)

This field indicates the type of DL service. A value of 0x00 indicates a network control message among DLMs, and 0x01 indicates the unconfirmed service request among DLS-users.

7.4.3.3.4.4 Network Control Message Type (NCMT)

NCMT field indicates the type of network control message.

7.4.3.3.4.4.1 NCM_LINK_ACTV

NCMT: 0x01

This NCMT indicates that a new RAPIEnet link is established through the R-port. This network control message is transmitted through the newly activated R-port. The destination DL-address contains C_NCM_ADDR. When the DLM receives this message, the DLM increments the hop count in the frame and forwards the frame through the other R-port. This message is discarded by the LNM or the device that generated the message.

7.4.3.3.4.4.2 NCM_ADV_THIS

NCMT: 0x02

This network control message is used to transmit the recipient's local device information when the recipient receives NCM_LINK_ACTV message from a new device on the network. This message is transmitted through the R-port that is used to receive the NCM_LINK_ACTV message. The destination DL-address contains C_NCM_ADDR. When the DLM receives this message, the DLM increments the hop count in the frame and forwards the frame through the other R-port. This message is discarded by the LNM or the device that generated the message.

7.4.3.3.4.4.3 NCM_LINE_START

NCMT: 0x03

This network control message is used to broadcast that the network topology has been automatically configured as a line network. This message is initiated by the DLM whose state is changed to LNM when the existing line network is divided into two line networks, or when a link failure is detected in a ring network and the network is reconfigured as a line network. This message is broadcast on the network using the broadcast address.

7.4.3.3.4.4.4 NCM_RING_START

NCMT: 0x04

This network control message is used to broadcast that the network topology has been automatically configured as a ring network. This message is initiated and broadcast through both R-ports by the DLM whose state is changed to RNMP.

7.4.3.3.5 Extension (EXT)

This field exists when the VoE bit in the frame control field is set to TRUE. The extension field is specified as shown in Figure 30.

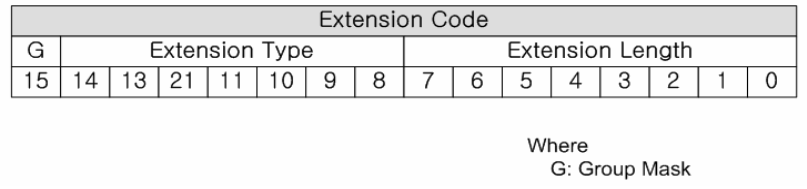


Figure 30 – Extension field

The separate field and its permissible values are described in Table 70.

Table 70 – Extension

Field name	Position	Value/Description
Group Mask Enable	b15	0x0: Group mask is enabled 0x1: Group mask is disabled
Extension Type	b14–b8	0: Invalid extension type 1–127: reserved for future use
Extension Length	b7–b0	0–255: the length of extension field

7.4.3.3.5.1 Group mask enable

Group Mask Enable is a bit field to specify whether the frame is to be accepted by the peer device or not, when the frame is broadcast or multicast. When the value is set to TRUE, group mask is enabled in the peer device that receives the frame. Otherwise, group mask is disabled in the peer device. When the group mask is enabled, the group mask fields are appended in the option field (see 8.4.3.3.8).

7.4.3.3.5.2 Extension type

This field indicates the type of extension field. The value 0x00 indicates an invalid extension type and the other values are reserved for future use.

7.4.3.3.5.3 Extension length

This field indicates the length of the extension field. When group mask enable is set to TRUE and extension type is set to 0x00, the extension length specifies the length of the group mask field. When group mask enable is set to FALSE and extension type is set to a value other than 0x00, extension length specifies the length of the extension field. When group mask enable is set to TRUE and extension type is not set to 0x00, the first two bytes specify the length of the group mask field and the next two bytes specify the extension type.

7.4.3.3.6 DSAP

This field indicates the SAP of the DLE to which the DLPDU is sent. The permissible values are in the range 0–65535. The DSAP is specified as shown in Figure 31 and Table 71.

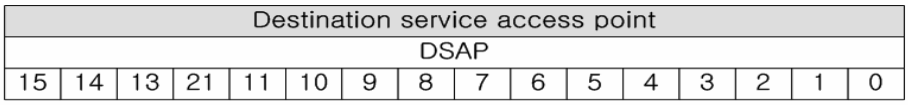


Figure 31 – DSAP field

Table 71 – Destination service access point

Field name	Position	Value/Description
DSAP	b15–b0	Service access point of destination DLE

7.4.3.3.7 SSAP

This field indicates the SAP of the DLE from which the DLPDU is generated. The permissible values are in the range 0–65535. The DSAP is specified as shown in Figure 32 and Table 72.

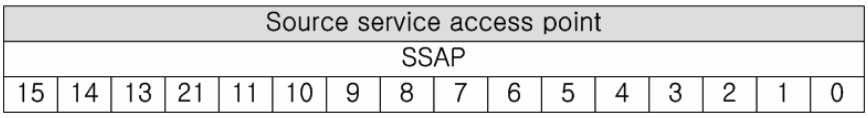


Figure 32 – Source service access point field

Table 72 – Source service access point

Field name	Position	Value/Description
SSAP	B15–b0	Service access point of source DLE

7.4.3.3.8 Option

This field indicates the option field when the VoE (see 8.4.3.3.4.1) is set to TRUE. The option field is used to indicate group mask information or other additional information. The maximum length of the option field is limited to 256 bytes.

7.4.3.3.8.1 Length of group mask and extension information

This field indicates the length of group mask and extension information. When group mask enable is set to TRUE and the extension type is not 0x00, the first two bytes indicate the length of the group mask field and the next two bytes indicate the length of the extension type field. When group mask enable is set to FALSE and the extension type is 0x00, the length of group mask and extension information field is ignored.

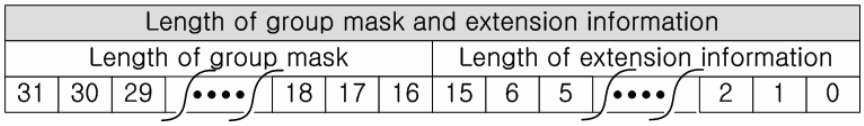


Figure 33 – Length of group mask and extension information

7.4.3.3.8.2 Group mask

This field uses a bit sequence to indicate the receipt selection of a message. When group mask enable is set to TRUE, the group mask field is appended in 4-byte units, i.e., 4, 8, 12 … 32 bytes. Each bit indicates the receipt selection of the frame for the corresponding DL-address. A 1 means TRUE for frame receipt and 0 means FALSE. The first bit indicates the frame receipt option for highest DL-address. Figure 34 shows the bit sequence order of group mask field when its length is set to 255. When the extension type is set to 0x00, the group

mask field is appended to the option field. Otherwise, the group mask field is appended after the length of group mask and extension information fields.

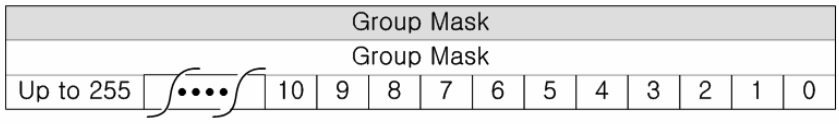


Figure 34 – Group mask option field

7.4.3.3.8.3 Extension information

This field is reserved for future extension. This field contains 4 bytes of extension information.

7.4.3.3.9 Data and pad

This field indicates the data field received from a DLS-user.

7.4.3.3.10 Frame check sequence

The FCS construction, polynomial, and expected residual are identical to ISO/IEC 8802-3, Clause 3. In this clause, any reference to bit K of an octet is a reference to the bit whose weight in a one-octet unsigned integer is 2^K.

NOTE This is sometimes referred to as “little-endian” bit numbering.

For most of the protocol types in this specification, as in other international specifications (for example, ISO/IEC 3309, ISO/IEC 8802 and ISO/IEC 9314-2), DLPDU-level error detection is provided by calculating and appending a multi-bit FCS to the other DLPDU fields during transmission. This forms a systematic code word of length n, consisting of k DLPDU message bits followed by $n - k$ (=32) redundant bits. During reception, a check is made to ensure that the message and the concatenated FCS form a legal (n, k) code word. The mechanism for this check is as follows.

The generic form of the generator polynomial for this FCS construction is specified in equation (6) of 8.4.3.3.10 and the polynomial for the receiver’s expected residue is specified in equation (11) of 8.4.3.3.11. The specific polynomials for each DL-protocol type are specified in Table 73.

Table 73 – FCS length, polynomials and constants

Protocol	Item	Value
RAPIEnet	*n-k*	32
	G(X)	$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + 1$ (notes 1, 2, 3)
	R(X)	$X^{31}+X^{30}+X^{26}+X^{25}+X^{24}+X^{18}+X^{15}+X^{14}+X^{12}+X^{11}+X^{10}+X^8+X^6+X^5+X^4+X^3+X+1$ (notes 3, 4)

	NOTE 1 Code words D(X) constructed from this G(X) polynomial have Hamming distance 4 for lengths ≤ 22 901 octets, Hamming distance 5 for lengths ≤ 371 octets, Hamming distance 6 for lengths ≤ 33 octets, Hamming distance 7 for lengths ≤ 21 octets, and Hamming distance 8 for lengths ≤ 11 octets. NOTE 2 This G(X) polynomial is relatively prime to all and is thus not compromised by any of the polynomials commonly used in DCEs (modems), i.e., the differential encoding polynomial $1 + X^{-1}$ and all primitive scrambling polynomials of the form $1 + X^{-j} + X^{-k}$. NOTE 3 These are the same polynomials and methods as specified in ISO/IEC 8802-3 (Ethernet). NOTE 4 The remainder R(x) should be 1100 0111 0000 0100 1101 1101 0111 1011 (X^{31} to X^0, respectively) in the absence of errors.

7.4.3.3.11 Action at the sending DLE

The original message (i.e., DLPDU without an FCS), the FCS, and the composite message code word (concatenated DLPDU and FCS) are regarded as vectors M(X), F(X), and D(X), of dimension k, $n - k$, and n, respectively, in an extension field over GF(2). If the message bits are $m_1 \ldots m_k$ and the FCS bits are $f_{n-k-1} \ldots f_0$, where

$m_1 \ldots m_8$	form the first octet sent;
$m_{8N-7} \ldots m_{8N}$	form the *N*th octet sent;
$f_7 \ldots f_0$	form the last octet sent;
m_1	is sent by the first physical layer symbol(s) of the message and f_0 is sent by the last physical layer symbol(s) of the message (not counting physical layer framing information).

NOTE This "as transmitted" ordering is critical to the error-detection properties of the FCS.

Then, the message vector M(X) is

$$M(X) = m_1X^{k-1} + m_2X^{k-2} + \ldots + m_{k-1}X^1 + m_k \quad (1)$$

and the FCS vector F(X) is

$$F(X) = f_{n-k-1}X^{n-k-1} + \ldots + f_0 \text{ (for the case of k = 32)} \quad (2)$$
$$= f_{31}X^{31} + \ldots + f_0$$

The composite vector D(X), for the complete DLPDU, is constructed as the concatenation of the message and FCS vectors

$$D(X) = M(X)\, X^{n-k} + F(X) \quad (3)$$
$$= m_1X^{n-1} + m_2X^{n-2} + \ldots + m_kX^{n-k} + f_{n-k-1}X^{n-k-1} + \ldots + f_0$$
$$= m_1X^{n-1} + m_2X^{n-2} + \ldots + m_kX^{32} + f_{31}X^{31} + \ldots + f_0 \text{ (for the case of k = 32)}$$

The DLPDU presented to the physical layer shall consist of an octet sequence in the specified order.

The redundant check bits $f_{n-k-1} \ldots f_0$ of the FCS are the coefficients of the remainder F(X), after division by G(X), of $L(X)\,(X^k + 1) + M(X)\, X^{n-k}$ where G(X) is the degree ***n-k*** generator polynomial for the code words

$$G(X) = X^{n-k} + g_{n-k-1}X^{n-k-1} + \ldots + 1 \quad (4)$$

and L(X) is the maximal weight (all ones) polynomial of degree ***n-k-1***

$$L(X) = X^{n-k-1} + X^{n-k-2} + \ldots + X + 1 \quad (5)$$
$$= X^{31} + X^{14} + X^{13} + X^{12} + \ldots + X^2 + X + 1 \text{ (for the case of k = 32)}$$

That is,

$$F(X) = L(X)\,(X^k + 1) + M(X)\,X^{n-k} \text{ (modulo } G(X)) \quad (6)$$

NOTE The L(X) terms are included in the computation to detect initial or terminal message truncation or extension by adding a length-dependent factor to the FCS.

NOTE As a typical implementation when n-k = 32, the initial remainder of the division is preset to all ones. The transmitted message bit stream is multiplied by X^{n-k} and divided (modulo 2) by the generator polynomial G(X), specified in equation (4). The ones complement of the resulting remainder is transmitted as the (n-k)-bit FCS, with the coefficient of X^{n-k-1} transmitted first.

7.4.3.3.12 Action at the receiving DLE

The octet sequence indicated by the PhE is concatenated into the received DLPDU and FCS, and regarded as a vector V(X) of dimension ***u***

$$V(X) = v_1X^{u-1} + v_2X^{u-2} + \ldots + v_{u-1}X + v_u \quad (7)$$

NOTE Because of errors, u can be different from n, the dimension of the transmitted code vector.

The remainder R(X) is computed for V(X), the received DLPDU and FCS, by a method similar to that used by the sending DLE (see 7.4.3.3.11) in computing F(X).

$$R(X) = L(X)\,X^u + V(X)\,X^{n-k} \text{ (modulo } G(X)) \quad (8)$$
$$= r_{n-k-1}X^{n-k-1} + \ldots + r_0$$

E(X) is defined to be the error code vector of the additive (modulo 2) differences between the transmitted code vector D(X) and the received vector V(X) resulting from errors encountered in the PhS provider between the sending and receiving DLEs.

$$E(X) = D(X) + V(X) \quad (9)$$

If no error has occurred, then E(X) = 0, and R(X) will equal a non-zero constant remainder polynomial

$$R_{ok}(X) = L(X)\,X^{n-k} \text{ (modulo } G(X)) \quad (10)$$

whose value is independent of D(X). Unfortunately, R(X) will also equal $R_{ok}(X)$ in those cases where E(X) is an exact non-zero multiple of G(X), in which case there are undetectable errors. In all other cases, R(X) will not equal $R_{ok}(X)$; such DLPDUs are erroneous and can be discarded without further analysis.

NOTE In a typical implementation, the initial remainder of the division is preset to all ones. The received bit stream is multiplied by X^{n-k} and divided (modulo 2) by the generator polynomial G(X), specified in equation (8).

7.4.4 Order of bit transmission

The order of bit transmission is identical to ISO/IEC 8802-3, Clause 3. Each octet of the DLPDU with the exception the FCS is transmitted with the low-order bit first.

7.4.5 Invalid DLPDU

An invalid DLPDU shall be defined as one that meets at least one of the following conditions; this is almost identical to ISO/IEC 8802-3, Clause 3.

a) The frame length is inconsistent with a length value specified in the length/type field. If the length/type field contains a type value defined by ISO/IEC 8802-3, 3.2.6, then the frame length is assumed consistent with this field and should not be considered an invalid DLPDU on this basis.

b) It is not an integral number of octets in length.

c) The bits of the incoming DLPDU excluding the FCS field do not generate a CRC value identical to the one received.

d) It is inconsistent with a F-type value of RAPIEnet fieldbus DLPDU.

The contents of invalid DLPDUs shall not be passed to the DL-user or DLE. The occurrence of an invalid DLPDU may be communicated to the network management.

NOTE Invalid DLPDUs may be ignored, discarded, or used in a private manner by a DL-user other than the RTE DL-user. The use of such DLPDUs is beyond the scope of this PAS.

7.5 DLPDU structure and procedure

7.5.1 General

This clause defines the structure, contents, and encoding for each type and format of the DLPDU, along with procedural elements. Subclauses describe the structure, contents, parameters, and encoding of the DLPDU, along with the RAPIEnet-specific part of the DLPDU structure, which is shown in Figure 26. The aspects relating to sending and receiving by DLS-users and their DLEs are also described.

NOTE In this section, any reference to bit K of an octet is a reference to the bit whose weight in a one-octet unsigned integer is 2^K, and this is sometimes referred to as “little-endian” bit numbering.

7.5.2 Common DLPDU field

Figure 35 shows the common DLPDU field. The version and length field is common to every DLPDU and is not described further for each DLPDU type.

Version and Length															
Version					Length										
15	14	13	21	11	10	9	8	7	6	5	4	3	2	1	0

Figure 35 – Common DLPDU field

7.5.2.1 Version

This field indicates the RAPIEnet protocol version with a 5-bit sequence. The highest 2 bits specify the major version and the lowest 3 bits specify the minor version (see 8.4.3.3.1).

7.5.2.2 Length

This field indicates the length of the data field in bytes including the FCS field. The permissible values are in the range 12–1498.

7.5.3 DL-DATA transfer

7.5.3.1 DT DLPDU

7.5.3.1.1 General

The DT DLPDU is used to carry RAPIEnet data from one device to another.

7.5.3.1.2 DT DLPDU structure

Figure 36 shows the process of building a DT DLPDU and Figure 37 shows the structure of the DT DLPDU.

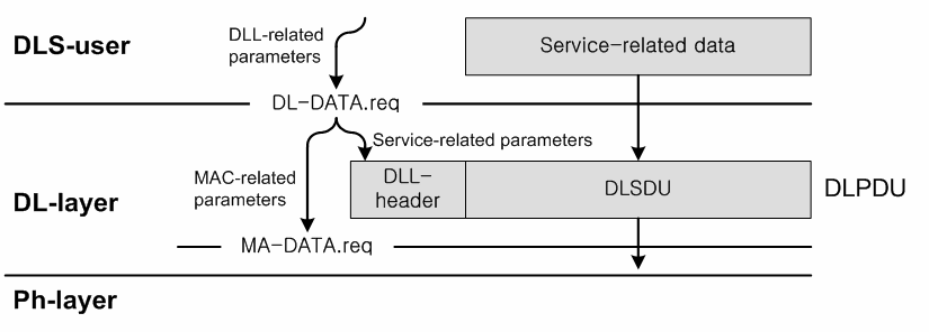


Figure 36 – Building a DT DLPDU

Figure 37 – DT DLPDU structure

7.5.3.1.3 DT DLPDU parameters

Table 24 shows the list of DT DLPDU parameters.

Table 74 – DT DLPDU parameters

Parameter	Data type	Value/Description
DST_addr	UNSIGNED16	Destination DL-address [Optional] -Broadcast -Multicast (user-defined) -Unicast
Priority	UNSIGNED8	0x00–0x03 0x04–0xFF: Reserved
DSAP	UNSIGNED16	Destination DLSAP of remote device
SSAP	UNSIGNED16	Source DLSAP
Group mask	-	Intended destination device bit array. Only when broadcast or multicast.

Parameter	Data type	Value/Description
Group mask length	UNSIGNED8	Length of group mask. Only when broadcast or multicast
Data	-	DLSDU
Data length	UNSIGNED16	Length of DLSDU

7.5.3.1.3.1 DST_addr

This parameter indicates the destination DL-address of the DLE(s) for which the DLPDU is intended. This may be an individual or multicast (including broadcast) DL-address, not its MAC address. The destination address for unicast is explicitly assigned a value between 0x0000 and 0x00FF. A user-defined multicast address is assigned a value in the range 0xFFDE–0xFFFD. A broadcast address is assigned the value 0xFFFF. The values 0x0100–0xFFDC are reserved. The value 0xFFDD indicates an invalid address (see 8.4.3.3.2).

7.5.3.1.3.2 Priority

This field indicates the frame priority with a 2-bit sequence. This field contains the value of the message priority parameter of the DL service. The highest priority is 0x03, and 0x00 is the lowest (see 8.4.3.3.4.2)

7.5.3.1.3.3 DSAP

See 8.4.3.3.6

7.5.3.1.3.4 SSAP

See 8.4.3.3.7

7.5.3.1.3.5 Group mask

See 8.4.3.3.8.2

7.5.3.1.3.6 Group mask length

See 8.4.3.3.8.1

7.5.3.1.3.7 Data

This data field is the DLSDU.

7.5.3.1.3.8 Data length

This field indicates the length of the data field in bytes. The permissible values are in the range 0–1486.

7.5.3.1.4 Sending

When the local DLS-user initiates a DL-DATA service request to transfer a DLSDU to a peer DLS-user, the DLSDU is stored in the RT-queue and transmitted by the DLPM using a DT DLPDU. The group mask option is available when the DL-DATA DLPDU is broadcast or multicast.

Table 75 shows the required data link service primitives and parameters to send a DT DLPDU. To send a DT DLPDU to a peer DLE, the DLPM queries the peer device's path information to the DLM using the DLM-GET_PATH service. Then, the DT DLPDU is transmitted using an MA-DATA service request primitive.

Table 75 – Primitives exchanged between DLS-user and DLE to send a DT DLPDU

Primitive	Source	Associated parameters
DL-DATA request	DLS-user	DST_addr DSAP SSAP Priority Group mask Group mask length Data Data length
DLM-GET_PATH request	DLE (DLPM)	DST_addr
DLM-GET_PATH confirm	DLM	Status R-port MAC address
MA-DATA request	DLE (DLPM)	DLPDU DLPDU length

7.5.3.1.5 Receiving

When a DT DLPDU is received through the MAC layer, the DLPM checks that it is valid and if it is intended for the local DLS-user. If the DT DLPDU is broadcast or multicast, the DLPM examines the group mask field to check if the DT DLPDU is intended for the local DLS-user (see 8.4.3.3.5). If the DLPDU is intended for the local DLS-user, the DLPM extracts the DLSDU from the received DT DLPDU and delivers the DLSDU to the appropriate DLS-user using a DL-DATA indication primitive. If no DLS-user is registered for the DSAP in the received DT DLPDU, the DLPM discards the DT DLPDU just received.

Table 76 shows the required data link service primitives and parameters to receive a DT DLPDU. When a DT DLPDU is received through the MA-DATA indication service primitive, the DLPM extracts the DLSDU from the received DT DLPDU and delivers the DLSDU to the appropriate DLS-user.

Table 76 – Primitives exchanged between DLS-user and DLEs to receive a DT DLPDU

Primitive	source	Associated parameters
MA-DATA indication	MAC	DLPDU DLPDU length
DL-DATA indication	DLPM	DLSDU DLSDU length

7.5.4 DL-SPDATA transfer

7.5.4.1 SPDT DLPDU

7.5.4.1.1 General

SPDT DLPDUs are used to carry non-RAPIEnet data, such as TCP/IP or UDP.

7.5.4.1.2 SPDT DLPDU structure

An SPDT DLPDU is composed of an ISO/IEC MAC frame and DLSDU. The RAPIEnet header is not included in the SPDT DLPDU. Figure 38 shows the process of building an SPDT DLPDU and Figure 25 shows the structure of the SPDT DLPDU.

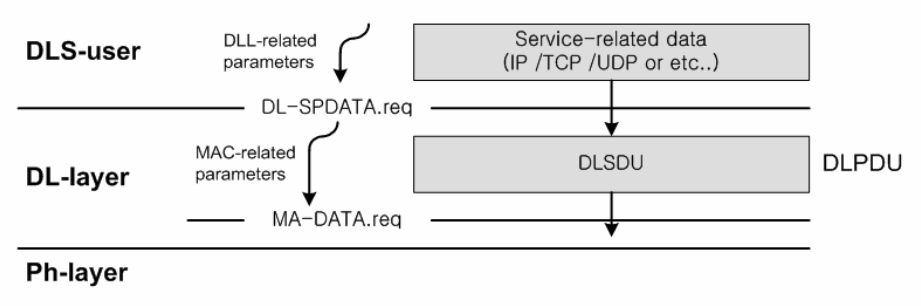


Figure 38 – SPDT DLPDU structure

7.5.4.1.3 SPDT DLPDU parameters

Table 77 shows the parameter list for the SPDT DLPDU.

Table 77 – SPDT DLPDU parameters

Parameter	Data type	Value/Description
Data	-	DLSDU
Data length		Length of DLSDU

7.5.4.1.3.1 Data

This field indicates the DLSDU, which is to be transmitted as a standard ISO/IEC 8802-3 Ethernet frame including common protocol data units, such as TCP/IP or UDP.

7.5.4.1.3.2 Data length

This field indicates the length of the data field in bytes. The permissible values are in the range 0–1514.

7.5.4.1.4 Sending

When the local DLS-user initiates a DL-SPDATA service request to transfer a DLSDU to the peer DLS-user, the DLSDU is stored in the NRT-queue and transmitted by the DLPM using an SPDT DLPDU. In this case, the DLSDU is transmitted as the SPDT DLPDU without a RAPIEnet header.

Table 78 shows the required data link service primitives and parameters to send an SPDT DLPDU. The SPDT DLPDU is transmitted by the DLPM using a MA-DATA request service primitive.

Table 78 – Primitive exchanged between DLS-User and DLEs to send an SPDT DLPDU

Primitive	Source	Associated parameters
DL-SPDATA request	DLS-User	Data: DLSDU Data length: length of DLSDU

MA-DATA request	DLPM	DLPDU DLPDU length
MA-DATA confirm	MAC	Status

7.5.4.1.5 Receiving

When an SPDT DLPDU is received through the MAC layer, the DLPM checks that it is valid and if it is intended for the local DLS-user. If this is the case, the DLPM extracts the DLSDU from the SPDT DLPDU and delivers it to the appropriate DLS-user using a DL-DATA indication primitive.

Table 79 shows the required data link service primitives and parameters to receive an SPDT DLPDU. When an SPDT DLPDU is received through the MA-DATA indication service primitive, the DLPM extracts the DLSDU from the received SPDT DLPDU and delivers the DLSDU to the appropriate local DLS-user.

Table 79 – Primitives exchanged between DLS-user and DLEs to receive an SPDT DLPDU

Primitive	Source	Associated parameters
MA-DATA indication	DLPM	DLPDU DLPDU length
DL-SPDATA indication	MAC	DLSDU DLSDU length

7.5.5 Network control messages

Network control message (NCM) DLPDUs are used to transfer the network control messages among DLMs. Five message types are provided to share the network information.

7.5.5.1 NCM_LA DLPDU

7.5.5.1.1 General

NCM_LA DLPDU is used to transfer local device information to the peer device when a link is established. When the DLM detects the change of link status through the Ph-LINK_STATUS_CHANGE indication primitive, the DLM queries the link status information of the physical layer. After that, the DLM generates an NCM_LA DLPDU to send the changed link information to the peer device over the newly established link.

7.5.5.1.2 NCM_LA DLPDU structure

NCM messages are based on the basic RAPIEnet DLPDU structure shown in Figure 27. Figure 39 shows the structure of the NCM_LA DLPDU after the Ethernet header.

DST Addr (C_NCM_ADDR)	SRC Addr	FC	EXT	DSAP	SSAP	DATA (Local Device Information)

Figure 39 – NCM_LA DLPDU structure

7.5.5.1.3 NCM_LA DLPDU parameters

Table 80 shows the parameter list of the NCM_LA DLPDU.

Table 80 – NCM_LA DLPDU parameters

Parameter	Data type	Description
DST_addr	UNSIGNED16	C_NCM_ADDR (see 8.4.3.3.2)
SRC_addr	UNSIGNED16	Local DL-address DL_ADDR (see 8.4.3.3.3)
NCMT	UNSIGNED8	NCM_LINK_ACTV (see 7.5.4.3.2.3)
DLMDU	-	DL management data unit Local device information (see 8.3.6.4)
Length	UNSIGNED16	Length of DLMDU
R-port	UNSIGNED8	Newly link-activated R-port

7.5.5.1.4 Sending

When the DLM detects the change in the link status through the Ph-LINK_STATUS_CHANGE indication primitive, the DLM queries the link status information of the physical layer. If the link is newly established, the DLM generates NCM_LA DLPDU to send the changed link information to the peer device over the newly established link. NCM_LA DLPDU is stored in the RT-queue and transmitted by the DLPM using the MA-DATA request primitive.

7.5.5.1.5 Receiving

The C_NCM_ADDR is used as the destination DL-address in the NCM_LA DLPDU. When an NCM_LA DLPDU is received through the MAC layer, the DLM increases the hop count field in the DLPDU and forwards the DLPDU through the other R-port. The NCM_LA DLPDU is discarded by the LNM or the device that generated the DLPDU. NCM_LA DLPDU is received and processed by the DLM as follows.

a) The DLM checks whether the DLPDU is generated by the device itself. If this is the case and the DLPDU is received through the other R-port, it means the network is configured as a ring network. If this is the case, proceed to step f) and finish the procedure without step g). Otherwise, proceed to step b).

b) Set the R-port's linked device type in the PORT_INFO (see 8.3.6.4.8) to "homogeneous device type".

c) Extract the device path information (DPI) (see 8.3.6.6) from the DLMDU and check for a DL-address collision using the DPI. If the source address is not registered in the DPI, generate an EVENT_IN_DEVICE event. If a local DL-address collision is detected, set the DEV_FLAG (see 8.3.6.4.2) to "Local DL-address Collision" and generate an EVENT_THIS_ADDR_COLLISION event. If a network DL-address collision is detected, set the NET_FLAG (see 8.3.6.5.10) to "Network DL-address Collision Status" and generate an EVENT_NET_ADDR_COLLISION event. Notification of any generated event is sent to the local DLMS-user using the DLM-EVENT indication service primitive.

d) Increment the HOP_CNT (see 8.3.6.4.12) in the received DLMDU.

e) Forward the modified NCM_LA DLPDU through the other R-port.

f) Check the DLM state trigger event condition using the PORT_INFO and perform the DLM state transition (see 8.6.3.3).

g) Generate an NCM_AT DLPDU and transmit the DLPDU through the opposite R-port.

7.5.5.2 NCM_AT DLPDU

7.5.5.2.1 General

An NCM_AT DLPDU is used to transfer local device information to the other devices on the network when a device receives an NCM_LA DLPDU. This DLPDU is transmitted through the R-port that is used to receive the NCM_LA DLPDU.

7.5.5.2.2 NCM_AT DLPDU structure

See 8.5.5.1.2

7.5.5.2.3 NCM_AT DLPDU parameters

Table 81 shows the parameter list of NCM_AT DLPDU.

Table 81 – NCM_AT DLPDU parameters

Parameter	Data type	Description
DST_addr	UNSIGNED16	C_NCM_ADDR (see 8.4.3.3.2.2)
SRC_addr	UNSIGNED16	Local DL-address DL_ADDR (see 8.4.3.3.3)
NCMT	UNSIGNED8	NCM_ADV_THIS (see 7.5.4.3.2.3)
DLMDU	-	Local device information (see 8.3.6.4)
Length	UNSIGNED16	Length of DLMDU
R-port	UNSIGNED8	R-port used to receive the NCM_LA DLPDU

7.5.5.2.4 Sending

When the DLM receives an NCM_LA DLPDU, the DLM generates an NCM_AT DLPDU for the DLMDU including the local device information. The DLMDU is stored in the RT-queue and transmitted by the DLMP through the R-port through which the NCM_LA DLPDU was received. In this case, C_NCM_ADDR is used as the destination DL-address.

7.5.5.2.5 Receiving

The C_NCM_ADDR is used as the destination address in an NCM_AT DLPDU. When an NCM_AT DLPDU is received through the MAC layer, the DLM increments the hop count field in the DLPDU and forwards the DLPDU through the other R-port. The NCM_AT DLPDU is discarded by the LNM or the device that generated the DLPDU. The NCM_AT DLPDU is received and processed by the DLM as follows.

a) The DLM checks whether the DLPDU was generated by the device itself. If the DLPDU was generated locally and received through the other R-port, it means the network is configured as a ring network. In this case, proceed to step f). Otherwise, proceed to step b).

b) Set the R-port's linked device type in the PORT_INFO (see 8.3.6.4.8) to "homogeneous device type."

c) Extract the device path information (DPI) (see 8.3.6.6) from the DLMDU and check for a DL-address collision using the DPI. If the source address is not registered in the DPI, generate an EVENT_IN_DEVICE event. If a local DL-address collision is detected, set the DEV_FLAG (see 8.3.6.4.2) to "Local DL-address Collision" and generate an EVENT_THIS_ADDR_COLLISION event. If a network DL-address collision is detected, set the NET_FLAG (see 8.3.6.5.10) to "Network DL-address Collision Status" and generate an EVENT_NET_ADDR_COLLISION event. Notification of any generated event is sent to the local DLMS-user using the DLM-EVENT indication service primitive.

d) Increment HOP_CNT (see 8.3.6.4.12) in the received DLMDU.

e) Forward the modified NCM_LA DLPDU through the other R-port.

f) Check the DLM state trigger event condition using the PORT_INFO and perform the DLM state transition (see 8.6.3.3).

7.5.5.3 NCM_LS DLPDU

7.5.5.3.1 General

The NCM_LS DLPDU is used to indicate that the network is automatically configured as a line topology when a line network is newly established, or that an existing line network is divided into two lines, or that a ring network has been reconfigured as a line network. This DLPDU is generated by the device that is selected as the LNM on the network.

7.5.5.3.2 NCM_LS DLPDU structure

See 8.5.5.1.2

7.5.5.3.3 NCM_LS DLPDU parameters

Table 82 shows the NCM_LS DLPDU parameter list.

Table 82 – NCM_LS DLPDU parameters

Parameter	Data type	Description
DST_addr	UNSIGNED16	Broadcast address(see 8.4.3.3.2.1)
SRC_addr	UNSIGNED16	Local DL-address DL_ADDR (see 8.4.3.3.3)
NCMT	UNSIGNED8	NCM_LINE_START (see 7.5.4.3.2.3)
DLMDU	-	Local device information (see 8.3.6.4)
Length	UNSIGNED16	Length of DLMDU
R-port	UNSIGNED8	link-activated R-port

7.5.5.3.4 Sending

When the state of the DLM is changed to LNM, the DLM generates an NCM_LS DLPDU to inform every device on the network of the change in network topology. The NCM_LS DLPDU is stored in the RT-queue and it is broadcast by the DLPM using the MA-DATA request primitive.

7.5.5.3.5 Receiving

The broadcast DL-address is used as the destination address in an NCM_LS DLPDU. Therefore, the NCM_LS DLPDU is directly forwarded by the MAC layer. The NCM_LS DLPDU is processed by the DLM as follows.

a) Extract the DPI (see 8.3.6.6) from the DLMDU and use it to check for a DL-address collision. If the source address is not registered in the DPI, generate an EVENT_IN_DEVICE event. If a local DL-address collision is detected, set the DEV_FLAG (see 8.3.6.4.2) to “Local DL-address Collision” and generate an EVENT_THIS_ADDR_COLLISION event. If a network DL-address collision is detected, set the NET_FLAG (see 8.3.6.5.10) to “Network DL-address Collision Status” and generate an EVENT_NET_ADDR_COLLISION event. Notification of any generated event is sent to the local DLMS-user using the DLM-EVENT indication service primitive.

b) Check the DLM state trigger event condition using the PORT_INFO and perform the DLM state transition (see 8.6.3.3).

c) If the DLM state is not LNM or RNM, enable the frame forward functions and set “frame forwarding function from R-port1 to R-port2” and “Frame Forwarding Function from R-port2 to R-port1” in the PORT_INFO (see 8.3.6.4.8) to TRUE.

7.5.5.4 NCM_RS DLPDU

7.5.5.4.1 General

The NCM_RS DLPDU is used to indicate that the network is automatically configured as a ring topology. This DLPDU is generated by the device that is selected as the RNMP on the network.

7.5.5.4.2 NCM_RS DLPDU structure

See 8.7.5.1.2

7.5.5.4.3 NCM_RS DLPDU parameters

Table 83 shows the NCM_RS DLPDU parameter list.

Table 83 – NCM_RS DLPDU parameters

Parameter	Data type	Description
DST_addr	UNSIGNED16	Broadcast address
SRC_addr	UNSIGNED16	Local DL-address, DL_ADDR (see 8.4.3.3.3)
NCMT	UNSIGNED8	NCM_RING_START (see 7.5.4.3.2.3)
DLMDU	-	Local device information (see 8.3.6.4)
Length	UNSIGNED16	Length of DLMDU
R-port	UNSIGNED8	R-port1, R-port2

7.5.5.4.4 Sending

When the state of the DLM is changed to RNMP, the DLM generates an NCM_RS DLPDU to inform every device on the network of the change in network topology. The NCM_RS DLPDU is stored in the RT-queue and broadcast by the DLPM using the MA-DATA request primitive.

7.5.5.4.5 Receiving

The broadcast DL-address is used as the destination address in an NCM_RS DLPDU. Therefore, the NCM_RS DLPDU is directly forwarded by the MAC layer. The NCM_RS DLPDU is processed by the DLM as follows.

a) Extract the DPI (see 8.3.6.6) from the DLMDU and use it to check for a DL-address collision. If the source address is not registered in the DPI, generate an EVENT_IN_DEVICE event. If a local DL-address collision is detected, set the DEV_FLAG (see 8.3.6.4.2) to “Local DL-address Collision” and generate an EVENT_THIS_ADDR_COLLISION event. If a network DL-address collision is detected, set the NET_FLAG (see 8.3.6.5.10) to “Network DL-address Collision Status” and generate an EVENT_NET_ADDR_COLLISION event. Notification of any generated event is sent to the local DLMS-user using the DLM-EVENT indication service primitive.

b) Check the DLM state trigger event condition using the PORT_INFO and perform the DLM state transition (see 8.6.3.3).

c) If the DLM state is not RNMP or RNMS, enable the frame forward functions and set “frame forwarding function from R-port1 to R-port2” and “Frame Forwarding Function from R-port2 to R-port1” in PORT_INFO (see 8.3.6.4.8) to TRUE.

d) When the device is designated as the RNMS by the RNMP, disable the frame forwarding function in the RNMP direction.

e) Enter the RNMS state if the frame forwarding control is successfully completed. Otherwise, remain in the GD state.

f) When the DLM enters the RNMS state, send NCM_ACK_RNMS to the RNMP.

7.5.5.5 NCM_AR_DLPDU

7.5.5.5.1 General

The NCM_AR DLPDU is used to indicate that the designated RNMS has been successfully assigned and is operational. This DLPDU is generated by the device that is selected as the RNMS and transferred to the RNMP.

7.5.5.5.2 NCM_AR DLPDU structure

See 8.5.5.1.2

7.5.5.5.3 NCM_AR DLPDU parameters

Table 84 shows the NCM_AR DLPDU parameter list.

Table 84 – NCM_AR DLPDU parameters

Parameter	Data type	Description
DST_addr	UNSIGNED16	DL-address of RNMP
SRC_addr	UNSIGNED16	Local DL-address DL_ADDR (see 8.4.3.3.3)
NCMT	UNSIGNED8	NCM_ACK_RNMS (see 7.5.4.3.2.3)
DLMDU	-	Local device information (see 8.3.6.4)
Length	UNSIGNED16	Length of DLMDU
R-port	UNSIGNED8	Destination R-port in the Path table

7.5.5.5.4 Sending

When the state of the DLM changes from GD to RNMS, the DLM generates an NCM_AR DLPDU to indicate that the designated RNMS is successfully assigned, and unicasts it to the RNMP.

7.5.5.5.5 Receiving

The NCM_AR DLPDU is unicast from the RNMS to the RNMP. The NCM_RS DLPDU is processed by the DLM in the RNMP device as follows.

a) Extract the DPI (see 8.3.6.6) from the DLMDU and use it to check for a DL-address collision. If the source address is not registered in the DPI, generate an EVENT_IN_DEVICE event. If a local DL-address collision is detected, set the DEV_FLAG (see 8.3.6.4.2) to “Local DL-address Collision” and generate an EVENT_THIS_ADDR_COLLISION event. If a network DL-address collision is detected, set the NET_FLAG (see 8.3.6.5.10) to “Network DL-address Collision Status” and generate an EVENT_NET_ADDR_COLLISION event. Notification of any generated event is sent to the local DLMS-user using the DLM-EVENT indication service primitive.

b) Disable the frame forwarding function in the RNMS direction.

7.6 DLE elements of procedure

7.6.1 Overall structure

The DLL is composed of the control elements of the DLPM, the dual MAC (DMAC), the dual physical interface (DPHY), and the DLL management Interface. The DLPM is the primary control element. It provides the functions for deterministic MAC by coordinating the DMAC and the NCMs for reliable and efficient support both of higher-level connectionless real-time

and non-real-time data transfer services. The DLL management interface provides DLL management functions. Figure 40 depicts the overall structure of the DLL.

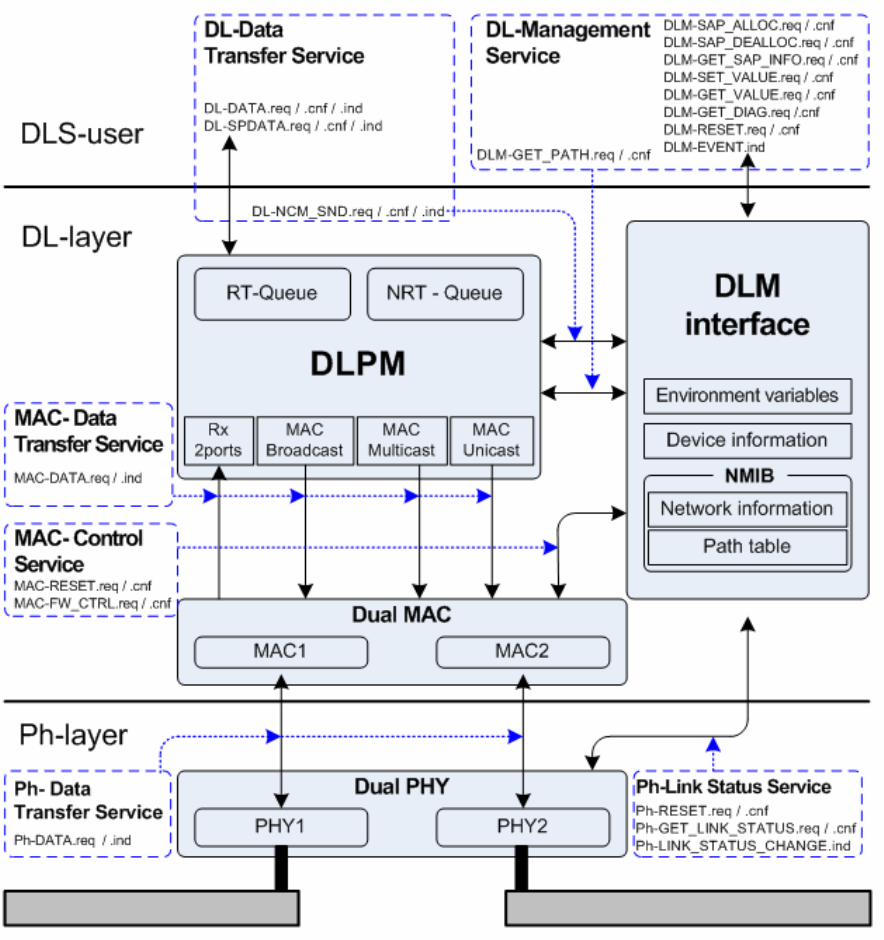


Figure 40 – DLL structure and elements

7.6.2 DL-protocol machine (DLPM)

7.6.2.1 Overview

The DLPM maintains two transmitter queues and one receiver queue. When a local DLS-user or DLMS-user generates a message, the message is stored in the RT-queue or the NRT-queue in the form of a DLPDU. The DLPM handles the messages in the RT-queue according to the DLPMSP. The DLPM transmits the message using the MA-DATA request primitive. When a message is received by the MAC layer, the received message is stored in the receiver queue and processed on a first-in first-out basis by the DLPM.

7.6.2.2 Primitive definitions

7.6.2.2.1 Primitives exchanged between DLPM and DLS-user

Table 85 shows the data link service primitives exchanged between the DLPM and the DLS-user.

Table 85 – Primitives exchanged between DLPM and DLS-user

Primitive	Source	Associated parameters	Description
DL-DATA.req	DLS-user	DST_addr DSAP	Transmit request to remote RAPIEnet device

		SSAP Priority DLSDU DLSDU length	
DL-DATA.cnf	DLPM	Status	Confirmation to calling DLS-user
DL-DATA.ind	DLPM	DST_addr SRC_addr SSAP DLSDU DLSDU length	Receive indication from a remote RAPIEnet device
DL-SPDATA.req	DLS-user	DLSDU DLSDU length	Sporadic data transmit request to remote RAPIEnet device
DL-SPDATA.cnf	DLPM	Status	Confirmation to calling DLS-user
DL-SPDATA.ind	DLPM	DLSDU DLSDU length	Sporadic data receive indication from a remote RAPIEnet device

Table 86 shows the parameters exchanged between the DLPM and the DLS-user.

Table 86 – Parameters exchanged between DLPM and DLS-user

Parameter	Description
DST_addr	Destination DL-address. C_BROADCAST_ADDR: broadcast address 0xFFFF User-defined multicast address: 0xFFFD–0xFFDE 0-MAX_DEVICE_ADDR: unicast address
DSAP	Destination service access point
SSAP	Source service access point
Priority	Message priority. Permissible values are C_PRI_0 through C_PRI_3 C_PRI_3 is the highest priority.
DLSDU	DLSDU
DLSDU length	Length of DLSDU
Status	This parameter allows the DLMS-user to determine whether the requested DLMS was provided successfully. If it failed, the reason is specified. The value of this parameter is one of “OK – success – the variable could be updated”; “Failure – the variable does not exist or could not assume the new value”; “Failure – invalid parameters in the request”

NOTE C_NCM_ADDR is not used for DL-DATA and DL-SPDATA services.

7.6.2.3 Primitives exchanged between DLPM and DLM

Table 87 and Table 88 shows the data link service primitives and parameters exchanged between the DLPM and the DLM.

Table 87 – Primitives exchanged between DLPM and DLM

Primitive	Source	Associated parameters	Description
DLM-GET_PATH.req	DLPM	DST_addr	Path information request to DLM

DLM-GET_PATH.cnf	DLM	Status R-port MAC address	Confirmation to calling DLPM with R-port and MAC address
DLM-NCM_SND.req	DLM	DST_addr NCMT DLMDU Length R-port	Transmit request for network control message NCM_LINK_ACTV NCM_ADV_THIS NCM_LINE_START NCM_RING_START
DLM-NCM_SND.cnf	DLPM	Status	Confirmation to calling service user
DLM-NCM_SND.ind	DLPM	DST_addr SRC_addr NCMT DLMDU length R-port	Network control message receive indication from a remote RAPIEnet device

Table 88 – Parameters used with primitives exchanged between DLPM and DLM

Parameter	Description
DST_addr	Destination DL-address. C_BROADCAST_ADDR: broadcast address, 0xFFFF C_NCM_ADDR: Fixed DL-address for network control message, 0xFFFE 0-MAX_DEVICE_ADDR: unicast address
SRC_addr	Source DL-address. “0–MAX_DEVICE_ADDR: unicast address”
R-port	R-port which is used to transmit a frame or to receive a frame.
MAC address	ISO/IEC 8802-3 MAC address that corresponds to the Destination DL-address.
NCMT	Network Control Message Type. NCM_LINK_ACTV (0x01), NCM_ADV_THIS (0x02), NCM_LINE_START (0x03), NCM_RING_START (0x04)
DLMDU	DL management data unit. The data field of DLM network control message.
Length	Length of DLMDU
Status	This parameter allows the DLMS-user to determine whether or not the requested DLMS was provided successfully. If it failed, the reason is specified. The value of this parameter can be one of “OK – success – the variable could be updated”; “Failure – the variable does not exist or could not assume the new value”; “Failure – invalid parameters in the request”

NOTE User-defined multicast addresses are not used for the destination DL-address.

7.6.2.3.1 DLPM state table

Figure 41 depicts the state transition diagram of the DLPM. The state table of the DLPM is shown in Table 89.

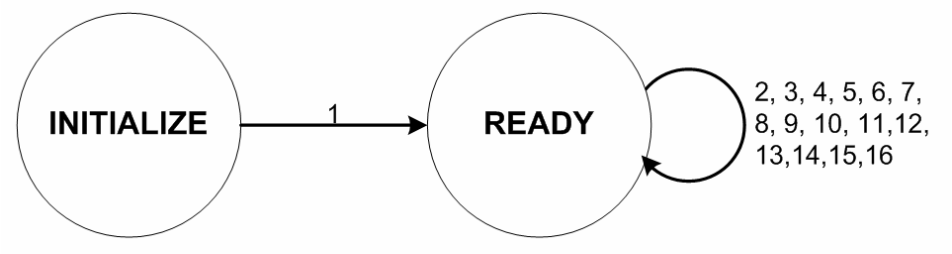


Figure 41 – State transition diagram of the DLPM

Table 89 – DLPM state table

#	Current	Event/Condition =>actions	Next state
1	INITIALIZE	POWER-ON or RESET / =>	READY
2	READY	DL-DATA.req {DST_addr, DSAP, SSAP, Priority, DLSDU, DLSDU length } / CHECK_PARA(DLSDU length) = “True” && DLM-GET_PATH.req {DST_addr} Status := DLM-GET_PATH.cfm { D_MAC_addr, R-port } Status =“Success” && CHECK_RTQUEUE( ) <> “Full” => DLPDU := BUILD_DLPDU(D_MAC_addr, DST_addr, DSAP, SSAP, Priority, DLSDU, DLSDU length) ENQUEUE_RT(Priority, DLPDU, DLPDU length, R-port) DL-DATA.cfm{Status := “Success”}	READY
3	READY	DL-DATA.req {DST_addr, DSAP, SSAP, Priority, DLSDU, DLSDU length } / CHECK_PARA(DLSDU length) <> “ True” => DL-DATA.cfm{Status := “Failure – Invalid parameter”}	READY
4	READY	DL-DATA.req {DST_addr, DSAP, SSAP, Priority, DLSDU, DLSDU length } / CHECK_PARA(DLSDU length) = “ True” && DLM-GET_PATH.ind { D_MAC_addr, R-port} =“Failure” => DL-DATA.cfm{Status := “Failure – not available destination”}	READY

#	Current	Event/Condition =>actions	Next state
5	READY	DL-DATA.req { DST_addr, DSAP, SSAP, Priority, DLSDU, DLSDU length } / CHECK_PARA(DLSDU length) = “ True” && DLM-GET_PATH.ind { D_MAC_addr, R-port } =“ Success” && CHECK_RTQUEUE( ) = “Full” => DL-DATA.cfm{Status := “Failure – The RT-queue is full”}	READY
6	READY	DL-SPDATA.req {DLSDU, DLSDU length} / CHECK_PARA(DLSDU length) = “True” && CHECK_NRTQUEUE() <> “Full” => DLPDU := DLSDU R-port := R-port1 && R-port2 ENQUEUE_NRT(DLPDU, DLPDU length, R-port) DL-SPDATA.cfm {Status := “Success”}	READY
7	READY	DL-SPDATA.req {DLSDU, DLSDU length} / CHECK_PARA(DLSDU length) = “False” => DL- SPDATA.cfm{Status := “Failure – Invalid requested parameter”}	READY
8	READY	DL-SPDATA.req {DLSDU, DLSDU length} / CHECK_PARA(DLSDU length) = “True” && CHECK_NRTQUEUE( ) = “Full” => DL- SPDATA.cfm{Status := “Failure – The NRT-queue is full”}	READY
9	READY	DL-NCM_SND.req {DST_addr, NCMT, DLMDU, Length, R-port} / CHECK_PARA(Length) = “True” && CHECK_RTQUEUE( ) <> “Full” => Priority := C_HIGHEST_PRIORITY DLPDU := BUILD_DLPDU_NCM(DST_addr , Priority, NCMT , DLMDU, Length) QUEUE_RT(Priority, DLPDU, DLPDU length, R-port) DL-NCM_SND.cfm{Status := “Success”}	READY
10	READY	DL-NCM_SND.req {DST_addr, NCMT, DLMDU, Length, R-port} / CHECK_PARA(Length) = “False” => DL- NCM_SND.cfm{Status := “Failure – Invalid parameter”}	READY

#	Current	Event/Condition =>actions	Next state
11	READY	DL-NCM_SND.req {DST_addr, NCMT, DLMDU, Length, R-port} / CHECK_PARA(Length) = “True” && CHECK_RTQUEUE( ) = “Full” => DL- NCM_SND.cfm{Status := “Failure – The RT-queue is full”}	READY
12	READY	MA-DATA.ind{DLPDU, DLPDU length, R-port} / CHECK_FRAME_ETH_TYPE(DLPDU) = “RAPIEnet” && CHECK_FRAME_FOR_THIS(DLPDU) = “True” && CHECK_FC(DLPDU) <> “Network control message” => DST_addr := GET_D_ADDR(DLPDU) SRC_addr := GET_S_ADDR(DLPDU) DSAP := GET_DSAP(DLPDU) SSAP := GET_SSAP(DLPDU) DLSDU := GET_DLSDU(DLPDU) DLSDU length := GET_DLSDU(DLPDU length) DLS-user := GET_DLS_USER(DSAP) dst_dls_user := SET_DLS_USER(DLS-user) dst_dls_user.DL-DATA.ind{ DST_addr, SRC_addr, SSAP, DLSDU, DLSDU length}	READY
13	READY	MA-DATA.ind{DLPDU, DLPDU length, R-port} / CHECK_FRAME_ETH_TYPE(DLPDU) <> “RAPIEnet” => DLSDU := DLPDU DLSDU length := DLPDU length DL-SPDATA.ind{DLSDU, DLSDU length}	READY
14	READY	MA-DATA.ind{DLPDU, DLPDU length, R-port} / CHECK_FRAME_ETH_TYPE(DLPDU) = “RAPIEnet” && CHECK_FRAME_FOR_THIS(DLPDU) = “True” && CHECK_FC(DLPDU) = “Network Control Message” => DST_addr := GET_D_ADDR(DLPDU) SRC_addr := GET_S_ADDR(DLPDU) DLMDU := GET_DLSDU(DLPDU) length := GET_DLSDU(DLPDU length) cmd := GET_NCM_CMD(DLPDU) DL-NCM_SND.ind{DST_addr, SRC_addr, NCMT, DLMDU, DLMDU length, R-port}	READY

#	Current	Event/Condition =>actions	Next state
15	READY	New DLPDU from DLS-user / CHECK_RTQUEUE( ) <> “Empty” => DLPDU := DQUEUE_RT(policy) MA-DATA.req {DLPDU, DLPDU length, R-port}	READY
16	READY	New DLPDU from DLS-user / CHECK_RTQUEUE( ) = “Empty” && / CHECK_NRTQUEUE( ) <> “Empty” => DLPDU := DQUEUE_NRT( ) MA-DATA.req {DLPDU, DLPDU length, R-port}	READY

7.6.2.4 DLPM functions

All functions of the DLPM are summarized in Table 90.

Table 90 – DLPM functions table

Function name	Input	Output	Operation
CHECK_PARA	DLSDU length	True/False	Return FALSE if the DLSDU exceeds the C_MAX_USER_DLMDU_SIZE. Otherwise, return TRUE
CHECK_RTQUEUE	(none)	status	Check the RT-queue condition. The returned status is “Full,” “Empty” or “Queued”
BUILD_DLPDU	D_MAC_addr DST_addr DSAP SSAP Priority DLSDU DLSDU length	DLPDU DLPDU length	Build DLPDU and return the DLPDU and its length
ENQUEUE_RT	Priority DLPDU DLPDU length R-port	(none)	Queues the input data into the tail of the RT-queue
CHECK_NRTQUEUE	(none)	status	Check that the NRT-Queue condition for DATA is fully queued. The returned status is “Full,” “Empty” or “Queued”
ENQUEUE_NRT	DLPDU DLPDU length R-port	(none)	Queues the input data into the NRT-QUEUE on a FIFO basis

Function name	Input	Output	Operation
BUILD_DLPDU_NCM	DST_addr Priority NCMT DLMDU DLMDU length	DLPDU DLPDU length	Build NCM DLPDU and return the DLPDU and its length
DQUEUE_RT	Policy	DLPDU DLPDU length R-port	Get a DLPDU from the RT-queue as specified by the DLPMSP
DQUEUE_NRT	(none)	DLPDU DLPDU length R-port	Get a DLPDU from the NRT-queue according to the FIFO method
CHECK_FRAME_ETH_TYPE	DLPDU	status	Check RAPIEnet EtherType (0x88FE) Return “RAPIEnet” when the type is correct. Otherwise, return “Other”
CHECK_FRAME_FOR_THIS	DLPDU	True/False	Check if the frame has been unicast to local device Return TRUE if the Destination DL-address is equal to the Local DL-address. Otherwise, return FALSE
CHECK_FC	DLPDU	status	Check if the received frame is a network control message Return “Network control message” if the FC field indicates NCM. Otherwise, return “Other”
GET_D_ADDR	DLPDU	DST_addr	Return the Destination DL-address of the DLPDU
GET_S_ADDR	DLPDU	SRC_addr	Return the Source DL-address of the DLPDU
GET_DSAP	DLPDU	DSAP	Return the Destination SAP of the DLPDU
GET_SSAP	DLPDU	SSAP	Return the Source SAP of the DLPDU
GET_DLSDU	DLPDU	DLSDU	Return DLSDU of the DLPDU
GET_DLSDU LENGTH	DLPDU length	DLSDU length	Return the DLSDU length
GET_DLS_USER	DSAP	DLS-user ID	Return the DLS-user ID that owns the DSAP
SET_DLS_USER	DLS-user ID	DLS-user object	Return the DLS-user object using the DLS-user ID
GET_NCMT	DLPDU	NCMT	Return the NCMT of the NCM DLPDU

7.6.3 DLL management protocol

7.6.3.1 Overview

This clause describes the interface protocol between the DLM and the DLMS-user. This description of the DLL management protocol provides the DLL management services specified in Clause 7 by making use of the services available to the DLMS-user. Full implementation details and matters of local responsibility are not included in this clause.

7.6.3.2 Primitive definitions

7.6.3.2.1 Primitive exchanged between DLM and DLS-user

Table 91 summarizes all primitives exchanged between the DLM and the DLS-user.

Table 91 – Primitives exchanged between DLM and DLS-user

Primitive	source	Associated parameters	Description
DLM-RESET.req	DLS-user	<none>	This request primitive causes the DLM to reset the DLE
DLM-RESET.cnf	DLM	DLM-status	This indicates the status of the reset
DLM-SET_VALUE.req	DLS-user	Variable name Desired value	This service is used to assign new values to the DLE variables
DLM-SET_VALUE.cnf	DLM	Status	The DLMS-user receives confirmation that the specified variables have been set to the new values
DLM-GET_VALUE.req	DLS-user	Variable name	This service is used to read the value of a DLE variable
DLM-GET_VALUE.cnf	DLM	Status Current value	This service returns the actual value of the specified variable
DLM-SAP_ALLOC.req	DLS-user	SAP DLS-user ID	This service is used by the DLMS-user to obtain a SAP assignment from the DLM
DLM-SAP_ALLOC.cnf	DLM	Status	This service returns the result status of DLM-SAP_ALLOC.req
DLM-SAP_DEALLOC.req	DLS-user	SAP	This service is used by the DLMS-user to release and return the allocated SAP to the DLM
DLM-SAP_DEALLOC.cnf	DLM	Status	This service returns the result status of DLM-SAP_DEALLOC.req
DLM-GET_SAP_INFO.req	DLS-user	SAP	This service is used by the DLMS-user to obtain the information of already allocated SAP from the DLM
DLM-GET_SAP_INFO.cnf	DLM	Status DLS-user ID	This service returns the result status of DLM-GET_SAP_INFO.req, specifically the result status and DLS-user ID
DLM-GET_DIAG.req	DLS-user	Diag-type Addr	This service is used by the DLMS-user to obtain the diagnostic information from the DLM
DLM-GET_DIAG.cnf	DLM	Status Diag	This service returns the result status of DLM-GET_SAP_INFO.req, specifically the result status and diagnostic information
DLM-EVENT.ind	DLM	Event	This service is used to inform the DLMS-user about certain events or errors in the DLL

The parameters used with the primitives exchanged between the DLM and the DLS-user are described in Table 92.

Table 92 – Parameters used with primitives exchanged between DLM and DLS-user

Parameter	Description
DLM-status	This parameter allows the DLMS-user to determine whether or not the requested DLMS was provided successfully. It it failed, the reason is specified. The value of this parameter can be one of “OK – successfully completed”; “Failure – terminated before completion”

Variable name	This parameter specifies the DLE variable whose value is to be set
Desired value	This parameter specifies the desired value for the selected variable
Status	This parameter allows the DLMS-user to determine whether or not the requested DLMS was provided successfully. If it failed, the reason is specified. The value of this parameter can be one of "OK – success – the variable could be updated"; "Failure – the variable does not exist or could not assume the new value"; "Failure – invalid parameters in the request"
Current value	This parameter indicates the current value of the designated variable
DLS-user ID	This parameter indicates the numeric identification of the local DLS-user. DLS-user ID is unique in a device
SAP	This parameter indicates the DLSAP
Diag-type	This parameter indicates the type of diagnostic information "DIAG_TYPE_L_DEVICE_INFO – local device information" "DIAG_TYPE_R_DEVICE_INFO – remote device information" "DIAG_TYPE_NET_INFO – network information"
Addr	This parameter indicates the DL-address of the designated node
Event	This parameter specifies the primitive or composite event being announced. The possible values are defined in the corresponding part of 8.6.3.3

7.6.3.2.2 Primitives exchanged between DLM and DLPM

Table 87 summarizes all primitives exchanged between the DLM and the DLPM.

7.6.3.2.3 Primitives exchanged between DLM and DMAC

Table 93 and Table 94 summarize all primitives and parameters exchanged between the DLM and the DMAC.

Table 93 – Primitive exchanged between DLM and DMAC

Primitive name	source	Associated parameters	Description
MAC-RESET.req	DLM	<none>	Reset MAC layer
MAC-FW_CTRL.req	DLM	R-port F_en	This service is used to control the frame forward functions in the MAC layer
MAC-FW_CTRL.cnf	D-MAC	Status	This service returns the result status of MAC-FW_CTRL.req

Table 94 – Parameters used with primitives exchanged between DLM and DMAC

Parameter name	Description
R-port	This parameter indicates the RAPIEnet Ethernet port. Every RAPIEnet device has two R-ports: R-port1 and R-port2.
F_en	This parameter indicates the frame forward control status between the two R-ports: "Enable - enable the hardware-based frame forward function" "Disable - disable the hardware-based frame forward function"
Status	This parameter allows the DLMS-user to determine whether or not the requested DLMS was provided successfully. If it failed, the reason is specified. The value of this parameter can be one of "OK – success – the variable could be updated"; "Failure – the variable does not exist or could not assume the new value"; "Failure – invalid parameters in the request"

7.6.3.2.4 Primitive exchanged between DLM and DPHY

Table 95 and Table 96 summarize all primitives and parameters exchanged between the DLM and the DPHY.

Table 95 – Primitive exchanged between DLM and DPHY

Primitive name	source	Associated Parameters	Description
Ph-RESET.req	DLM	<none>	Reset physical layer
Ph-GET_LINK_STATUS.req	DLM	R-port	This service is used to obtain link status information from the physical layer
Ph-GET_LINK_STATUS.cnf	DPHY	R-port L_status	This service returns the result status of Ph-GET_LINK_STATUS.req
Ph-LINK_STATUS_CHANGE.ind	DPHY	R-port	This service is used to notify the DLM of the link status change event

Table 96 – Parameters used with primitives exchanged between DLM and DPHY

Parameter name	Description
R-port	This parameter indicates the RAPIEnet Ethernet port. Every RAPIEnet device has two R-ports: R-port1 and R-port2
L_status	"link active – The link is activated and available for communication" "link inactive – The link is not activated and not available for communication"

7.6.3.3 DLM state table

The DLM is maintained with 6 states: INITIALIZE, SA, LNM, GD, RNMP, and RNMS. The DLM controls the frame forward functions in the MAC layer according to its DLM state. DLM state transition diagram and descriptions are shown in Figure 42 and Table 97.

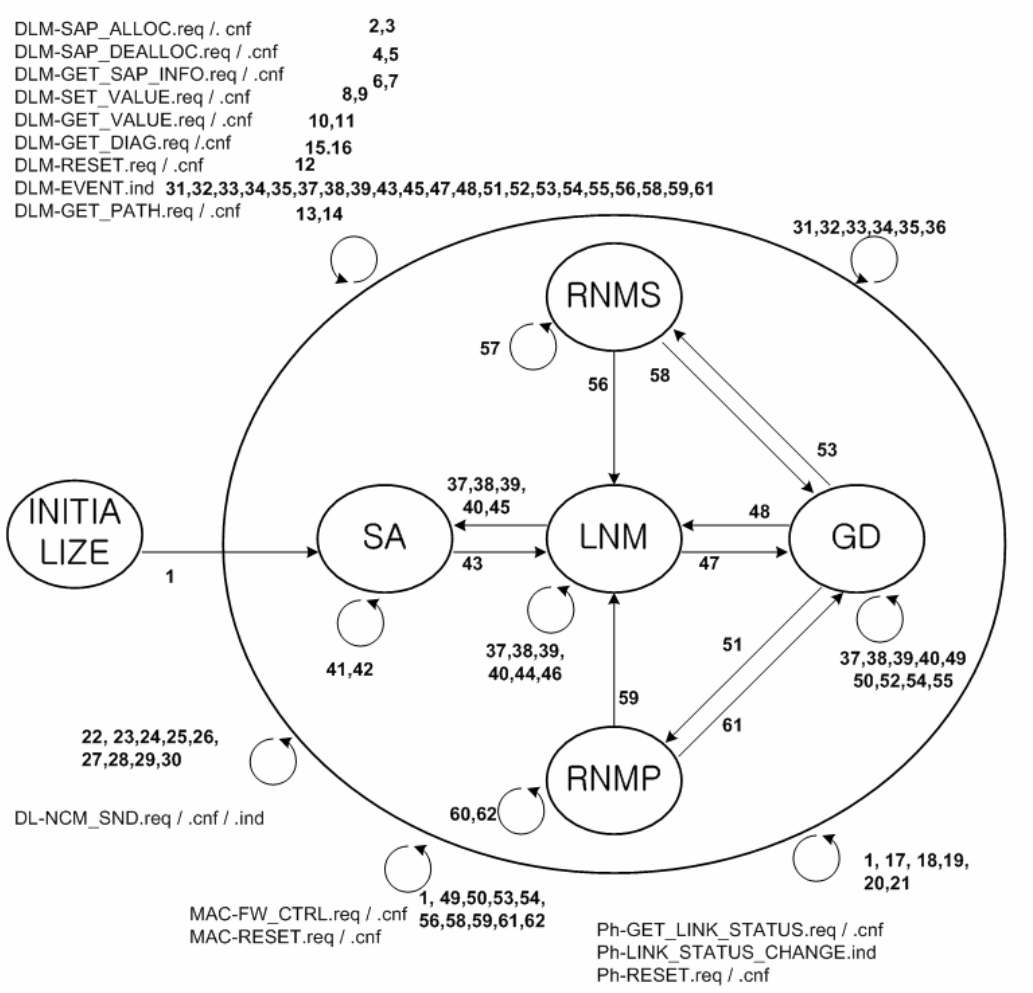


Figure 42 – State transition diagram of DLM

SA (stand-alone)

This state means that the local initialization procedures have been successfully completed and the device is ready to be linked to the other devices. In the SA state, the DLM tries to find the other devices on the network. When a link is established, the state changes to the LNM state.

LNM (line network manager)

This state means that the local device is located at the end of a line network. The LNM device is linked on the line network with one of its two R-ports. Both frame forward functions are disabled in the LNM device.

GD (general device)

This state means that the local device is connected to the network through both its R-ports. Both frame forward functions are enabled in the GD device. However, the frame forward

functions in the GD device are suspended until the device receives NCM_LS DLPDU or NCM_RS DLPDU.

RNMP (primary ring network manager)

This state means that the local device is automatically selected as the primary ring network manager in a ring network. The RNMP device selects one of its neighbouring devices as the secondary ring network manager (RNMS) using the NCM_RS DLPDU. The RNMP device disables the frame forward function in the RNMS direction but keeps the frame forward function in the other direction enabled.

RNMS (secondary ring network manager)

This state means that the local device is selected as the secondary ring network manager in a ring network. The RNMS device disables the frame forward function in the RNMP direction but keeps the frame forward function in the other direction enabled.

Table 97 – DLM state table

#	Current	Event/ Condition =>actions	Next state
1	INITIALIZE	POWER-ON or RESET / => INIT_ENV_VAR( ) INIT_SAP_INFO( ) INIT_DEV_INFO( ) INIT_NET_INFO( ) INIT_PATH_INFO( ) MAC-RESET.req{ } Ph-RESET.req{ }	SA
		Events triggered by the DL-Management service are listed below.	
2	Any state	DLM-SAP_ALLOC.req {SAP, DLS-user ID} / CHECK_ALLOC_SAP(SAP) = “True” => ALLOC_SAP(SAP, DLS-user ID) Status := “success” DLM-SAP_ALLOC.cnf {Status}	Any state
3	Any state	DLM-SAP_ALLOC.req { SAP, DLS-user ID} / CHECK_ALLOC_SAP(SAP) <> “True” => Status := “Failure – SAP is already allocated to another DLS-user” DLM-SAP_ALLOC.cnf {Status}	Any state

#	Current	Event/ Condition =>actions	Next state
4	Any state	DLM-SAP_DEALLOC.req { SAP } / CHECK_DEALLOC_SAP(SAP) = “True” => DEALLOC_SAP(SAP) Status := “success” DLM-SAP_DEALLOC.cnf {Status}	Any state
5	Any state	DLM-SAP_DEALLOC.req { SAP } / CHECK_DEALLOC_SAP(SAP) <> “True” => Status := “Failure – SAP is not allocated to a DLS-user” DLM-SAP_DEALLOC.cnf {Status}	Any state
6	Any state	DLM-GET_SAP_INFO.req { SAP } / CHECK_ALLOCEDSAP(SAP) = “True” => DLS-user ID := GET_USERID_FOR_SAP(SAP) Status := “Success” DLM-GET_SAP_INFO.cnf { Status, DLS-user ID}	Any state
7	Any state	DLM-GET_SAP_INFO.req { SAP } / CHECK_ALLOCEDSAP(SAP) <> “True” => DLS-user ID := INVALID_USER_ID Status := “Failure – SAP is not allocated to a DLS-user” DLM-GET_SAP_INFO.cnf { Status, DLS-user ID}	Any state
8	Any state	DLM-SET_VALUE.req { variable name, desired value} / CHECK_VALUE(variable name, desired value) = “valid” => SET_VALUE(variable name, desired value) Status := “success” DLM-SET_VALUE.cnf { Status }	Any state
9	Any state	DLM-SET_VALUE.req { variable name, desired value} / CHECK_VALUE(variable name, desired value) <> “valid” => Status := “Failure – invalid parameters in the request” DLM-SET_VALUE.cnf { Status }	Any state

#	Current	Event/ Condition =>actions	Next state
10	Any state	DLM-GET_VALUE.req {variable name} / CHECK_VAR(variable name) = “valid” => Current value := GET_CURRENT_VAL(variable name) Status := “success” DLM-GET_VALUE.cnf { Status , Current value }	Any state
11	Any state	DLM-GET_VALUE.req {variable name} / CHECK_VAR(variable name) <> “valid” => Current value := INVALID_VALUE Status := “Failure – invalid parameters in the request” DLM-GET_VALUE.cnf { Status , Current value }	Any state
12	Any state	DLM-RESET.req { } / => INIT_ENV_VAR( ) INIT_SAP_INFO( ) INIT_DEV_INFO( ) INIT_NET_INFO( ) INIT_PATH_INFO( ) MAC-RESET.req{ } Ph-RESET.req{ } Status := “success” DLM-RESET.cnf { Status }	SA
13	Any state	DLM-GET_PATH.req {Addr} / CHECK_ADDR(Addr) = “valid” => R-port := GET_DST_PORT(addr) D_MAC_addr := GET_DST_MAC_ADDR(addr) Status := “success” DLM-GET_PATH.cnf { Status, R-port, D_MAC_Addr }	Any state

#	Current	Event/ Condition =>actions	Next state
14	Any state	DLM-GET_PATH.req {Addr} / CHECK_ADDR(Addr) <> “valid” => R-port := INVALID_R_PORT D_MAC_addr := INVLID_MAC_ADDR Status := “Failure – invalid parameters in the request” DLM-GET_PATH.cnf { Status, R-port, D_MAC_Addr }	Any state
15	Any state	DLM-GET_DIAG.req { Diag-type, addr} / CHECK_DIAG_TYPE (Diag-type, addr) = “True” => Status := “success” Diag-info := GET_DIAG_INFO(Diag-type) DLM-GET_DIAG.cnf {Status, Diag-info}	Any state
16	Any state	DLM-GET_DIAG.req { Diag-type, addr} / CHECK_DIAG_TYPE (Diag-type, addr) <> “True” => Status := “Failure – invalid parameters in the request” DLM-GET_DIAG.cnf {Status, Diag-info}	Any state
		Events triggered by the DLM event generator are listed below.	
17	Any state	Ph_LINK_STATUS_CHANGE.ind {R-port} / => Ph-GET_LINK_STATUS.req{R-port}	Any state
18	Any state	Ph-GET_LINK_STATUS.cnf {R-port, L_status } / L_status = “link active” && CHECK_NEWLY_LINK_ACTV(R-port) = “True” => UPDATE_PORT_INFO(R-port, L_status) NEWLY_LINK_ACTV(R-port)	Any state
19	Any state	Ph-GET_LINK_STATUS.cnf {R-port, L_status } / L_status = “link active” && CHECK_NEWLY_LINK_ACTV(R-port) <> “True” => (none)	Any state

#	Current	Event/ Condition =>actions	Next state
20	Any state	Ph-GET_LINK_STATUS.cnf {R-port, L_status } / L_status = “link inactive” && CHECK_NEWLY_LINK_INACTV(R-port) = “True” => UPDATE_PORT_INFO(R-port, L_status) DLM_trg_event := GET_DLM_STATE_TRG_EVENT( ) DLM_STATE_MACHINE(DLM_trg_event)	Any state
21	Any state	Ph-GET_LINK_STATUS.cnf {R-port, L_status } / L_status = “link inactive” && CHECK_NEWLY_LINK_INACTV(R-port) <> “True” => (none)	Any state
		Events triggered by the received frame are listed below.	

22	Any state	DL-NCM_SND.ind {
		DST_addr, SRC_addr, NCMT, DLMDU, Length, R-port }
		/ NCMT = NCM_LINK_ACTV &&
		CHECK_ECHO(DLMDU, Length) <> “True”
		=>
		Neighbor-type := “homogeneous”
		UPDATE_PORT_INFO(R-port, Neighbor-type)
		STORE_DEV_INFO(DLMDU, Length)
		Forward-R-port := INVERT_PORT(R-port)
		INC_HOP_CNT(DLMDU)
		DL-NCM_SND.req{
		DST_addr, NCMT, DLMDU, Length, Forward-R-port}
		DLM_trg_event := GET_DLM_STATE_TRG_EVENT( )
		DLM_STATE_MACHINE(DLM_trg_event)
		DST_addr := C_NCM_ADDR
		NCMT := NCM_ADV_THIS
		DLMDU = device information
		Length = size of DLMDU
		DL-NCM_SND.req{
		DST_addr, NCMT, DLMDU, Length, R-port }
			Any state

#	Current	Event/ Condition =>actions	Next state
23	Any state	DL-NCM_SND.ind { DST_addr, SRC_addr, NCMT, DLMDU, Length, R-port } / NCMT = NCM_LINK_ACTV && CHECK_ECHO(DLMDU, Length) = “True” CHECK_NET_TOPOLOGY( ) = NET_TPG_RING => DLM_trg_event := DLM_STATE_TRG_RING DLM_STATE_MACHINE(DLM_trg_event)	Any state
24	Any state	DL-NCM_SND.ind { DST_addr, SRC_addr, NCMT, DLMDU, Length, R-port } / NCMT = NCM_LINK_ACTV && CHECK_ECHO(DLMDU, Length) = “True” CHECK_NET_TOPOLOGY( ) <> NET_TPG_RING => DLM_trg_event := GET_DLM_STATE_TRG_EVENT( ) DLM_STATE_MACHINE(DLM_trg_event)	Any state

25	Any state	DL-NCM_SND.ind {
		DST_addr, SRC_addr, NCMT, DLMDU, Length, R-port }
		/ NCMT = NCM_ADV_THIS &&
		CHECK_ECHO(DLMDU, Length) <> “True”
		=>
		Neighbor-type := “homogeneous”
		UPDATE_PORT_INFO(R-port, Neighbor-type)
		STORE_DEV_INFO(DLMDU, Length)
		Forward-R-port := INVERT_PORT(R-port)
		INC_HOP_CNT(DLMDU)
		DL-NCM_SND.req{
		DST_addr, NCMT, DLMDU, Length, Forward-R-port}
		DLM_trg_event := GET_DLM_STATE_TRG_EVENT( )
		DLM_STATE_MACHINE(DLM_trg_event)
			Any state

26	Any state	DL-NCM_SND.ind { DST_addr, SRC_addr, NCMT, DLMDU, Length, R-port } / NCMT = NCM_ADV_THIS && CHECK_ECHO(DLMDU, Length) = “True” CHECK_NET_TOPOLOGY( ) = NET_TPG_RING =>	Any state

#	Current	Event/ Condition =>actions	Next state
		DLM_trg_event := DLM_STATE_TRG_RING DLM_STATE_MACHINE(DLM_trg_event)	
27	Any state	DL-NCM_SND.ind { DST_addr, SRC_addr, NCMT, DLMDU, Length, R-port } / NCMT = NCM_ADV_THIS && CHECK_ECHO(DLMDU, Length) = “True” CHECK_NET_TOPOLOGY( ) <> NET_TPG_RING => DLM_trg_event := GET_DLM_STATE_TRG_EVENT( ) DLM_STATE_MACHINE(DLM_trg_event)	Any state
28	Any state	DL-NCM_SND.ind { DST_addr, SRC_addr, NCMT, DLMDU, Length, R-port } / NCMT = DLM_ RING_START && CHECK_ECHO(DLMDU, Length) <> “True” => STORE_DEV_INFO(DLMDU, Length) DLM_trg_event := DLM_STATE_TRG_RING_STARTED DLM_STATE_MACHINE(DLM_trg_event)	Any state
29	Any state	DL-NCM_SND.ind { DST_addr, SRC_addr, NCMT, DLMDU, Length, R-port } / NCMT = DLM_ RING_START && CHECK_ECHO(DLMDU, Length) = “True” => (none)	Any state
30	Any state	DL-NCM_SND.ind { DST_addr, SRC_addr, NCMT, DLMDU, Length, R-port } / NCMT = DLM_ LINE_START => STORE_DEV_INFO(DLMDU, Length) DLM_trg_event := DLM_STATE_TRG_LINE_STARTED DLM_STATE_MACHINE(DLM_trg_event)	Any state
		Device information-related procedures are listed below.	

#	Current	Event/ Condition =>actions	Next state
31	Any state	STORE_DEV_INFO(DLMDU, length, R-port) / CHECK_NET_ADDR_COLLISION(DLMDU) <> “True” && CHECK_THIS_ADDR_COLLISION(DLMDU) = “True” && CHECK_NEWLY_IN_DEVICE(DLMDU) = “True” => events := EVENT_THIS_ADDR_COLLISION | EVENT_IN_DEVICE DLM-EVENT.ind(events) SAVE_DEV_INFO(DLMDU, length, R-port)	Any state
32	Any state	STORE_DEV_INFO(DLMDU, length, R-port) / CHECK_ NET_ADDR_COLLISION(DLMDU) = “True” CHECK_THIS_ADDR_COLLISION(DLMDU) <> “True” CHECK_NEWLY_IN_DEVICE(DLMDU) = “True” => events := EVENT_NET_ADDR_COLLISION | EVENT_IN_DEVICE DLM-EVENT.ind(events) SAVE_DEV_INFO(DLMDU, length, R-port)	Any state
33	Any state	STORE_DEV_INFO(DLMDU, length, R-port) / CHECK_ NET_ADDR_COLLISION(DLMDU) = “True” CHECK_THIS_ADDR_COLLISION(DLMDU) <> “True” CHECK_NEWLY_IN_DEVICE(DLMDU) <> “True” => events := EVENT_NET_ADDR_COLLISION DLM-EVENT.ind(events) SAVE_DEV_INFO(DLMDU, length, R-port)	Any state
34	Any state	STORE_DEV_INFO(DLMDU, length, R-port) / CHECK_ NET_ADDR_COLLISION(DLMDU) <> “True” CHECK_THIS_ADDR_COLLISION(DLMDU) = “True” CHECK_NEWLY_IN_DEVICE(DLMDU) <> “True” => events := EVENT_THIS_ADDR_COLLISION DLM-EVENT.ind(events) SAVE_DEV_INFO(DLMDU, length, R-port)	Any state
35	Any state	STORE_DEV_INFO(DLMDU, length, R-port) / CHECK_ NET_ADDR_COLLISION(DLMDU) <> “True” CHECK_THIS_ADDR_COLLISION(DLMDU) <> “True” CHECK_NEWLY_IN_DEVICE(DLMDU) = “True”	Any state

#	Current	Event/ Condition =>actions	Next state
		=> events := EVENT_IN_DEVICE DLM-EVENT.ind(events) SAVE_DEV_INFO(DLMDU, length, R-port)	
36	Any state	STORE_DEV_INFO(DLMDU, length, R-port) / CHECK_NET_ADDR_COLLISION(DLMDU) <> “True” CHECK_THIS_ADDR_COLLISION(DLMDU) <> “True” CHECK_NEWLY_IN_DEVICE(DLMDU) <> “True” => SAVE_DEV_INFO(DLMDU, length, R-port)	Any state
		Path table update procedures are listed below	
37	Any state	DELETE_DEV_INFO(Device UID) / CHECK_UID(UID) = “valid” CHECK_THIS_ADDR_COLLISION_CLEAR(UID) <> “True” CHECK_NET_ADDR_COLLISION_CLEAR(UID) <> “True” => DEL_DEV(UID) Events := EVENT_OUT_DEVICE DLM-EVENT.ind(Events)	Any state
38	Any state	DELETE_DEV_INFO(Device UID) / CHECK_UID(UID) = “valid” CHECK_THIS_ADDR_COLLISION_CLEAR(UID) = “True” CHECK_NET_ADDR_COLLISION_CLEAR(UID) <> “True” => DEL_DEV(UID) Events := EVENT_OUT_DEVICE | EVENT_THIS_ADDR_COLLISION_CLEAR DLM-EVENT.ind(Events)	Any state
39	Any state	DELETE_DEV_INFO(Device UID) / CHECK_UID(UID) = “valid” CHECK_THIS_ADDR_COLLISION_CLEAR(UID) <> “True” CHECK_NET_ADDR_COLLISION_CLEAR(UID) = “True” => DEL_DEV(UID) Events := EVENT_OUT_DEVICE | EVENT_NET_ADDR_COLLISION_CLEAR DLM-EVENT.ind(Events)	Any state

#	Current	Event/ Condition =>actions	Next state
40	Any state	DELETE_DEV_INFO(Device UID) / CHECK_UID(UID) <> “valid” => (<none>)	Any state
		State transitions in the DLM state machine are listed below.	
41	SA	NEWLY_LINK_ACTV(R-port) / => DST_addr := C_NCM_ADDR NCMT := DLM_LINK_ACTV DLMDU := local device information Length := size of local device information DL-NCM_SND.req(DST_addr, NCMT, DLMDU, Length, R-port)	SA
42	SA	DLM_STATE_MACHINE(DLM_trg_event) / DLM_trg_event <> DLM_STATE_TRG_P1 => (<none>)	SA
43	SA	DLM_STATE_MACHINE(DLM_trg_event) / DLM_trg_event = DLM_STATE_TRG_P1 => CHANGE_TOPOLOGY(NET_TPG_LINE) CHAGNE_DLM_STATE(LNM) Events := EVENT_NET_TPG_CHG | EVENT_DEV_STATE_CHG DLM-EVENT.ind (events)	LNM
44	LNM	NEWLY_LINK_ACTV(R-port) / => DST_addr := C_NCM_ADDR NCMT := DLM_LINK_ACTV DLMDU := local device information Length := size of local device information DL-NCM_SND.req(DST_addr, NCMT, DLMDU, Length, R-port)	LNM

#	Current	Event/ Condition =>actions	Next state
45	LNM	DLM_STATE_MACHINE(DLM_trg_event) / DLM_trg_event = DLM_STATE_TRG_P0 => CHANGE_TOPOLOGY(NET_TPG_SA) CHAGNE_DLM_STATE(SA) Events := EVENT_NET_TPG_CHG | EVENT_DEV_STATE_CHG DLM-EVENT.ind (events) UPDATE_PATH_TABLE(SA)	SA
46	LNM	DLM_STATE_MACHINE(DLM_trg_event) / DLM_trg_event = DLM_STATE_TRG_P1 => (<none>)	LNM
47	LNM	DLM_STATE_MACHINE(DLM_trg_event) / DLM_trg_event = DLM_STATE_TRG_P2 => CHAGNE_DLM_STATE(GD) Events := EVENT_DEV_STATE_CHG DLM-EVENT.ind (events)	GD
48	GD	DLM_STATE_MACHINE(DLM_trg_event) / DLM_trg_event = DLM_STATE_TRG_P1 => CHANGE_TOPOLOGY(NET_TPG_LINE) CHAGNE_DLM_STATE(LNM) Events := EVENT_NET_TPG_CHG | EVENT_DEV_STATE_CHG DLM-EVENT.ind (events) DST_addr := C_BROADCAST_ADDR NCMT := NCM_LINE_START DLMDU := local device information Length := size of local device information R-port := link active R-port DL-NCM_SND.req(DST_addr, NCMT, DLMDU, Length, R-port) UPDATE_PATH_TABLE(LINE)	LNM

#	Current	Event/ Condition =>actions	Next state
49	GD	DLM_STATE_MACHINE(DLM_trg_event) / DLM_trg_event = DLM_STATE_TRG_P2 && Network information.topology = LINE && CHECK_LNMS( ) = “valid” => R-port := R-port1 Forward_control :=FW_ENABLE FW_CTRL(R-port, Forward_control) R-port := R-PORT2 FW_CTRL(R-port, Forward_control)	GD
50	GD	DLM_STATE_MACHINE(DLM_trg_event) / DLM_trg_event = DLM_STATE_TRG_P2 && Network information.topology = LINE && CHECK_LNMS( ) <> “valid” => R-port := R-port1 Forward_control := FW_DISABLE FW_CTRL(Forward_control) R-port := R-PORT2 FW_CTRL(R-port, Forward_control)	GD
51	GD	DLM_STATE_MACHINE(DLM_trg_event) / DLM_trg_event = DLM_STATE_TRG_RING && CHECK_RNMP( ) = RNMP => CHANGE_TOPOLOGY(NET_TPG_RING) CHAGNE_DLM_STATE(RNMP) Events := EVENT_NET_TPG_CHG | EVENT_DEV_STATE_CHG DLM-EVENT.ind (events) DST_addr := C_BROADCAST_ADDR NCMT := NCM_RING_START DLMDU := local device information Length := size of local device information R-port := both R-port Rnms_UID := GET_RNMS_UID( ) SET_RNMPS_UID(Rnms_UID, DLMDU) DL-NCM_SND.req(DST_addr, NCMT, DLMDU, Length, R-port)	RNMP
52	GD	DLM_STATE_MACHINE(DLM_trg_event) / DLM_trg_event = DLM_STATE_TRG_RING && CHECK_RNMP( ) <> RNMP	GD

#	Current	Event/ Condition =>actions	Next state
		=> CHANGE_TOPOLOGY(NET_TPG_RING) Events := EVENT_NET_TPG_CHG DLM-EVENT.ind (events)	
53	GD	DLM_STATE_MACHINE(DLM_trg_event) / DLM_trg_event = DLM_STATE_TRG_RING_STARTED && GET_RNMS_UID( ) = device information.UID => Forward-control := FW_DISABLE R-port := GET_R_PORT_FOR(addr of RNMP) FW_CTRL(R-port, Forward-control) R-port := INVER_R_PORT(GET_R_PORT_FOR(addr of RNMP)) Forward-control := FW_ENABLE FW_CTRL(R-port, Forward-control) CHANGE_TOPOLOGY(NET_TPG_RING) CHAGNE_DLM_STATE(RNMS) Events := EVENT_NET_TPG_CHG | EVENT_DEV_STATE_CHG DLM-EVENT.ind (events) DST_addr := C_BROADCAST_ADDR NCMT := C_DLM_ACK_RNMS DLMDU := Local device information Length := size of local device information R-port := both R-port DL-NCM_SND.req(DST_addr, NCMT, DLMDU, Length, R-port)	RNMS

#	Current	Event/ Condition =>actions	Next state
54	GD	DLM_STATE_MACHINE(DLM_trg_event) / DLM_trg_event = DLM_STATE_TRG_RING_STARTED && GET_RNMS_UID( ) <> device information.UID => Forward_control :=FW_ENABLE R-port := R-PORT1 FW_CTRL(R-port , Forward_control) R-port := R-PORT2 FW_CTRL(R-port , Forward_control) CHANGE_TOPOLOGY(NET_TPG_RING) Events := EVENT_NET_TPG_CHG DLM-EVENT.ind (events) UPDATE_PATH_TABLE(RING)	GD
55	GD	DLM_STATE_MACHINE(DLM_trg_event) / DLM_trg_event = DLM_STATE_TRG_LINE_STARTED => CHANGE_TOPOLOGY(NET_TPG_LINE) Events := EVENT_NET_TPG_CHG DLM-EVENT.ind (events) UPDATE_PATH_TABLE(LINE)	GD

#	Current	Event/ Condition =>actions	Next state
56	RNMS	DLM_STATE_MACHINE(DLM_trg_event) / DLM_trg_event = DLM_STATE_TRG_P1 => Forward_control := FW_DISABLE R-port := R-PORT1 FW_CTRL(R-port, Forward_control) R-port := R-PORT2 FW_CTRL(R-port, Forward_control) CHANGE_TOPOLOGY(NET_TPG_LINE) CHAGNE_DLM_STATE(LNM) Events := EVENT_NET_TPG_CHG | EVENT_DEV_STATE_CHG DLM-EVENT.ind (events) DST_addr := C_BROADCAST_ADDR NCMT := NCM_LINE_START DLMDU := local device information Length := size of local device information R-port := link active R-port DL-NCM_SND.req(DST_addr, NCMT, DLMDU, Length, R-port) UPDATE_PATH_TABLE(LINE)	LNM
57	RNMS	DLM_STATE_MACHINE(DLM_trg_event) / DLM_trg_event = DLM_STATE_TRG_P2 => (<none>)	RNMS
58	RNMS	DLM_STATE_MACHINE(DLM_trg_event) / DLM_trg_event = DLM_STATE_TRG_LINE_STARTED => R-port := R-PORT1 Forward_control :=FW_ENABLE FW_CTRL(R-port, Forward_control) R-port := R-PORT2 FW_CTRL(R-port, Forward_control) CHANGE_TOPOLOGY(NET_TPG_LINE) CHAGNE_DLM_STATE(GD) Events := EVENT_NET_TPG_CHG | EVENT_DEV_STATE_CHG DLM-EVENT.ind (events) UPDATE_PATH_TABLE(LINE)	GD

#	Current	Event/ Condition =>actions	Next state
59	RNMP	DLM_STATE_MACHINE(DLM_trg_event) / DLM_trg_event = DLM_STATE_TRG_P1 => R-port := R-PORT1 Forward_control := FW_DISABLE FW_CTRL(R-port, Forward_control) R-port := R-PORT2 FW_CTRL(R-port, Forward_control) CHANGE_TOPOLOGY(NET_TPG_LINE) CHAGNE_DLM_STATE(LNM) Events := EVENT_NET_TPG_CHG | EVENT_DEV_STATE_CHG DLM-EVENT.ind (events) DST_addr := C_BROADCAST_ADDR NCMT := NCM_LINE_START DLMDU := local device information Length := size of local device information R-port := link active R-port DL-NCM_SND.req(DST_addr, NCMT, DLMDU, Length, R-port) UPDATE_PATH_TABLE(LINE)	LNM
60	RNMP	DLM_STATE_MACHINE(DLM_trg_event) / DLM_trg_event = DLM_STATE_TRG_P2 => (<none>)	RNMP
61	RNMP	DLM_STATE_MACHINE(DLM_trg_event) / DLM_trg_event = DLM_STATE_TRG_LINE_STARTED => R-port := R-PORT1 Forward_control := FW_ENABLE FW_CTRL(R-port, Forward_control) R-port := R-PORT2 FW_CTRL(R-port, Forward_control) CHANGE_TOPOLOGY(NET_TPG_LINE) CHAGNE_DLM_STATE(GD) Events := EVENT_NET_TPG_CHG | EVENT_DEV_STATE_CHG DLM-EVENT.ind (events) UPDATE_PATH_TABLE(LINE)	GD

#	Current	Event/ Condition =>actions	Next state
62	RNMP	DLM_STATE_MACHINE(DLM_trg_event) / DLM_trg_event = DLM_STATE_TRG_ACK_RNMS => Forward-control := FW_DISABLE R-port := GET_R_PORT_FOR(addr of RNMS) FW_CTRL(R-port, Forward-control) Forward-control := FW_ENABLE R-port := INVER_R_PORT(GET_R_PORT_FOR(addr of RNMS)) FW_CTRL(R-port, Forward-control)	RNMP

7.6.3.4 DLM functions

Table 98 shows the internal functions provided by the DLM.

Table 98 – DLM function table

Function name	Input	Output	Operation
INIT_CONF_PAR			Initialize DLL configuration parameters (see 8.3.6.1)
INIT_SAP_INFO			Initialize SAP information (see 8.3.6.3)
INIT_DEV_INFO			Initialize local device information (see 8.3.6.4)
INIT_NET_INFO			Initialize network information(see 8.3.6.5)
INIT_PATH_INFO			Initialize path table information (see 8.3.6.6)
CHECK_ALLOC_SAP	SAP	True/False	Return TRUE if the SAP is available Otherwise, return FALSE
ALLOC_SAP	SAP DLS-user ID		Register DLS-user ID in the SAP information
CHECK_DEALLOC_SAP	SAP	True/ False	Check if the SAP can be de-allocated
DEALLOC_SAP	SAP		Remove the DLS-user ID from the SAP information
CHECK_ALLOCEDSAP	SAP	True/ False	Check if the SAP is already allocated to a certain DLS-user
GET_USERID_FOR_SAP	SAP	DLS-user ID	Return the DLS-user ID that owns the allocated SAP
CHECK_VALUE	Variable name	Valid/Invalid	Check that the requested variables with desired value are valid
SET_VALUE	Variable name desired value		Set the value of the requested variable

Function name	Input	Output	Operation
GET_CURRENT_VAL	Variable name	Current_value	Get the value of the requested variable
CHECK_ADDR	DL-Address	Valid/Invalid	Check if the designated DL-address is valid. Return “Valid” if the unicast DL-address is registered in the path table. Return “Invalid” if the unicast DL-address is not registered in the path table Return “Valid” if the designated DL-address is not in the range of the unicast DL-address
GET_DST_PORT	DL-Address	R-port	Return the destination R-port for the designated device in the path table
GET_DST_MAC_ADDR	Address	MAC address	Return the ISO/IEC 8802-3 MAC address for the designated device in the path table
CHECK_DIAG_TYPE	Diagnostic_type Addr	True/False	Check if the diagnostic information type from the DLS-user is available
GET_DIAG_INFO	Diagnostic_type Addr	Diagnostic information	Return the diagnostic information according to the diagnostic information type
CHECK_NEWLY_LINK_ACTV	R-port	True/False	Check if the R-port has changed from “link Inactive” to “link Active” status
UPDATE_PORT_INFO	R-port status		Update the port information in the device information (see 8.3.6.4.8)
CHECK_NEWLY_LINK_INACTV	R-port	True/False	Check if the R-port has changed from “link Active” to link Inactive” status
GET_DLM_STATE_TRG_EVENT		Trigger Event	Get DLM state triggering event using the port information in the device information DLM_STATE_TRG_P0 DLM_STATE_TRG_P1 DLM_STATE_TRG_P2 DLM_STATE_TRG_RING DLM_STATE_TRG_RING_STARTED DLM_STATE_TRG_LINE DLM_STATE_TRG_LINE_STARTED
CHECK_ECHO	DLMDU, length	True/False	Check if the DLPDU was generated by the device itself
INC_HOP_CNT	DLMDU		Increment the Hop count field in the DLMDU
INVERT_PORT	R-port	R-port	Invert logical interface to the R-port. Return R-port2 if the input is R-port1 and return R-port1 if the input is R-port2
CHECK_NET_TOPOLOGY		Network topology	Check if the network is configured as a ring topology Return NET_TPG_RING if the broadcast network control message is received by the device itself. Otherwise, return NET_TPG_LINE
CHECK_NET_ADDR_COLLISION	DLMDU	True/False	Return TRUE if the source DL-address in the DLMDU is duplicated at one of the other devices on the network. Otherwise, return FALSE

Function name	Input	Output	Operation
CHECK_THIS_ADDR_COLLISION	DLMDU		Return TRUE if the source DL-address in the DLMDU is duplicated in the local device itself. Otherwise, return FALSE
CHECK_NEWLY_IN_DEVICE	DLMDU	True/False	Check if the source DL-address in the DLMDU is not registered in the Device information
SAVE_DEV_INFO	DLMDU length R-port		Store the device information in the DLMDU to the path table
DEL_DEV	Device UID		Remove the path table item corresponding to the Device UID
CHANGE_TOPOLOGY	Topology		Update the topology in the network information
CHAGNE_DLM_STATE	DLM-state		Update the DLS state in the device information
UPDATE_PATH_TABLE	Topology		Update and recalculate the destination port in the path table according to the network topology (see Annex A)
CHECK_LNMS		True/False	Check if two LNMs are listed in the path table
FW_CTRL	R-port Control		Enable or disable the frame forward function for the R-port
GET_RNMS_UID		UID	Return the device UID registered as RNMS in the path table. Return INVALID_UID if the RNMS is not found in the path table
GET_R_PORT_FOR	DL-address	R-port	Return the destination R-port to the DL-address from the path table. Return INVALID_ADDR if no table entry is found

7.7 Constants

7.7.1 General

This section describes the RAPIEnet DLL constants and their values.

7.7.2 Constants

Table 99 summarizes all DLL constants.

Table 99 – DLL constants

Constant	Value	Description
C_HIGHEST_PRIORITY	3	Highest message priority
INVALID_USER_ID	0	Invalid user identification
INVALID_R_PORT	0	Invalid R-port
NCM_LINK_ACTV	1	“link active” network control message
NCM_ADV_THIS	2	“advertise this information” network control message
NCM_ RING_START	3	“starting line topology network” network control message
NCM_LINE_START	4	“starting ring topology network” network control message
NCM_ACK_RNMS	5	ACK message from RNMS to RNMP
EVENT_NET_TPG_CHG	0x01	Event generated when the network topology is changed
EVENT_DEV_STATE_CHG	0x02	Event generated when the DLM state is changed
EVENT_THIS_ADDR_COLLISION	0x04	Event generated when the local DL-address collision is detected
EVENT_THIS_ADDR_COLLISION_CLEAR	0x08	Event generated when the local DL-address collision is cleared
EVENT_NET_ADDR_COLLISION	0x10	Event generated when the network DL-address collision is detected
EVENT_NET_ADDR_COLLISION_CLEAR	0x20	Event generated when the network DL-address collision is cleared
EVENT_IN_DEVICE	0x40	Event generated when a device first joins the network
EVENT_OUT_DEVICE	0x80	Event generated when a device disconnects from the network
DLM_STATE_TRG_P0	1	No RAPIEnet device is linked through the R-port
DLM_STATE_TRG_P1	2	One of two R-ports is linked to the RAPIEnet device
DLM_STATE_TRG_P2	3	Both R-ports are linked to RAPIEnet devices
DLM_STATE_TRG_RING	4	Ring network topology is automatically configured
DLM_STATE_TRG_RING_STARTED	5	RNMP is detected in the ring network
DLM_STATE_TRG_LINE	6	Line network is automatically configured
DLM_STATE_TRG_LINE_STARTED	7	New line network is configured
C_NCM_ADDR	0xFFFE	The DL-address for network control messages
C_BROADCAST_ADDR	0xFFFF	DL-address for broadcast
FW_ENABLE	1	Enable frame forwarding function
FW_DISABLE	0	Disable frame forwarding function
INVALID_UID	0	Invalid unique identification
INVALID_ADDR	0xFFDC	Invalid address

7.7.3 Data link layer error codes

Table 100 summarizes all DLL error codes.

Table 100 – RAPIEnet DLL error codes

Error	value	Description
ERR_FRAME_SIZE	0x1001	Frame size error
ERR_PROTOCOL_VER	0x1002	Protocol version error
ERR_NOT4ME_FRAME	0x1003	Received frame is not intended for the local device
ERR_SAP	0x1004	Service access point error
ERR_SERVICE_CMD	0x1005	Service command error
ERR_NO_DST_ADDR	0x1006	No device found matching the destination DL-address
ERR_PRM	0x1007	Invalid parameter
ERR_NG_DEV_INFO	0x1008	Device information error
ERR_GET_DEV_INFO	0x1009	Device information size error
ERR_Q_ALREADY_FULL	0x100A	Queue is full
ERR_SAP_ALREADY_REGED	0x100B	The SAP is already allocated to a DLS-user.
ERR_SAP_NOT_REGED	0x100C	The SAP is not allocated to a DLS-user
ERR_NO_MATCHED_ITEM	0x100D	No item is found in the NMIB
ERR_INVALID_DIAG_TYPE	0x100E	Diagnostic information type is invalid
ERR_DEV_CNT_EXIED	0x2001	DL-address is bigger than MAX_ADDR
NOT_SUPPORTEDSAP	0x2002	Invalid SAP address error
INVALID_DEV_INFO	0x2003	Invalid device information error

8 Application layer service definition

8.1 Introduction

This clause defines the RAPIEnet application layer service. This clause is to be one of the real-time Ethernet (RTE) specifications to facilitate the interconnection of automation system components. RAPIEnet application services provide reliable and transparent data communication among RAPIEnet application users through the logical interface between the RAPIEnet data link layer and the application layer.

The application service is provided by the application protocol making use of the services available from the application or other lower layers. This clause defines the application layer service characteristics that higher layer protocols may exploit. The application layer protocol defines some procedures to share and maintain the application information in an automation system.

8.2 Scope

8.2.1 Overview

The fieldbus application layer (FAL) provides user programmes with a means to access the fieldbus communication environment. In this respect, the FAL can be considered a window between corresponding application programmes.

This specification provides the common elements for basic time-critical and non-time-critical messaging communications between application programs in an automation environment as well as material specific to the RAPIEnet protocol. The term “time-critical” is used to represent the presence of a time-window within which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the time window risks failure of the applications requesting the actions, with attendant risk to equipment, plant and, possibly, human life.

This specification defines, in an abstract way, the externally visible service provided by the FAL in terms of

a) an abstract model for defining application resources (objects) capable of being manipulated by users via the FAL service;
b) the primitive actions and events of the service;
c) the parameters associated with each primitive action and event and the form that they take;
d) the interrelationship between these actions and events and their valid sequences.

The purpose of this specification is to define the services provided to

a) the FAL-user at the boundary between the user and the application layer of the fieldbus Reference Model;
b) systems management at the boundary between the application layer and systems management of the fieldbus Reference Model.

This specification describes the structure and services of the IEC FAL, in conformance with the OSI Basic Reference Model (ISO/IEC 7498) and the OSI Application layer Structure (ISO/IEC 9545).

FAL services and protocols are provided by FAL application entities (AEs) contained in the application processes. The FAL AE is composed of a set of object-oriented Application Service Elements (ASEs) and a Layer Management Entity (LME) that manages the AE. The ASEs provide communication services that operate on a set of related application process object (APO) classes. One of the FAL ASEs is a management ASE that provides a common set of services for management of the instances of FAL classes.

Although these services specify how requests and responses are issued and delivered from the perspective of applications, they do not include a specification of what the requesting and responding applications are to do with them. That is, these services only define what requests and responses applications can send or receive, not the functions of the applications themselves. This permits greater flexibility to the FAL-users in standardizing such object behaviour. In addition to these services, some supporting services are also defined in this specification to provide access to the FAL to control certain aspects of its operation.

8.2.2 Specifications

The principal objective of this specification is to specify the characteristics of conceptual application layer services suitable for time-critical communications, and thus supplement the OSI Basic Reference Model in guiding the development of application layer protocols for time-critical communications.

A secondary objective is to provide migration paths from previously existing industrial communications protocols. This latter objective gives rise to the diversity of services standardized as the various types of IEC 61158, and the corresponding protocols standardized in subparts of IEC 61158-6.

This specification may be used as the basis for formal application programming interfaces. Nevertheless, it is not a formal programming interface, and any such interface shall address implementation issues not covered by this specification, including

a) sizes and octet ordering of various multi-octet service parameters;

b) correlation of paired primitives for request and confirmation, or indication and response.

8.2.3 Conformance

This specification does not specify individual implementations or products, nor does it constrain the implementations of application layer entities in industrial automation systems.

There is no conformance of equipment to this application layer service definition standard. Instead, conformance is achieved through the implementation of conforming application layer protocols that fulfil any given type of application layer services as defined in this PAS.

8.3 Normative references

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

IEC 60559, *Binary floating-point arithmetic for microprocessor systems*

ISO/IEC 7498-1, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*

ISO/IEC 7498-3, *Information technology – Open Systems Interconnection – Basic Reference Model: Naming and addressing*

ISO/IEC 8822, *Information technology – Open Systems Interconnection – Presentation service definition*

ISO/IEC 9545, *Information technology – Open Systems Interconnection – Application layer structure*

ISO/IEC 10731:1994, *Information technology – Open Systems Interconnection – Basic Reference Model – Conventions for the definition of OSI services*

8.4 Terms, definitions, symbols, abbreviations, and conventions

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

8.4.1 ISO/IEC 7498-1 terms

a) application entity
b) application process
c) application protocol data unit
d) application service element
e) application entity invocation
f) application process invocation
g) application transaction
h) real open system
i) transfer syntax

8.4.2 ISO/IEC 8822 terms

a) abstract syntax
b) presentation context

8.4.3 ISO/IEC 9545 terms

a) application-association

b) application-context
c) application context name
d) application-entity-invocation
e) application-entity-type
f) application-process-invocation
g) application-process-type
h) application-service-element
i) application control service element

8.4.4 Fieldbus data link layer terms

For the purposes of this document, the following terms as defined in Clauses 7 and 8 apply.

a) DL-Time
b) DL-Scheduling-policy
c) DLCEP
d) DLC
e) DL-connection-oriented mode
f) DLPDU
g) DLSDU
h) DLSAP
k) link
l) ISO/IEC 8802-3 MAC address
m) DL-address

8.4.5 Fieldbus application layer specific definitions

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

8.4.5.1 Application

Function or data structure for which data are consumed or produced

8.4.5.2 Application objects

Multiple object classes that manage and provide a runtime exchange of messages across the network and within the network device.

8.4.5.3 Application process

Part of a distributed application on a network, which is located on one device and addressed unambiguously.

8.4.5.4 Application process identifier

Distinguishes multiple application processes used in a device.

8.4.5.5 Application process object

Component of an application process that is identifiable and accessible through an FAL application relationship.

NOTE Application process object definitions are composed of a set of values for the attributes of their class (see the definition for “application process object class”). Application process object definitions may be accessed remotely using the services of the FAL Object Management ASE. FAL Object Management services can be used to load or update object definitions, to read object definitions, and to create and delete application objects and their corresponding definitions dynamically.

8.4.5.6 Application process object class

A class of application process objects defined in terms of the set of their network-accessible attributes and services.

8.4.5.7 Application relationship

Cooperative association between two or more application-entity-invocations for the purpose of exchange of information and coordination of their joint operation. This relationship is activated either by the exchange of application-protocol-data-units or as a result of pre-configuration activities.

8.4.5.8 Application relationship application service element

Application-service-element that provides the exclusive means for establishing and terminating all application relationships.

8.4.5.9 Application relationship endpoint

Context and behaviour of an application relationship as seen and maintained by one of the application processes involved in the application relationship.

NOTE Each application process involved in the application relationship maintains its own application relationship endpoint.

8.4.5.10 Attribute

Description of an externally visible characteristic or feature of an object.

NOTE The attributes of an object contain information about variable portions of an object. Typically, they provide status information or govern the operation of an object. Attributes may also affect the behaviour of an object. Attributes are divided into class attributes and instance attributes.

8.4.5.11 Behaviour

Indication of how an object responds to particular events.

8.4.5.12 Channel

Single physical or logical link of an input or output application object of a server to the process.

8.4.5.13 Class

Set of objects, all of which represent the same type of system component.

NOTE A class is a generalization of an object, a template for defining variables and methods. All objects in a class are identical in form and behaviour but usually contain different data in their attributes.

8.4.5.14 Class attributes

Attribute shared by all objects within the same class.

8.4.5.15 Class code

Unique identifier assigned to each object class.

8.4.5.16 Class-specific service

Service defined by a particular object class to perform a required function that is not performed by a common service.

NOTE A class-specific object is unique to the object class that defines it.

8.4.5.17 Client

a) Object that uses the services of another (server) object to perform a task.

b) Initiator of a message to which a server reacts.

8.4.5.18 Consumer

Act of receiving data from a producer.

8.4.5.19 Consumer

Node or sink that receives data from a producer.

8.4.5.20 Consuming application

Application that consumes data.

8.4.5.21 Conveyance path

Unidirectional flow of APDUs across an application relationship.

8.4.5.22 Cyclic

Repetitive in a regular manner.

8.4.5.23 Data consistency

Means for coherent transmission and access of the input- or output-data object between and within client and server.

8.4.5.24 Device

Physical hardware connected to the link.

NOTE A device may contain more than one node.

8.4.5.25 Device profile

A collection of device-dependent information and functionality providing consistency between similar devices of the same device type.

8.4.5.26 Diagnostic information

All data available at the server for maintenance purposes.

8.4.5.27 End node

Producing or consuming node.

8.4.5.28 Endpoint

One of the communicating entities involved in a connection.

8.4.5.29 Error

Discrepancy between a computed, observed, or measured value or condition and the specified or theoretically correct value or condition.

8.4.5.30 Error class

General grouping for related error definitions and corresponding error codes.

8.4.5.31 Error code

Identification of a specific type of error within an error class.

8.4.5.32 Event

Instance of a change of conditions.

8.4.5.33 FIFO variable

A variable object class composed of a set of homogeneously typed elements, where the first written element is the first element that can be read.

NOTE In a fieldbus system, only one complete element can be transferred as a result of one service invocation.

8.4.5.34 Frame

Simplified synonym for data link protocol data unit (DLPDU).

8.4.5.35 Group

a) (General): a general term for a collection of objects.

b) (Addressing): when describing an address, an address that identifies more than one entity.

8.4.5.36 Invocation

Act of using a service or other resource of an application process.

NOTE Each invocation represents a separate thread of control that may be described by its context. Once the service completes, or use of the resource is released, the invocation ceases to exist. For service invocations, a service that has been initiated but not yet completed is referred to as an outstanding service invocation. For service invocations, an Invoke ID may be used to identify the service invocation unambiguously and differentiate it from other outstanding service invocations.

8.4.5.37 Index

Address of an object within an application process.

8.4.5.38 Instance

The actual physical occurrence of an object within a class that identifies one of many objects in the same object class.

EXAMPLE California is an instance of the object class US-state.

NOTE The terms object, instance, and object instance are used to refer to a specific instance.

8.4.5.39 Instance attributes

Attribute that is unique to an object instance and not shared by the object class.

8.4.5.40 Instantiated

Object that has been created in a device.

8.4.5.41 Logical device

A specific FAL class that abstracts a software component or a firmware component as an autonomous self-contained facility of an automation device.

8.4.5.42 Manufacturer ID

Identification of each product manufacturer by a unique number.

8.4.5.43 Management information

Network-accessible information that supports management of the operation of the fieldbus system, including the application layer.

NOTE Managing includes functions, such as controlling, monitoring, and diagnosis.

8.4.5.44 Network

A set of nodes connected by some type of communication medium, including any intervening repeaters, bridges, routers, and lower-layer gateways.

8.4.5.45 Object

Abstract representation of a particular component within a device, usually a collection of related data in the form of variables, and methods (procedures) for operating on that data that have clearly defined interface and behaviour.

8.4.5.46 Object dictionary

Collection of definitions, communication-specific attributes and parameters, and application-dependent data.

8.4.5.47 Object-specific service

Service unique to the object class that defines it.

8.4.5.48 Physical device

Automation or other network device.

8.4.5.49 Point-to-point connection

Connection that exists between exactly two application objects.

8.4.5.50 Pre-established AR endpoint

AR endpoint placed in an established state during configuration of the AEs that control its endpoints.

8.4.5.51 Process data

Object(s) that are already pre-processed and transferred cyclically for the purpose of information or further processing.

8.4.5.52 Produce

Act of sending data to be received by a consumer.

8.4.5.53 Producer

Node that is responsible for sending data.

8.4.5.54 Property

General term for descriptive information about an object.

8.4.5.55 Provider

Source of a data connection.

8.4.5.56 Publisher

Role of an AR endpoint that transmits APDUs onto the fieldbus for consumption by one or more subscribers.

NOTE A publisher may not be aware of the identity or number of subscribers.

8.4.5.57 Publishing manager

Role of an AR endpoint in which it issues one or more confirmed service request application protocol data units (APDUs) to a publisher to request that a specified object be published. Two types of publishing managers are defined by this PAS, pull publishing managers and push publishing managers, each of which is defined separately.

8.4.5.58 Push publisher

Type of publisher that publishes an object in an unconfirmed service request APDU.

8.4.5.59 Push publishing manager

Type of publishing manager that requests that a specified object be published using an unconfirmed service.

8.4.5.60 Push subscriber

Type of subscriber that recognizes received unconfirmed service request APDUs as published object data.

8.4.5.61 Server

a) Role of an application relationship endpoint (AREP) in which it returns a confirmed service response APDU to the client that initiated the request.

b) Object that provides services to another (client) object.

8.4.5.62 Service

Operation or function than an object and/or object class performs upon request from another object and/or object class.

8.4.5.63 Station

Host of one AP, identified by a unique data link connection endpoint (DLCEP)-address.

8.4.5.64 Subscriber

Role of an AREP in which it receives APDUs produced by a publisher.

8.4.6 Abbreviations and symbols

AE	Application Entity
AL	Application Layer
ALME	Application Layer Management Entity
ALP	Application Layer Protocol
APO	Application Object
AP	Application Process
APDU	Application Protocol Data Unit
AR	Application Relationship
AREP	Application Relationship End Point
ASCII	American Standard Code for Information Interchange
ASE	Application Service Element
Cnf	Confirmation
DL-	(as a prefix) Data Link -
DLCEP	Data Link Connection End Point
DLL	Data Link Layer
DLM	Data Link Management
DLSAP	Data Link Service Access Point
DLSDU	DL-service-data-unit
DNS	Domain Name Service
FAL	Fieldbus Application Layer
Ind	Indication
Req	Request
Rsp	Response

8.4.7 Conventions

8.4.7.1 Overview

The FAL is defined as a set of object-oriented ASEs. Each ASE is specified in a separate subsection. Each ASE PAS is composed of two parts: its class specification and its service specification.

The class specification defines the attributes of the class. Access to these attributes is beyond the scope of this specification except where specified. The service specification defines the services provided by the ASE.

8.4.7.2 General conventions

This specification uses the descriptive conventions given in ISO/IEC 10731.

8.4.7.3 Conventions for class definitions

Class definitions are described using templates. Each template consists of a list of attributes for the class. The general form of the template is as shown below.

FAL ASE:			**ASE name**
CLASS:			**Class name**
CLASS ID:			**#**
PARENT CLASS:			**Parent class name**
ATTRIBUTES:			
1	(o)	Key Attribute:	numeric identifier
2	(o)	Key Attribute:	name
3	(m)	Attribute:	attribute name(values)
4	(m)	Attribute:	attribute name(values)
4.1	(s)	Attribute:	attribute name(values)
4.2	(s)	Attribute:	attribute name(values)
4.3	(s)	Attribute:	attribute name(values)
5.	(c)	Constraint:	constraint expression
5.1	(m)	Attribute:	attribute name(values)
5.2	(o)	Attribute:	attribute name(values)
6	(m)	Attribute:	attribute name(values)
6.1	(s)	Attribute:	attribute name(values)
6.2	(s)	Attribute:	attribute name(values)
SERVICES:			
1	(o)	OpsService:	service name
2	(c)	Constraint:	constraint expression
2.1	(o)	OpsService:	service name
3	(m)	MgtService:	service name

(1) The FAL ASE: entry is the name of the FAL ASE that provides the services for the class being specified.

(2) The CLASS: entry is the name of the class being specified. All objects defined using this template will be an instance of this class. The class may be specified by this specification, or by a user of this specification.

(3) The CLASS ID: entry is a number that identifies the class being specified. This number is not used for RAPIEnet elements.

(4) The PARENT CLASS: entry is the name of the parent class for the class being specified. All attributes defined for the parent class and inherited by it are inherited for the class being defined, and therefore do not have to be redefined in the template for this class.

NOTE The parent-class TOP indicates that the class being defined is an initial class definition. The parent class TOP is used as a starting point from which all other classes are defined. The use of TOP is reserved for classes defined by this specification.

(5) The ATTRIBUTES label indicates that the following entries are attributes defined for the class.

a) Each of the attribute entries contains a line number in column 1; a mandatory (m), optional (o), conditional (c), or selector (s) indicator in column 2; an attribute type label in column 3; a name or a conditional expression in column 4; and an optional list of enumerated values in column 5. In the column following the list of values, the default value for the attribute may be specified.

b) Objects are normally identified by a numeric identifier or by an object name, or by both. In the class templates, these key attributes are defined under the key attribute.

c) The line number defines the sequence and the level of nesting of the line. Each nesting level is identified by period. The numbers below refer to the general template form above. Nesting is used to specify

i) fields of a structured attribute (4.1, 4.2, 4.3);

ii) attributes conditional on a constraint statement. Attributes may be mandatory (5.1) or optional (5.2) if the constraint is true. Not all optional attributes require constraint statements as does the attribute defined in (5.2);

iii) the selection fields of a choice type attribute (6.1 and 6.2).

(6) The SERVICES label indicates that the following entries are services defined for the class.

a) An (m) in column 2 indicates that the service is mandatory for the class, while an (o) indicates that it is optional. A (c) in this column indicates that the service is conditional. When all services defined for a class are defined as optional, at least one has to be selected when an instance of the class is defined.

b) The label “OpsService” designates an operational service (1).

c) The label “MgtService” designates a management service (2).

d) The line number defines the sequence and the level of nesting of the line. Each nesting level is identified by period. Nesting within the list of services is used to specify services conditional on a constraint statement.

8.4.7.4 Conventions for service definitions

8.4.7.4.1 General

The service model, service primitives, and time-sequence diagrams used are entirely abstract descriptions; they do not represent a specification for implementation.

8.4.7.4.2 Service parameters

Service primitives are used to represent interactions between service user and service provider (ISO/IEC 10731). They convey parameters that indicate information available in the user/provider interaction. In any particular interface, not all parameters shall be stated explicitly.

The service definition of this specification uses a tabular format to describe the component parameters of the ASE service primitives. The parameters that apply to each group of service primitives are set out in tables. Each table consists of up to five columns:

a) parameter name;

b) request primitive;

c) indication primitive;

d) response primitive;

e) confirmation primitive.

One parameter, or a component, is listed in each row of each table. Under the appropriate service primitive columns, a code is used to specify the type of usage of the parameter on the primitive specified in the column:

M The parameter is mandatory for the primitive.

U The parameter is a user option, and may or may not be provided depending on dynamic usage of the service user. When not provided, a default value for the parameter is assumed.

C The parameter is conditional upon other parameters or upon the environment of the service user.

— (blank) The parameter is never present.

S The parameter is a selected item.

Some entries are further qualified by items in parentheses. These may be

a) a parameter-specific constraint:

"(=)" indicates that the parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table;

b) an indication that some note applies to the entry:

"(n)" indicates that the following note "n" contains additional information pertaining to the parameter and its use.

8.4.7.4.3 Service procedures

The service procedures are defined in terms of

a) the interactions between application entities through the exchange of fieldbus APDUs;

b) the interactions between an application layer service provider and an application layer service user in the same system through the invocation of application layer service primitives.

These procedures are applicable to instances of communication between systems that support time-constrained communications services within the fieldbus application layer.

8.5 Concepts

8.5.1 Common concepts

8.5.1.1 Overview

The fieldbus is intended to be used in factories and process plants to interconnect primary automation devices (for example, sensors, actuators, local display devices, annunciators, programmable logic controllers, small single-loop controllers, and stand-alone field controls) with control and monitoring equipment located in control rooms.

Primary automation devices are associated with the lowest levels of the industrial automation hierarchy and perform a limited set of functions within a definite time window. Some of these functions include diagnostics, data validation, and handling of multiple inputs and outputs.

These primary automation devices, also called field devices, are located close to the process fluids, the fabricated part, the machine, the operator, and the environment. This use positions the fieldbus at the lowest levels of the computer integrated manufacturing (CIM) architecture.

Some of the expected benefits in using fieldbus systems are reductions in wiring, increases in the amount of data exchanged, a wider distribution of control between the primary automation devices and the control room equipment, and satisfaction of time-critical constraints.

This subsection describes the fundamentals of the FAL. Detailed descriptive information about each of the FAL ASEs can be found in the overview subsection of each of the communication model specifications.

8.5.1.2 Architectural relationships

8.5.1.2.1 Relationship to the application layer of the OSI Basic Reference Model

The functions of the FAL have been described according to OSI layering principles. However, the FAL's architectural relationship to the lower layers is different, as follows (see Figure 43).

a) The FAL groups OSI functions together with extensions to cover time-critical requirements. The OSI application layer structure standard (ISO/IEC 9545) was used as a basis for the FAL specification.

b) The FAL uses the services of the underlying layer direct. The underlying layer may be the DLL or any layer in between. When using the underlying layer, the FAL may provide functions normally associated with the OSI middle layers for proper mapping onto the underlying layer.

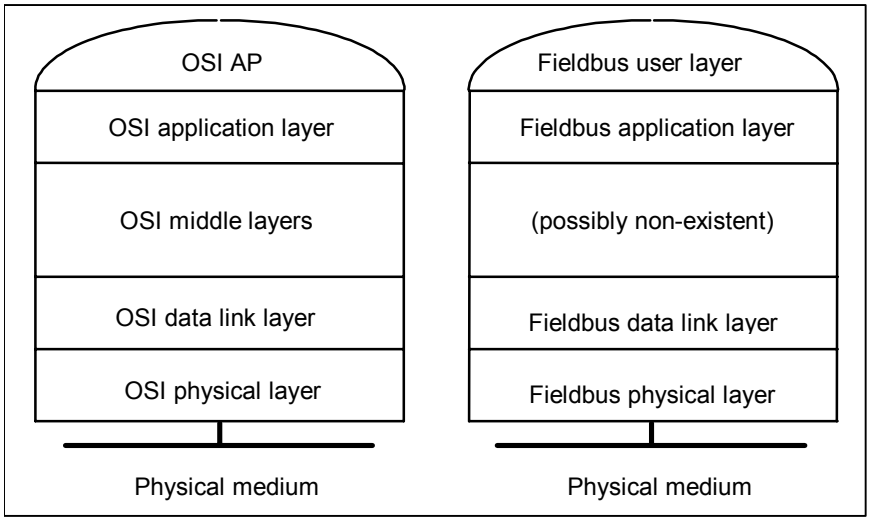


Figure 43 – Relationship to the OSI Basic Reference Model

8.5.1.2.2 Relationships to other fieldbus entities

8.5.1.2.2.1 General

The FAL architectural relationships illustrated in Figure 44 have been designed to support the interoperability needs of time-critical systems distributed in the fieldbus environment.

In this environment, the FAL provides communications services to time-critical and non-time-critical applications located in fieldbus devices.

In addition, the FAL uses the DLL directly to transfer its application layer protocol data units, using a set of data transfer services and a set of supporting services to control the operational aspects of the DLL.

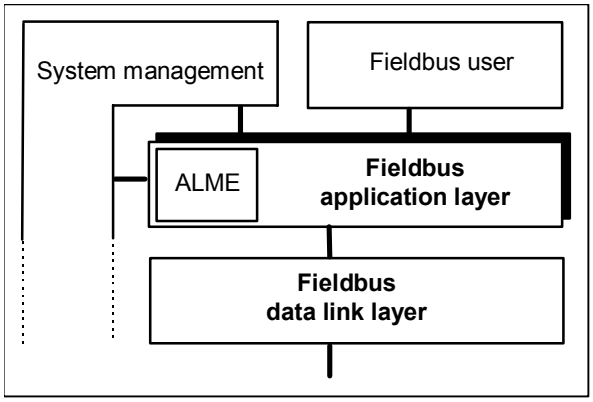


Figure 44 – Architectural positioning of the fieldbus application layer

8.5.1.2.2.2 Use of the fieldbus data link layer

The FAL provides network access to fieldbus APs. It interfaces direct to the fieldbus DLL for transfer of its APDUs.

The DLL provides various types of service to the FAL for transfer of data between data link endpoints (for example, DLSAPs and DLCEPs).

8.5.1.2.2.3 Support for fieldbus applications

Fieldbus applications appear to the network as application processes (APs). APs are the components of a distributed system that may be individually identified and addressed.

Each AP contains an FAL AE that provides network access for the AP. That is, each AP communicates with other APs through its AE. In this sense, the AE provides a window of visibility into the AP.

APs contain identifiable components that are also visible across the network. These components are represented to the network as APOs. They may be identified by one or more key attributes. They are located at the address of the application process that contains them.

The services used to access the APOs are provided by APO-specific application service elements (ASEs) contained in the FAL. These ASEs are designed to support user, function block, and management applications.

8.5.1.2.2.4 Support for system management

The FAL services can be used to support various management operations, including management of fieldbus systems, applications, and the fieldbus network.

8.5.1.2.2.5 Access to FAL layer management entities

One LME may be present in each FAL entity on the network. Fieldbus application layer management entities (FALMEs) provide access to the FAL for system management purposes.

The set of data accessible by the system manager is referred to as the system management information base (SMIB). Each FALME provides the FAL portion of the SMIB. Implementation of the SMIB is beyond the scope of this specification.

8.5.1.3 Fieldbus application layer structure

8.5.1.3.1 Overview

The structure of the FAL is a refinement of the OSI application layer structure (ISO/IEC 9545). As a result, the organization of this subsection is similar to that of ISO/IEC 9545. Certain concepts presented here have been refined from ISO/IEC 9545 for the fieldbus environment.

The FAL differs from the other layers of the OSI Basic Reference Model in the following two principal aspects.

a) The OSI Basic Reference Model defines the association, a single type of application layer communications channel to connect APs to each other. The FAL defines the application relationship (AR), of which there are several types, to permit application processes (APs) to communicate with each other.

b) The FAL uses the DLL and not the OSI presentation layer to transfer its APDUs. Therefore, there is no explicit presentation context available to the FAL. The FAL protocol may not be used concurrently with other application layer protocols between the same pair or set of data link service access points

8.5.1.3.2 Fundamental concepts

The operation of time-critical real open systems is modelled in terms of interactions between time-critical APs. The FAL permits these APs to pass commands and data between them.

Cooperation between APs requires that they share sufficient information to interact and carry out processing activities in a coordinated manner. Their activities may be restricted to a single fieldbus segment or they may span multiple segments. The FAL has been designed using a modular architecture to support the messaging requirements of these applications.

Cooperation between APs also sometimes requires that they share a common sense of time. The FAL or the DLL may provide for the distribution of time to all devices. They may also define local device services that can be used by APs to access the distributed time.

The remainder of this subclause describes each of the modular components of the architecture and their relationships with each other. The components of the FAL are modelled as objects, each of which provides a set of FAL communication services for use by applications. The FAL objects and their relationships are described below. The detailed specifications of FAL objects and their services are provided in the following clauses of this PAS. IEC 61158-6 specifies the protocols necessary to convey these object services between applications.

8.5.1.3.3 Fieldbus application processes

8.5.1.3.3.1 Definition of the fieldbus AP

In the fieldbus environment, an application may be partitioned into a set of components and distributed across a number of devices on the network. Each of these components is referred to as a fieldbus AP. A fieldbus AP is a variation of an AP as defined in the ISO OSI Reference Model (ISO/IEC 7498). Fieldbus APs may be addressed unambiguously by at least one individual DLL service access point address. Unambiguous addressing in this context means that no other AP may simultaneously be located at the same address. This definition does not prohibit an AP from being located at more than one individual or group data link service access point (DLSAP) address.

8.5.1.3.3.2 Communication services

Fieldbus APs communicate with each other using confirmed and unconfirmed services (ISO/IEC 10731). The services defined in this PAS for the FAL specify the semantics of the services as seen by the requesting and responding APs. The syntax of the messages used to

convey the service requests and responses is defined in Chapter 10. The AP behaviour associated with the services is specified by the AP.

Confirmed services are used to define request/response exchanges between APs.

On the other hand, unconfirmed services are used to define the unidirectional transfer of messages from one AP to one or more remote APs. From a communications perspective, there is no relationship between separate invocations of unconfirmed services as there is between the request and response of a confirmed service.

8.5.1.3.3.3 AP interactions

8.5.1.3.3.3.1 General

In the fieldbus environment, APs may interact with other APs as necessary to achieve their functional objectives. No constraints are imposed by this PAS on the organization of these interactions or the possible relationships that may exist between them.

For example, in the fieldbus environment, interactions may be based on request/response messages sent directly between APs, or on data/events sent by one AP for use by others. These two models of interaction between APs are referred to as client/server and publisher/subscriber interactions, respectively.

The services supported by an interaction model are conveyed by application relationship endpoints (AREPs) associated with the communicating APs. The role that the AREP plays in the interaction (for example, client, server, peer, publisher, or subscriber) is defined as an attribute of the AREP.

8.5.1.3.3.3.2 Client/server interactions

Client/server interactions are characterized by bidirectional data flow between a client AP and one or more server APs. Figure 45 illustrates the interaction between a single client and a single server. In this type of interaction, the client may issue a confirmed or unconfirmed request to the server to perform some task. If the service is confirmed then the server will always return a response. If the service is unconfirmed, the server may return a response using an unconfirmed service defined for this purpose.

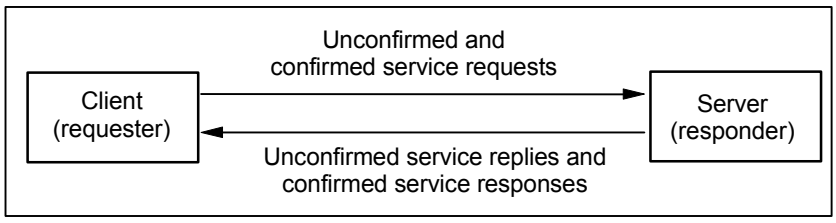


Figure 45 – Client/server interactions

8.5.1.3.3.3 Publisher/subscriber interactions

8.5.1.3.3.3.1 General

Publisher/subscriber interactions involve a single publisher AP and a group of one or more subscriber APs. This type of interaction has been defined to support variations of two models of interaction between Aps: the “pull” model and the “push” model. In both models, the setup of the publishing AP is outside the scope of this specification. The RAPIEnet protocol supports only the “push” model.

8.5.1.3.3.3.3.2 Pull model interactions

In the “pull” model, the publisher receives a request to publish from a remote publishing manager, and broadcasts (or multicasts) its response across the network. The publishing manager is responsible only for initiating publishing by sending a request to the publisher.

Subscribers wishing to receive the published data listen for responses transmitted by the publisher. In this fashion, data are “pulled” from the publisher by requests from the publishing manager.

Confirmed FAL services are used to support this type of interaction. Two characteristics of this type of interaction differentiate it from the others. First, a typical confirmed request/response exchange is performed between the publishing manager and the publisher. However, the underlying conveyance mechanism provided by the FAL returns the response not just to the publishing manager, but also to all subscribers wishing to receive the published information. This is accomplished by having the DLL transmit the response to a group address, rather than to the individual address of the publishing manager. Therefore, the response sent by the publisher contains the published data and is multicast to the publishing manager and to all subscribers.

The second difference occurs in the behaviour of the subscribers. Pull model subscribers, referred to as pull subscribers, are capable of accepting published data in confirmed service responses without having issued the corresponding request. Figure 46 illustrates these concepts.

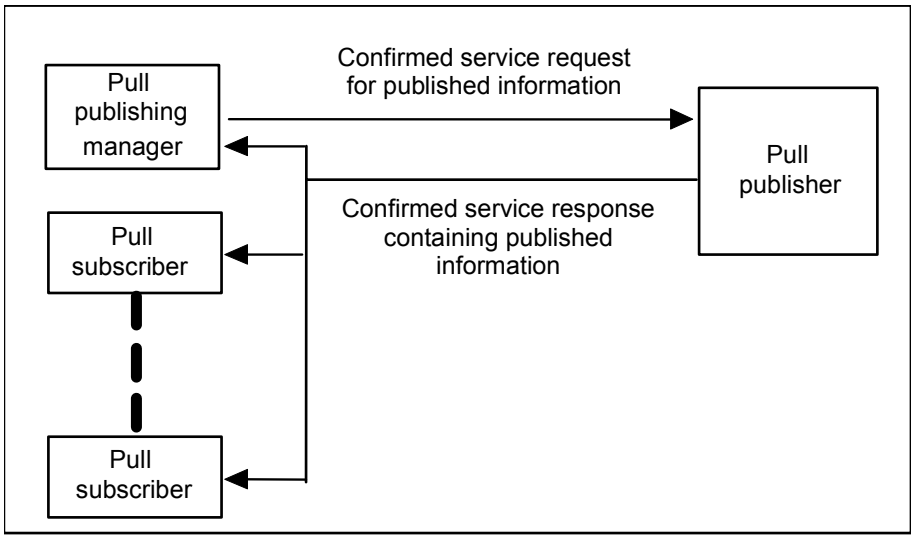


Figure 46 – Pull model interactions

8.5.1.3.3.3.3.3 Push model interactions

In the “push” model, two services may be used: one confirmed and one unconfirmed. The confirmed service is used by the subscriber in the request to subscribe to the publishing activity. The response to this request is returned to the subscriber, following the client/server model of interaction. This exchange is only necessary when the subscriber and the publisher are located in different APs.

The unconfirmed service of the push model is used by the publisher to distribute its information to subscribers. In this case, the publisher is responsible for invoking the correct unconfirmed service at the appropriate time and for supplying the appropriate information. In this fashion, it is configured to “push” its data onto the network.

Subscribers for the push model receive the published unconfirmed services distributed by publishers. Figure 47 illustrates the concept of the push model.

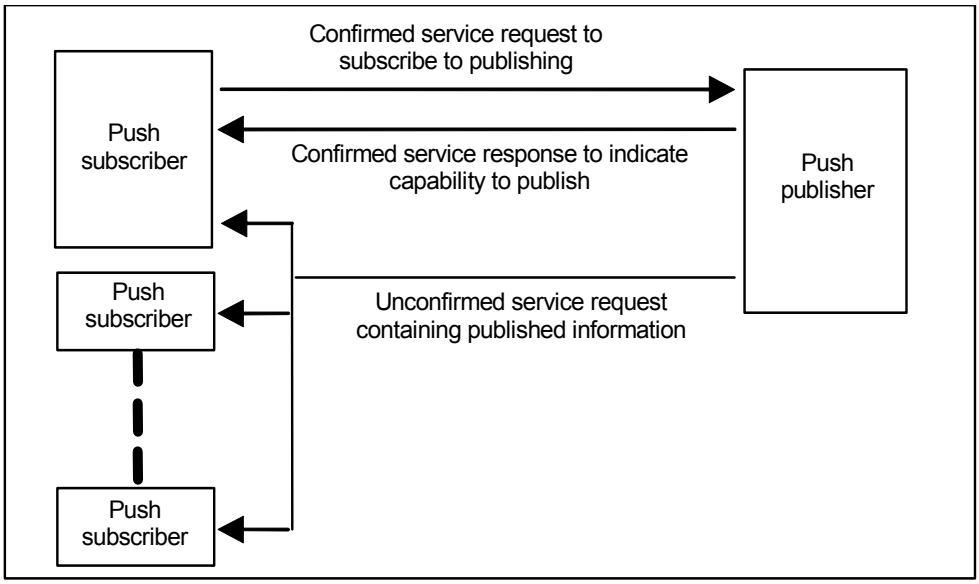


Figure 47 – Push model interactions

8.5.1.3.3.3.4 Timeliness of published information

To support the perishable nature of published information, the FAL may support four types of timeliness defined for publisher/subscriber interactions. Each makes it possible for subscribers of published data to determine if the data they are receiving is current or stale. These types are realized through mechanisms inside the DLL. Each is described briefly below.

Type	Description
Transparent	This type of timeliness allows the user application process to determine the timeliness quality of the data that it generates and have the timeliness quality accompany the information when it is transferred across the network. In this type, the network provides no computation or measurement of timeliness. It merely conveys the timeliness quality provided with the data by the user application process
Residence	When the FAL submits data from the publishing AP to the DLL for transmission, the DLL starts a timer. If the timer expires before the data has been transmitted, the DLL marks the buffer as "not timely" and conveys this timeliness information with the data
Synchronized	This type of timeliness requires the coordination of two pieces of published information: the data to be published and a special "sync mark." When the sync mark is received from the network, a timer starts in each of the participating stations. Subsequently, when data are received for transmission by the DLL at the publishing station, or when the transmitted data are received from the network at a subscribing station, the DLL timeliness attribute for the data is set to TRUE. It remains TRUE until reception of the next sync mark or until the timer expires. Data received after the timer expires but before the next sync mark arrives do not cause the timeliness attribute to be reset to TRUE. It is only reset to TRUE if data are received within the time window after receipt of the sync mark. Data transmitted by the publisher station with the timeliness attribute set to FALSE maintains the setting of FALSE at each of the subscribers, regardless of their timer operation
Update	This type of timeliness requires coordination of the same two pieces of published information defined for *synchronized* timeliness. In this type, the sync mark also starts a timer in each of the participating stations. Similar to *synchronized* timeliness, expiry of the timer always causes the timeliness attribute to be set to FALSE. Unlike *synchronized* timeliness, receipt of new data at any time (not just within the time window started with the receipt of a sync mark) causes the timeliness attribute to be set to TRUE

8.5.1.3.3.4 AP structure

The internals of APs may be represented by one or more APOs and accessed through one or more AEs. AEs provide the communication capabilities of the AP. For each fieldbus AP, there

is one and only one FAL AE. APOs are the network representation of application-specific capabilities (user application process objects) of an AP that are accessible through its FAL AE.

8.5.1.3.3.5 AP class

An AP class is a definition of the attributes and services of an AP. The standard class definition for APs is defined in this chapter. User-defined classes may also be specified. Class identifiers, also described in this chapter, are assigned from a set reserved for this purpose.

8.5.1.3.3.6 AP type

As described above in the previous subsections, APs are defined by instantiating an AP class. Each AP definition is composed of the attributes and services selected for the AP from those defined by its AP class. In addition, an AP definition contains values for one or more of the attributes selected for it. When two APs share the same definition, that definition is referred to as an AP type. Thus, an AP type is a generic specification of an AP that may be used to define one or more APs.

8.5.1.3.4 Application process objects (APOs)

8.5.1.3.4.1 Definition

An APO is a network representation of a specific aspect of an AP. Each APO represents a specific set of information and processing capabilities of an AP accessible through services of the FAL. APOs are used to represent these capabilities to other APs in a fieldbus system.

From the perspective of the FAL, an APO is modelled as a network-accessible object contained in an AP or in another APO (APOs may contain other APOs). APOs provide the network definition for objects contained in an AP that are remotely accessible. The definition of an APO includes an identification of the FAL services that can be used by remote APs for remote access. The FAL services, as shown in Figure 48, are provided by the FAL communications entity of the AP, known as the FAL applications entity (FAL AE).

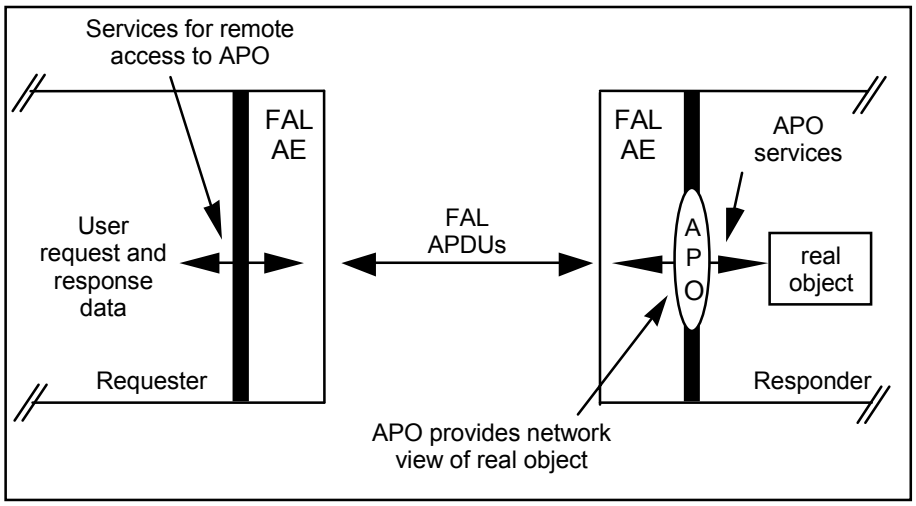


Figure 48 – APOs services conveyed by the FAL

In Figure 48, remote APs acting as clients may access the real object by sending requests through the APO that represents the real object. Local aspects of the AP convert between the network view (the APO) of the real object and the internal AP view of the real object.

To support the publisher/subscriber model of interaction, information about the real object can be published through its APO. Remote APs acting as subscribers see the APO view of the published information instead of having to know any of the real object-specific details.

8.5.1.3.4.2 APO classes

An APO class is a generic specification for a set of APOs, each of which is described by the same set of attributes and accessed using the same set of services.

APO classes provide the mechanism for standardizing network-visible aspects of APs. Each standard APO class definition specifies a particular set of network-accessible AP attributes and services. 9.10 specifies the syntax and the procedures used by the FAL protocol to provide remote access to the attributes and services of an APO class.

Standard APO classes are specified by this specification for the purpose of standardizing remote access to APs. User-defined classes may also be specified.

User-defined classes are defined as subclasses of standardized APO classes or of other user-defined classes. They may be defined by identifying new attributes or by indicating that optional attributes for the parent class are mandatory for the subclass. The conventions for defining classes defined in this chapter may be used for this purpose. The method for registering or otherwise making these new class definitions available for public use is outside the scope of this specification.

8.5.1.3.4.3 APOs as instances of APO classes

APO classes are defined in this specification using templates. These templates are used not only to define APO classes but also to specify the instances of a class.

Each APO defined for an AP is an instance of an APO class. Each APO provides the network view of a real object contained in an AP. An APO is defined by

a) selecting the attributes from its APO class template that are to be accessible from the real object;
b) assigning values to one or more attributes indicated as key in the template. Key attributes are used to identify the APO;
c) assigning values to zero, one, or more non-key attributes for the APO. Non-key attributes are used to characterize the APO;
d) selecting the services from the template that may be used by remote APs to access the real object.

Subclause 9.4.7.3 specifies the conventions for class templates. These conventions provide for the definition of mandatory, optional, and conditional attributes and services.

Mandatory attributes and services are required to be present in all APOs of the class. Optional attributes and services may be selected on an APO-by-APO basis for inclusion in an APO. Conditional attributes and services are defined with an accompanying constraint statement. Constraint statements specify the conditions under which the attribute is present in an APO.

8.5.1.3.4.4 APO types

APO types provide the mechanism for defining standard APOs.

As described above, APOs are defined by instantiating an APO class. Each APO definition is composed of the attributes and services selected for the APO from those defined by its APO class. In addition, an APO definition contains values for one or more of the attributes selected for it. When two APOs share the same definition except for the key attribute settings, that definition is referred to as an APO type. Thus, an APO type is a generic specification of an APO that may used to define one or more APOs.

8.5.1.3.5 Application entities

8.5.1.3.5.1 Definition of FAL AE

An application entity provides the communication capabilities for a single AP. An FAL AE provides a set of services and the supporting protocols to enable communications between APs in a fieldbus environment. The services provided by FAL AEs are grouped into ASEs in such a way that the FAL services provided to an AP are defined by the ASEs that its FAL AE contains. Figure 49 illustrates this concept.

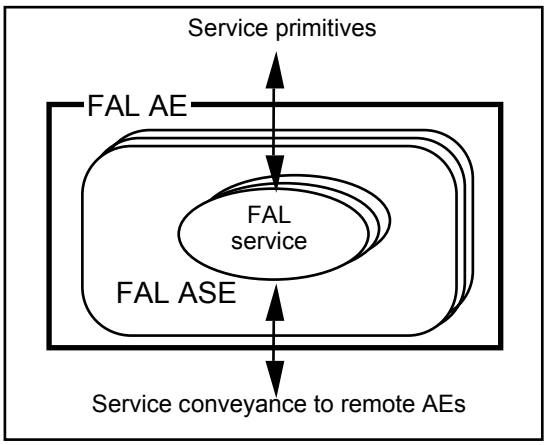


Figure 49 – Application entity structure

8.5.1.3.5.2 AE type

Application entities that provide the same set of ASEs are of the same AE-type. Two AEs that share a common set of ASEs are capable of communicating with each other.

8.5.1.3.6 Fieldbus ASEs

8.5.1.3.6.1 General

An ASE, as defined in ISO/IEC 9545, is a set of application functions that provide a capability for the interworking of application-entity-invocations for a specific purpose. ASEs provide a set of services for conveying requests and responses to and from application processes and their objects. AEs, as defined above, are represented by a collection of ASE invocations within the AE.

8.5.1.3.6.2 FAL services

FAL services convey functional requests/responses between APs. Each FAL service is defined to convey requests and responses for access to a real object modelled as an FAL accessible object.

The FAL defines both confirmed and unconfirmed services. Confirmed service requests are sent to the AP containing the real object. An invocation of a confirmed service request may be identified by a user-supplied invoke ID. This invoke ID is returned in the response by the AP containing the real object. When present, it is used by the requesting AP and its FAL AE to associate the response with the appropriate request.

Unconfirmed services may be sent from the AP containing the real object to send information about the object. They also may be sent to the AP containing the real object to access the real object. Both types of unconfirmed services may be defined for the FAL.

8.5.1.3.6.3 Definition of FAL ASEs

8.5.1.3.6.3.1 General

8.5.1.3.6.3.2 Object-management ASE

A special object-management ASE may be specified for the FAL to provide services for the management of objects. Its services are used to access object attributes, and create and delete object instances. These services are used to manage network-visible AP objects accessed though the FAL. The specific operational services that apply to each object type are specified in the definition of the ASE for the object type. Figure 50 illustrates the integration of management and operational services for an object within an AP.

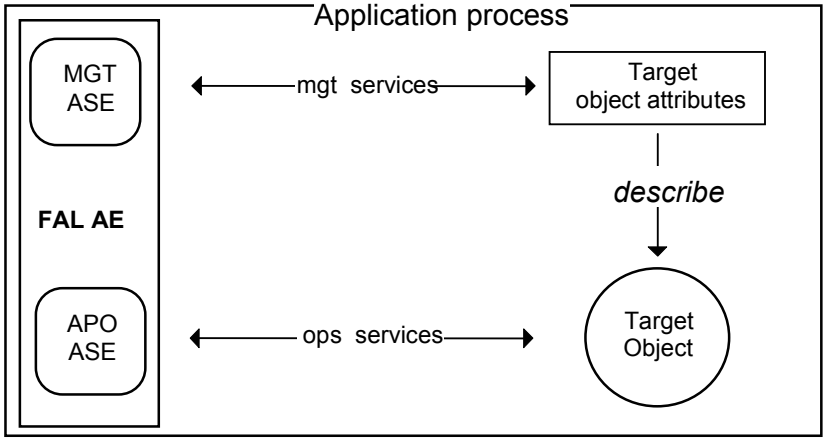


Figure 50 – FAL management of objects

8.5.1.3.6.3.3 AP ASE

An AP ASE may be specified for the identification and control of FAL APs. The attributes defined by the AP ASE give details about the AP's creator, and list its contents and capabilities.

8.5.1.3.6.3.4 APO ASEs

The FAL specifies a set of ASEs with services defined for accessing the APOs of an AP. The APO ASEs defined for the FAL are defined by each communication model.

8.5.1.3.6.3.5 AR ASE

An AR ASE is specified to establish and maintain ARs, which are used to convey FAL APDUs between or among APs. ARs represent application layer communication channels between APs. AR ASEs are responsible for providing services at the endpoints of ARs. AR ASE services may be defined for establishing, terminating, and aborting ARs. They may also be defined for conveying APDUs for the AE, and for informing the user of the local status of the AR. In addition, local services may be defined for accessing certain aspects of AR endpoints.

8.5.1.3.6.4 FAL service conveyance

FAL APO ASEs provide services to convey requests and responses between service users and real objects.

To convey service requests and responses, there are three types of activities defined for the sending user, and three corresponding types defined for the receiving user. At the sending user, FAL APO ASEs accept service requests and responses to be conveyed. They also select the type of FAL APDU that will be used to convey the request or response, and encode

the service parameters in the body portion. Then, they submit the encoded APDU body to the AR ASE for conveyance.

At the receiving user, FAL APO ASEs receive encoded APDU bodies from the AR ASE. They decode the APDU bodies and extract the service parameters conveyed by them. Then, they deliver the service request or response to the user. Figure 51 illustrates these concepts.

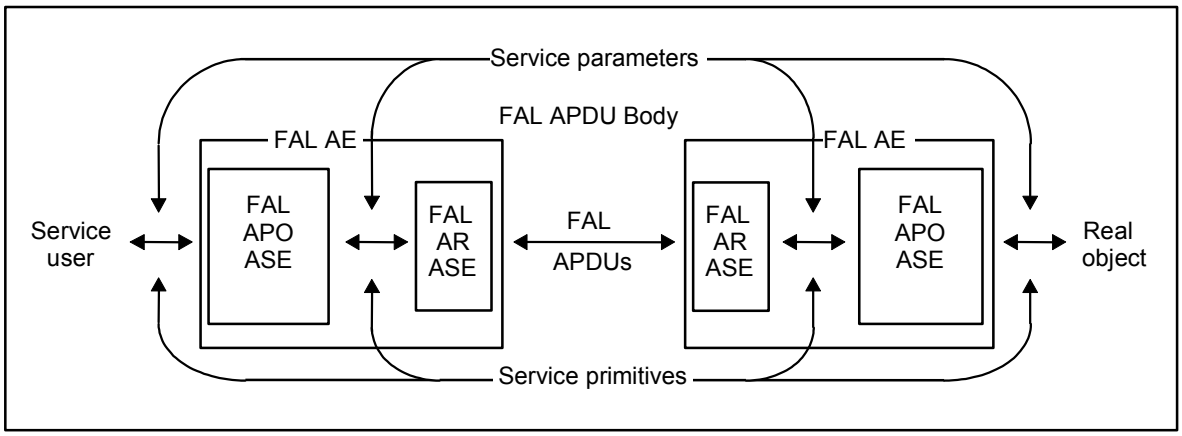


Figure 51 – ASE service conveyance

8.5.1.3.6.5 FAL presentation context

The presentation context in the OSI environment is used to distinguish the APDUs of one ASE from another, and to identify the transfer syntax rules used to encode each APDU. However, the fieldbus communications architecture does not include a presentation layer. Therefore, an alternate mechanism is provided for the FAL by each of the specific types of communication models.

8.5.1.3.7 Application relationships

8.5.1.3.7.1 Definition of AR

ARs represent communication channels between APs. They define how information is communicated between APs. Each AR is characterized by how it conveys ASE service requests and responses from one AP to another. These characteristics are described below.

8.5.1.3.7.2 AR-endpoints

ARs are defined as a set of cooperating APs. The AR ASE in each AP manages an endpoint of the AR, and maintains its local context. The local context of an AR endpoint is used by the AR ASE to control the conveyance of APDUs on the AR.

8.5.1.3.7.3 AR-endpoint classes

ARs are composed of a set of endpoints of compatible classes. AR endpoint classes are used to represent AR endpoints that convey APDUs in the same way. Through the standardization of endpoint classes, ARs for different models of interaction can be defined.

8.5.1.3.7.4 AR cardinality

ARs characterize communications between APs. One of the characteristics of an AR is the number of AR endpoints in the AR. ARs that convey services between two APs have a cardinality of 1:1. Those that convey services from one AP to a number of APs have a cardinality of 1:many. Those that convey services to or from multiple APs have a cardinality of many:many.

8.5.1.3.7.5 Accessing objects through ARs

ARs provide access to APs and the objects within them via the services of one or more ASEs. Therefore, one characteristic is the set of ASE services that may be conveyed to and from these objects by the AR. The list of services that can be conveyed by the AR is selected from those defined for the AE.

8.5.1.3.7.6 AR conveyance paths

ARs are modelled as one or two conveyance paths between AR endpoints. Each path conveys APDUs in one direction between one or more AR endpoints. Each receiving AR endpoint for a conveyance path receives all APDUs transmitted on the AR by the sending AR endpoint.

8.5.1.3.7.7 AREP roles

As APs interact with each other through endpoints, a basic determinant of their compatibility is the role that they play in the AR. The role defines how an AREP interacts with other AREPs in the AR.

For example, an AREP may operate as a client, a server, a publisher, or a subscriber. When an AREP interacts with another AREP on a single AR as both a client and a server, it is defined to have the role of “peer.”

Certain roles may be capable of initiating service requests, while others may be capable only of responding to service requests. This part of the definition of a role identifies the requirement for an AR to be capable of conveying requests in either direction, or in only one direction.

8.5.1.3.7.8 AREP buffers and queues

AREPs may be modelled as a queue or as a buffer. APDUs transferred over a queued AREP are delivered in the order received for conveyance. The transfer of APDUs over a buffered AREP is different. In this case, an APDU to be conveyed by the AR ASE is placed in a buffer for transfer. When the DLL gains access to the network, it transmits the contents of the buffer.

When the AR ASE receives another conveyance request, it replaces the previous contents of the buffer regardless of whether they were transmitted. Once an APDU is written into a buffer for transfer, it is preserved in the buffer until it is overwritten by the next APDU to be transmitted. While in the buffer, an APDU may be read more than once without deleting it from the buffer or changing its contents.

The operation is similar at the receiving end. The receiving endpoint places a received APDU into a buffer for access by the AR ASE. When a subsequent APDU is received, it overwrites the previous APDU in the buffer regardless of whether or not it was read. Reading the APDU from the buffer is not destructive; it does not destroy or change the contents of the buffer, allowing the contents to be read from the buffer one or more times.

8.5.1.3.7.9 User-triggered and scheduled conveyance

Another characteristic of an AREP is the time frame in which it conveys service requests and responses. User-triggered AREPs convey requests and responses immediately upon submission by the user. This is asynchronous with respect to network operation.

A scheduled AREP conveys requests and responses at predefined intervals, regardless of when they are received for transfer. A scheduled AREP may be capable of indicating when transferred data was submitted late for transmission, or when it was submitted on time but transmitted late.

8.5.1.3.7.10 AREP timeliness

AREPs convey APDUs between applications using the services of the DLL. When the timeliness capabilities are defined for an AREP and supported by the DLL, the AREP forwards the timeliness indicators provided by the DLL. These indicators make it possible for subscribers of published data to determine if the data they are receiving is current or stale.

To support these types of timeliness, the publishing AREP establishes a publisher data link connection reflecting the type of timeliness configured for it by management. After establishing the connection, the AREP receives user data and submits it to the DLL for transmission where the timeliness procedures are performed. When the DLL transmits the data, it includes the current timeliness status with the data.

At the subscriber AREP, a data line connection is opened to receive published data that reflects the type of timeliness configured for it by management. The DLL computes the timeliness of received data and then delivers it to the AREP. The data are then delivered to the user AP through the appropriate ASE.

8.5.1.3.7.11 Definition and creation of AREPs

AREP definitions specify instances of AREP classes. AREPs may be predefined or they may be defined using a “create” service if their AE supports this capability.

AREPs may be pre-defined and pre-established or they may be pre-defined and established dynamically. Figure 52 depicts these two cases. AREPs also may require both dynamic definition and establishment or they may be defined dynamically in such a way that they may be used without any establishment, i.e., they are defined in an established state.

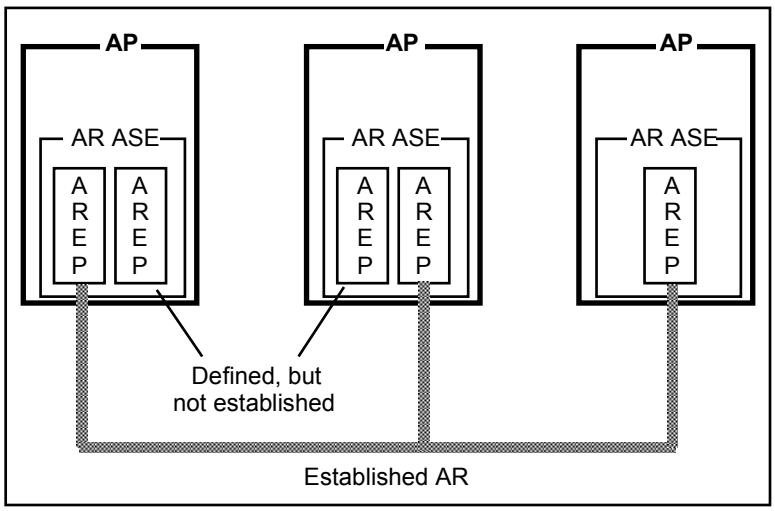


Figure 52 – Defined and established AREPs

8.5.1.3.7.12 AR establishment and termination

ARs may be established either before the operational phase of the AP or during its operation. When established during the operation of an AP, the AR is established through the exchange of AR APDUs.

Once an AR has been established, an AR may be terminated gracefully or it may be aborted, depending on the capabilities of the AR.

8.5.1.4 Fieldbus application layer naming and addressing

8.5.1.4.1 General

This subclause refines the principles defined in ISO 7498-3 that involve the identification (naming) and location (addressing) of APOs referenced through the FAL.

This subclause also defines how names and numeric identifiers are used to identify APOs accessible through the FAL. In addition, this subclause indicates how addresses from underlying layers are used to locate APs in the fieldbus environment.

8.5.1.4.2 Identifying objects accessed through the FAL

8.5.1.4.2.1 General

APOs accessed though the FAL are identified independent of their location. That is, if the location of the AP containing the APO changes, the APO may still be referenced using the same set of identifiers.

Identifiers for APs and APOs in the FAL are defined as key attributes in the class definitions for APOs. Two types of key attribute are commonly used in these APO definitions: names and numeric identifiers.

8.5.1.4.2.2 Names

Names are string-oriented identifiers. They are defined to permit APs and APOs to be named in the system where they are used. Therefore, although the scope of an APO name is specific to the AP in which it resides, the assignment of the name is administered in the system in which it is configured.

Names may be descriptive, although this is not mandatory. Descriptive names make it possible to provide meaningful information about the object they name, such as its use.

Names may also be coded. Coded names make it possible to identify an object using a short, compressed form of a name. They are typically simpler to transfer and process, but more difficult to understand than descriptive names.

8.5.1.4.2.3 Numeric identifiers

Numeric identifiers are identifiers whose values are numbers. They are designed for efficient use in the fieldbus system and may be assigned to APOs by their AP for efficient access.

8.5.1.4.3 Addressing APs accessed through the FAL

Fieldbus addresses represent the network locations of APs. Addresses relevant to the FAL are the addresses of the underlying layers that are used to locate the AREPs of an AP.

8.5.1.5 Architecture summary

This section presents a summary of the FAL architecture. Figure 53 illustrates the major components and how they relate to each other.

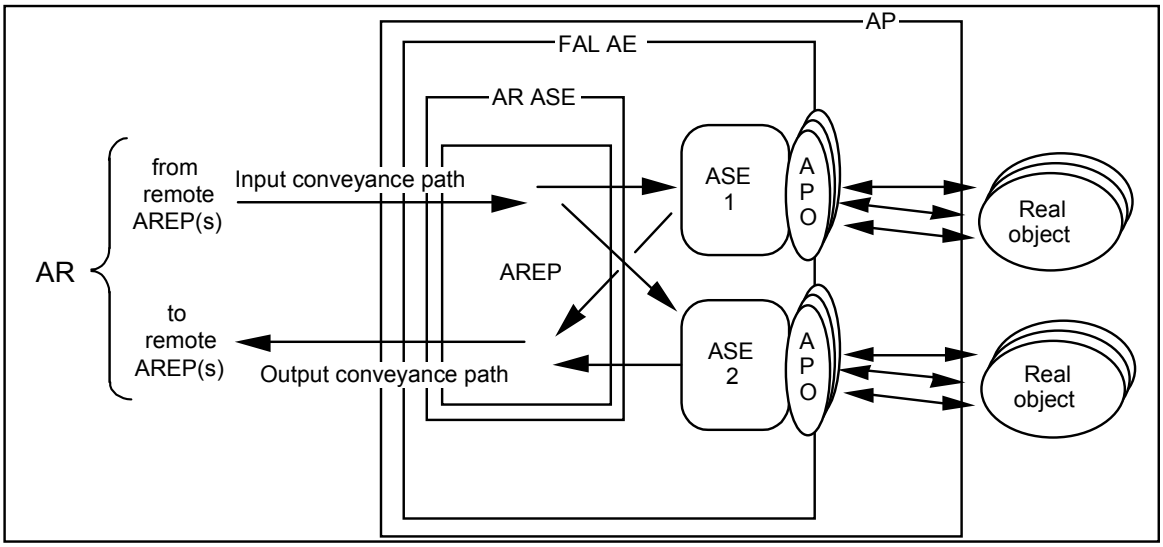


Figure 53 – FAL architectural components

Figure 53 shows an AP that communicates through the FAL AE. The AP represents its internal real objects as APOs for remote access to them. Two ASEs that provide remote access services to their related APOs are shown. The AR ASE contains a single AREP that conveys service requests and responses for the ASEs to one or more remote AREPs located in remote APs.

8.5.1.6 FAL service procedures

8.5.1.6.1 FAL confirmed service procedures

The requesting user invokes a confirmed-service request primitive of its FAL. The appropriate FAL ASE builds the related confirmed-service-request APDU body and conveys it on the specified AR.

Upon receipt of the confirmed-service-request APDU body, the responding ASE decodes it. If a protocol error did not occur, the ASE delivers a confirmed-service indication primitive to its user.

If the responding user is able to process the request successfully, the user returns a confirmed-service response (+) primitive.

If the responding user is unable to process the request successfully, the service fails and the user issues a confirmed-service response (-) primitive indicating the reason.

The responding ASE builds a confirmed-service-response APDU body for a confirmed-service response (+) primitive or a confirmed-service-error APDU body for a confirmed-service response (-) primitive and conveys it on the specified AR.

Upon receipt of the returned APDU body, the initiating ASE delivers a confirmed-service confirmation primitive to its user that specifies success or failure, and the reason for failure if a failure occurred.

8.5.1.6.2 FAL unconfirmed service procedures

The requesting user invokes an unconfirmed-service request primitive from its FAL AE. The appropriate FAL ASE builds the related unconfirmed-service request APDU body and conveys it on the specified AR.

Upon receipt of the unconfirmed-request APDU body, the receiving ASE(s) participating in the AR delivers the appropriate unconfirmed-service indication primitive to its user. Timeliness parameters are included in the indication primitive if the AR that conveyed the APDU body supports timeliness.

8.5.1.7 Common FAL attributes

In the specifications of the FAL classes that follow, many classes use the following common attributes. Therefore, these attributes are defined once here instead of with the other attributes for each individual class, except for the data type class.

ATTRIBUTES:

1	(o)	Key attribute:	Numeric identifier
2	(o)	Key attribute:	Name
3	(o)	Attribute:	User description
4	(o)	Attribute:	Object revision

Numeric identifier

This optional key attribute specifies the numeric ID of the object. It is used as a shorthand reference by the FAL protocol to identify the object. There are three possibilities for identification purposes: numeric identifier, name, or both. This attribute is required for the data type model.

Name

This optional key attribute specifies the name of the object. There are three possibilities for identification purposes: numeric identifier, name, or both.

User description

This optional attribute contains user-defined descriptive information about the object.

Object revision

This optional attribute specifies the revision level of the object. It is a structured attribute composed of major and minor revision numbers. If object revision is supported, it contains both a major revision and a minor revision with a value in the range 0–15 for each. The major and minor revision fields are used as follows.

a) The major revision field contains the major revision value for the object. A change to the major revision indicates that interoperability is affected by the change.

b) The minor revision field contains the minor revision value for the object. A change to the minor revision indicates that interoperability was not affected by the change, i.e., users of the object will continue to be capable of interoperating with the object if its minor revision is changed, provided that the major revision remains the same.

8.5.1.8 Common FAL service parameters

In the specifications of the FAL services that follow, many services use the following parameters. Therefore, they are defined once here instead of with the other parameters for each of the services.

AREP

This parameter contains sufficient information to identify the AREP locally so it can be used to convey the service. This parameter may use a key attribute of the AREP to identify the application relationship. When an AREP supports multiple contexts (established using the initiate service) at the same time, the AREP parameter is extended to identify the context as well as the AREP.

Invoke ID

This parameter identifies this invocation of the service. It is used to associate service requests with responses. Therefore, no two outstanding service invocations may be identified by the same invoke ID value.

FAL ASE/FAL class

This parameter specifies the FAL ASE (for example, AP, AR, variable, data type, event, function-invocation, load-region) and the FAL class of the ASE (for example, AREP, variable-list, notifier, action).

Numeric ID

This parameter is the numeric identifier of the object.

Error info

This parameter provides error information for service errors. It is returned in confirmed service response (-) primitives. It is composed of the following elements.

a) **Error class.** This parameter indicates the general class of error. Valid values are specified in the definition of the error code parameter below.

b) **Error code.** This parameter identifies the specific service error.

c) **Additional code.** If an error is encountered when processing the request, this optional parameter identifies the error specific to the object being accessed. When used, the value submitted in the response primitive is delivered unchanged in the confirmation primitive.

d) **Additional detail.** This optional parameter contains user data that accompanies a negative response. When used, the value submitted in the response primitive is delivered unchanged in the confirmation primitive.

8.5.1.9 APDU size

The APDU size is dependent on the communication model.

8.5.2 Type specific concepts

The industrial automation and process control system consists of primary automation devices (for example, sensors, actuators, local display devices, annunciators, programmable logic controllers, small single loop controllers, and stand-alone field controls) with control and monitoring equipment.

Data transfer between these devices is performed by peer-to-peer or multicast communication.

Figure 54 illustrates the interaction between RAPIEnet FAL and the DLL. RAPIEnet supports cyclic and acyclic data transfer for its own application processes. RAPIEnet can also be used in parallel with TCP/IP or UDP communication. The use of other standard communication protocols is beyond this specification.

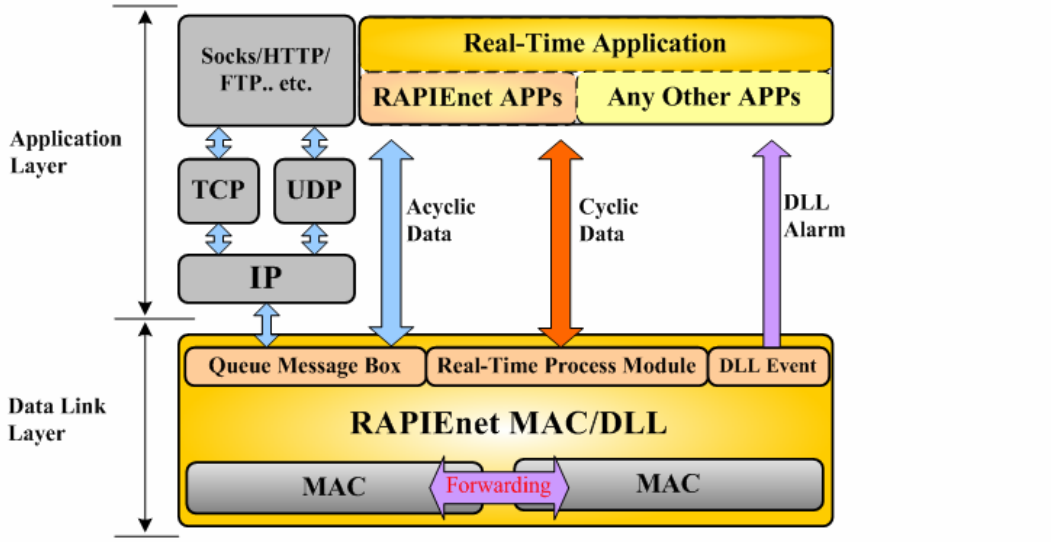


Figure 54 – Interaction between FAL and DLL

RAPIEnet supports the publisher-subscriber communication model for cyclic data sharing. Figure 55 illustrates this model. The publisher periodically multicasts preconfigured data, and subscribers receive the data. This cyclic data sharing is the most widely used model in industrial applicationas.

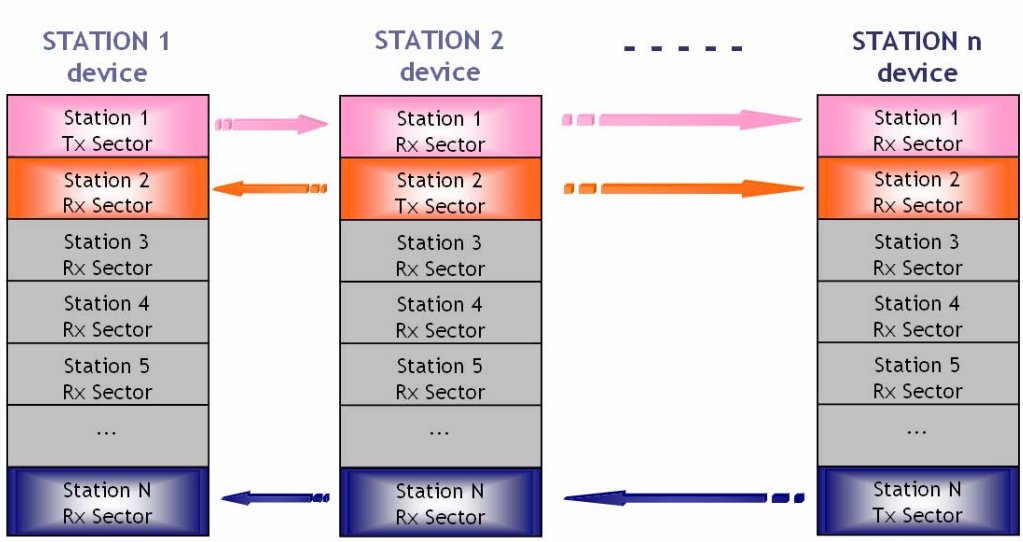


Figure 55 – Publisher-subscriber communication model

RAPIEnet supports the client-server communication model for event-triggered data transfer. Figure 56 illustrates this model. The client requests data as dictated by internal or external events. The server replies with the data. This can be used for event-triggered or user-triggered application processes.

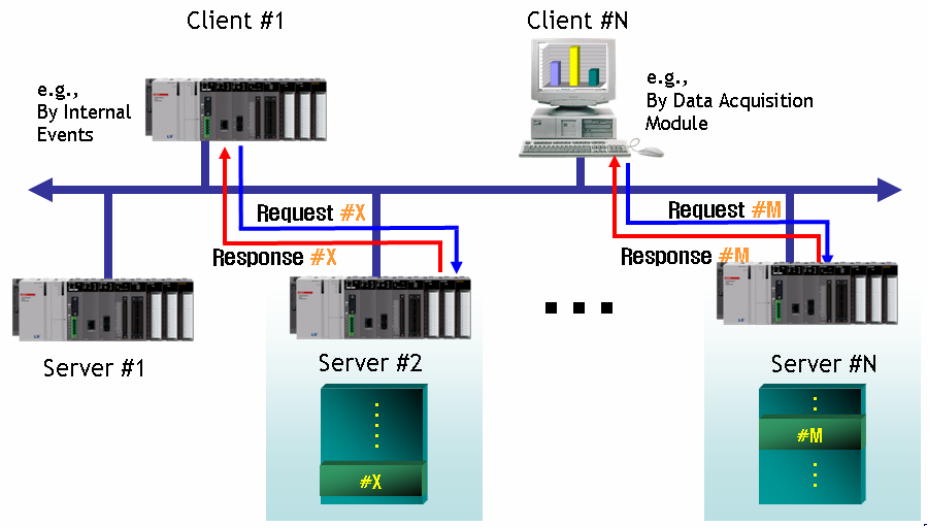


Figure 56 – Client-server communication model

8.5.2.1 Node, AP, and object dictionary

Each node hosts exactly one AP. All APOs for this AP are collected in the object dictionary (OD). Figure 57 illustrates the object model.

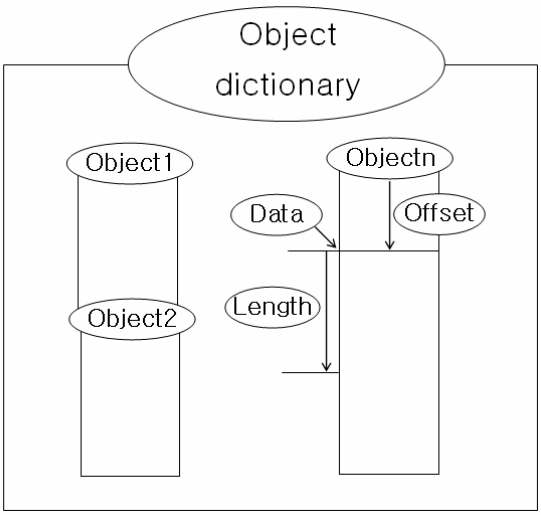


Figure 57 – Object model

The overall structure of the OD is as described in Table 101.

Table 101 – Overall structure of the OD

Area	Contents
Data type area	Definition of data types
Communication profile area	Contains communication-specific parameters for the RAPIEnet network. These entries are common to all devices
Manufacturer-specific area	Definition of manufacturer-specific variables
Device profile area	Definition of the variables defined in a device profile (outside the scope of this PAS)
Reserved area	Reserved for future use

8.5.2.2 APO ASEs

The FAL ASEs of a RAPIEnet application and their interrelationships are described in Figure 58.

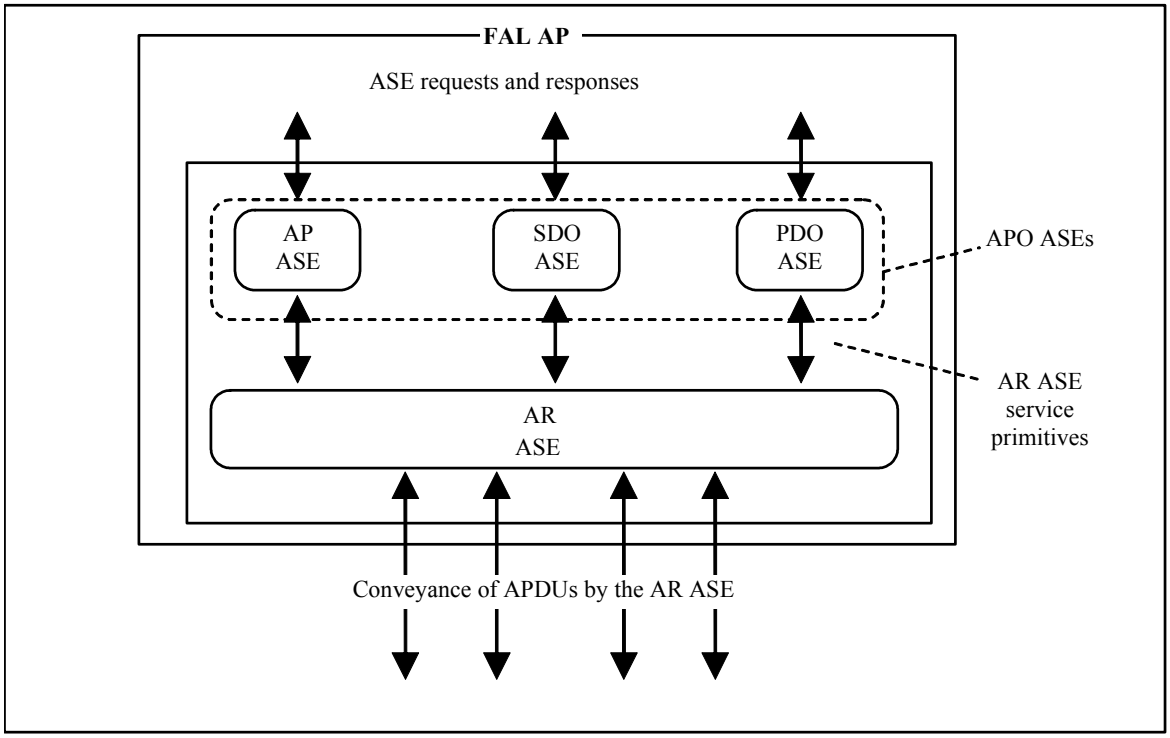


Figure 58 – ASEs of a RAPIEnet application

8.6 Data type ASE

8.6.1 General

8.6.1.1 Overview

The fieldbus data types specify the machine-independent syntax for application data conveyed by FAL services. The FAL supports the definition and transfer of both basic and constructed data types. The encoding rules for the data types specified in this clause are given in 9.10.

Basic types are atomic types that cannot be decomposed. Constructed types are types composed of basic types and other constructed types. This PAS does not constrain their complexity or depth of nesting.

Data types are defined as instances of the data type class, as shown in Figure 59. Only a subset of the data types defined in this clause are shown in this figure. Defining new types is accomplished by providing a numeric ID and supplying values for the attributes defined for the data type class.

The data type definitions shown in Figure 59 are represented as a class/format/instance structure beginning with the “Data type” data type class.

The basic data classes are used to define fixed-length and bit string data types. Standard types taken from ISO/IEC 8824 are referred to as simple data types. Other standard basic

data types are defined specifically for fieldbus applications, and are referred to as specific types.

The constructed types specified in this PAS are strings, arrays, and structures. There are no standard types defined for arrays or structures.

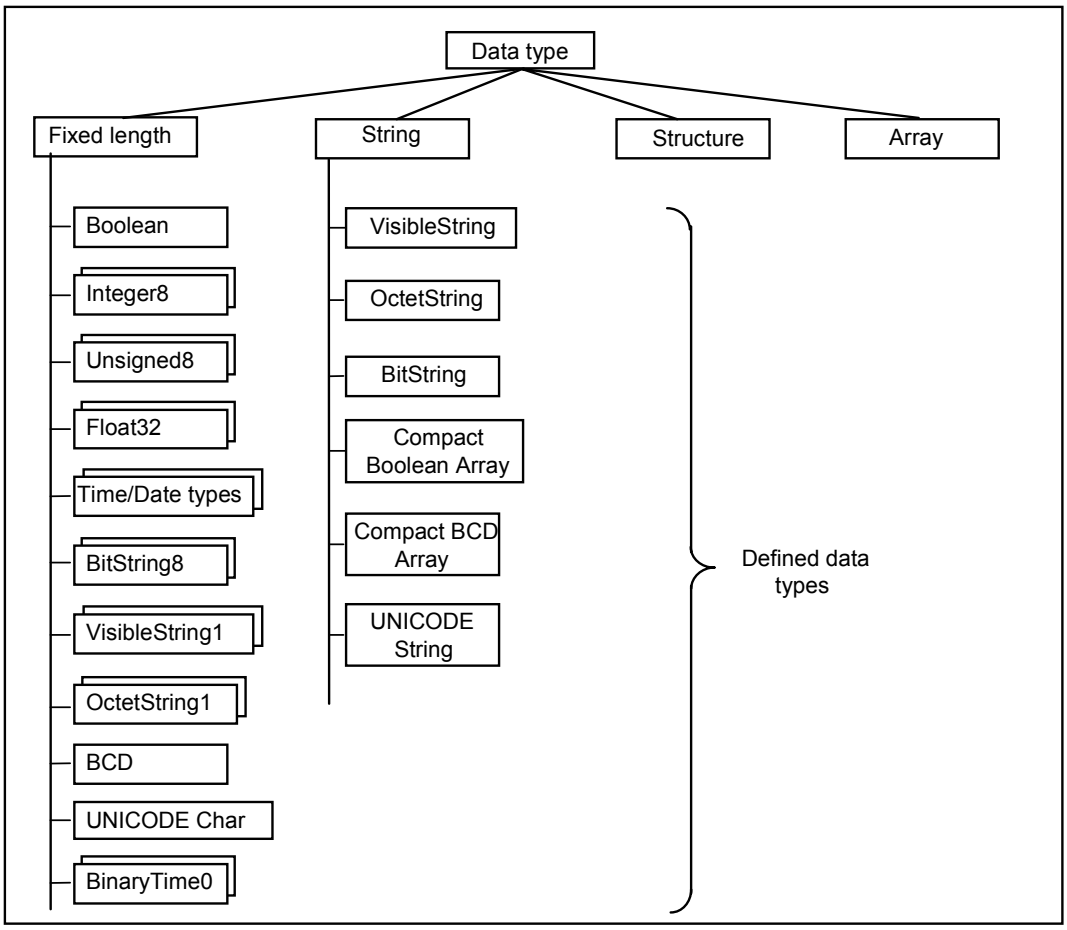


Figure 59 – Data type class hierarchy example

8.6.1.2 Basic type overview

Most basic types are defined from a set of ISO/IEC 8824 types (simple types). Some ISO/IEC 8824 types have been extended for fieldbus specific use (specific types).

Simple types are ISO/IEC 8824 universal types. They are defined in this specification to provide them with fieldbus class identifiers.

Specific types are basic types defined specifically for use in the fieldbus environment. They are defined as simple class subtypes.

Basic types have a constant length. Two variations are defined, one for defining data types whose length is an integral number of octets, and one for defining data types whose length is an arbitrary number of bits.

NOTE Boolean, Integer, OctetString, VisibleString, and UniversalTime are defined in this PAS for the purpose of assigning fieldbus class identifiers to them. This specification does not change their definitions as specified in ISO/IEC 8824.

8.6.1.3 Fixed-length type overview

The length of fixed-length types is an integral number of octets.

8.6.1.4 Constructed type overview

8.6.1.4.1 Strings

A string is composed of an ordered set, variable in number, of homogeneously typed fixed-length elements.

8.6.1.4.2 Arrays

An array is composed of an ordered set of homogeneously typed elements. This specification places no restriction on the data type of array elements, but it does require that each element be of the same type. Once defined, the number of elements in an array may not be changed.

8.6.1.4.3 Structures

A structure is made of an ordered set of heterogeneously typed elements called fields. Like arrays, this specification does not restrict the data type of fields. However, the fields within a structure do not have to be of the same type.

8.6.1.4.4 Nesting level

This specification permits arrays and structures to contain other arrays and structures. It places no restriction on the number of nesting levels allowed. However, the FAL services defined to access data provide for partial access to the level negotiated by the Initiate service. The default number of levels for partial access is one.

When an array or structure contains constructed elements, access to a single element in its entirety is always provided. Access to subelements of the constructed element is also provided, but only when explicitly negotiated during AR establishment or when explicitly preconfigured on pre-established ARs.

NOTE For example, suppose that a data type named “employee” is defined to contain the structure “employee name” and “employee name” is defined to contain “last name” and “first name.” To access the “employee” structure, the FAL permits independent access to the entire structure and to the first level field “employee name.” Without explicitly negotiating partial access to more than one level, independent access to “last name” or “first name” would not be possible; their values could only be accessed together as a unit through access to “employee” or “employee name.”

8.6.1.5 Specification of user-defined data types

Users may find it necessary to define custom data types for their own applications. User-defined types are supported by this specification as instances of data type classes.

User-defined types are specified in the same manner that all FAL objects are specified. They are defined by providing values for the attributes specified for their class.

8.6.1.6 Transfer of user data

User data are transferred between applications by the FAL protocol. All encoding and decoding is performed by the FAL-user.

The rules for encoding user data in FAL protocol data units depend on the data type. These are defined in 9.10. User-defined data types for which there are no encoding rules are transferred as variable-length sequences of octets. The format of the data within the octet string is defined by the user.

8.6.2 Formal definition of data type objects

8.6.2.1 Data type class

8.6.2.1.1 Template

The data type class specifies the root of the data type class tree. Its parent class “TOP” indicates the top of the FAL class tree.

FAL ASE:			**DATA TYPE ASE**
CLASS:	**DATA TYPE**		
CLASS ID:			5 (FIXED-LENGTH & STRING), 6 (STRUCTURE), 12 (ARRAY)
PARENT CLASS:			TOP
ATTRIBUTES:			
1	(o)	Key attribute:	Data type numeric identifier
2	(o)	Key attribute:	Data type name
3	(m)	Attribute:	Format (FIXED-LENGTH, STRING, STRUCTURE, ARRAY)
4	(c)	Constraint:	Format = FIXED-LENGTH | STRING
4.1	(m)	Attribute:	Octet length
5	(c)	Constraint:	Format = STRUCTURE
5.1	(m)	Attribute:	Number of fields
5.2	(m)	Attribute:	List of fields
5.2.1	(o)	Attribute:	Field name
5.2.2	(m)	Attribute:	Field data type
6	(c)	Constraint:	Format = ARRAY
6.1	(m)	Attribute:	Number of array elements
6.2	(m)	Attribute :	Array element data type

8.6.2.1.2 Attributes

Data type numeric identifier

This optional attribute identifies the numeric identifier of the related data type.

Data type name

This optional attribute identifies the name of the related data type.

Format

This attribute identifies the data type as a fixed-length, string, array, or data structure.

Octet length

This conditional attribute defines the representation of the dimensions of the associated type object. It is present when the value of the format attribute is “FIXED-LENGTH” or “STRING.” For FIXED-LENGTH data types, it represents the length in octets. For STRING data types, it represents the length in octets for a single element of a string.

Number of fields

This conditional attribute defines the number of fields in a structure. It is present when the value of the format attribute is “STRUCTURE.”

List of fields

This conditional attribute is an ordered list of fields contained in the structure. Each field is specified by its number and its type. Fields are numbered sequentially from 0 (zero) in the order in which they occur. Partial access to fields within a structure is supported by identifying the field by number. This attribute is present when the value of the format attribute is “STRUCTURE.”

Field name

This conditional, optional attribute specifies the name of the field. It may be present when the value of the format attribute is “STRUCTURE.”

Field data type

This conditional attribute specifies the data type of the field. It is present when the value of the format attribute is “STRUCTURE.” This attribute may itself specify a constructed data type either by referencing a constructed data type definition by its numeric ID, or by embedding a constructed data type definition here. When embedding a description, the embedded-data type description shown below is used.

Number of array elements

This conditional attribute defines the number of elements for the array type. Array elements are indexed from “0” through “n-1” for an array of “n” elements. This attribute is present when the value of the format attribute is “ARRAY.”

Array element data type

This conditional attribute specifies the data type for the elements of an array. All elements of the array have the same data type. It is present when the value of the format attribute is “ARRAY.” This attribute may itself specify a constructed data type either by referencing a constructed data type definition by its numeric ID, or by embedding a constructed data type definition here. When embedding a description, the embedded-data type description shown below is used.

Embedded-data type description

This attribute is used to define embedded data types recursively in a structure or array. The template below defines its contents. The attributes shown in the template are defined above in the data type class, except for the embedded-data type attribute, which is a recursive reference to this attribute. It is used to define nested elements.

ATTRIBUTES:

1	(m)	Attribute:	Format(FIXED-LENGTH, STRING, STRUCTURE, ARRAY)
2	(c)	Constraint:	Format = FIXED-LENGTH | STRING
2.1	(m)	Attribute:	Data type numeric ID value
2.2	(m)	Attribute:	Octet length
3	(c)	Constraint:	Format = STRUCTURE
3.1	(m)	Attribute:	Number of fields
3.2	(m)	Attribute:	List of fields
3.2.1	(m)	Attribute:	Embedded data type description
4	(c)	Constraint:	Format = ARRAY
4.1	(m)	Attribute:	Number of array elements
4.2	(m)	Attribute:	Embedded data type description

8.6.3 FAL defined data types

8.6.3.1 Fixed-length types

8.6.3.1.1 Boolean types

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type numeric identifier	=	1
2	Data type name	=	Boolean
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	1

This data type expresses a Boolean data type with the values TRUE and FALSE.

8.6.3.1.2 Date/time types

8.6.3.1.2.1 TimeOfDay

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type numeric identifier	=	12
2	Data type name	=	TimeOfDay
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	6

This data type is composed of two elements of unsigned values and expresses the time of day and the date. The first element is an Unsigned32 data type that gives the time after midnight in milliseconds. The second element is an Unsigned16 data type that gives the date as a count of the days from January 1, 1984.

8.6.3.1.2.2 TimeDifference

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type numeric identifier	=	13
2	Data type name	=	TimeDifference
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	6

This data type is composed of two elements of unsigned values that express a length of time. The first element is an Unsigned32 data type that provides the fractional portion of one day in milliseconds. The second element is an Unsigned16 data type that provides the number of days.

8.6.3.1.3 Numeric types

8.6.3.1.3.1 Real32

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type numeric identifier	=	8
2	Data type name	=	Real32
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	4

This type has a length of four octets. The format for Real32 is that defined by IEC 60559 as single precision.

8.6.3.1.3.2 Real64

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type numeric identifier	=	17
2	Data type name	=	Real64
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	8

This type has a length of eight octets. The format for Real64 is that defined by IEC 60559 as double precision.

8.6.3.1.3.3 Integer types

8.6.3.1.3.3.1 Integer8

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type numeric identifier	=	2
2	Data type name	=	Integer8
3	Format	=	FIXED-LENGTH
4.1	Octet length	=	1

This integer type is a two's complement binary number with a length of one octet.

8.6.3.1.3.3.2 Integer16

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type numeric identifier	=	3
2	Data type name	=	Integer16
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	2

This integer type is a two's complement binary number with a length of two octets.

8.6.3.1.3.3.3 Integer32

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type numeric identifier	=	4
2	Data type name	=	Integer32
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	4

This integer type is a two's complement binary number with a length of four octets.

8.6.3.1.3.3.4 Integer64

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type numeric identifier	=	21
2	Data type name	=	Integer64
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	8

This integer type is a two's complement binary number with a length of eight octets.

8.6.3.1.3.4 Unsigned types

8.6.3.1.3.4.1 Unsigned8

CLASS: **Data type**

ATTRIBUTES:

1	Data type numeric identifier	=	5
2	Data type name	=	Unsigned8
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	1

This type is a binary number. The most significant bit of the most significant byte is always the most significant bit of the binary number; no sign bit is included. This unsigned type has a length of one octet.

8.6.3.1.3.4.2 Unsigned16

CLASS: **Data type**

ATTRIBUTES:

1	Data type numeric identifier	=	6
2	Data type name	=	Unsigned16
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	2

This type is a binary number. The most significant bit of the most significant byte is always the most significant bit of the binary number; no sign bit is included. This unsigned type has a length of two octets.

8.6.3.1.3.4.3 Unsigned32

CLASS: **Data type**

ATTRIBUTES:

1	Data type numeric identifier	=	7
2	Data type name	=	Unsigned32
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	4

This type is a binary number. The most significant bit of the most significant byte is always the most significant bit of the binary number; no sign bit is included. This unsigned type has a length of four octets.

8.6.3.1.3.4.4 Unsigned64

CLASS: **Data type**

ATTRIBUTES:

1	Data type numeric identifier	=	27
2	Data type name	=	Unsigned64
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	8

This type is a binary number. The most significant bit of the most significant byte is always the most significant bit of the binary number; no sign bit is included. This unsigned type has a length of eight octets.

8.6.3.2 String types

8.6.3.2.1 OctetString

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type numeric identifier	=	10
2	Data type name	=	OctetString
3	Format	=	STRING
3.1	Octet length	=	1 to n

An OctetString is an ordered sequence of octets, numbered from 1 to n. For the purposes of discussion, octet 1 of the sequence is referred to as the first octet. The order of transmission is defined in Chapter 10.

8.6.3.2.2 VisibleString

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type numeric identifier	=	9
2	Data type name	=	VisibleString
3	Format	=	STRING
3.1	Octet length	=	1 to n

This type is defined as the ISO/IEC 646 string type.

8.6.3.2.3 UnicodeString

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type numeric identifier	=	11
2	Data type name	=	UnicodeString
3	Format	=	STRING
3.1	Octet length	=	1 to n

This type is defined as the UNICODE string type.

8.6.4 Data type ASE service specification

There are no operational services defined for the type object.

8.7 Communication model specification

8.7.1 ASEs

8.7.1.1 Application process ASE

8.7.1.1.1 Overview

This specification models a fieldbus AP. Fieldbus APs represent the information and processing resources of a system that can be accessed through FAL services.

The ASE in the FAL that provides these services is called an Application Process ASE (AP ASE). In the AP ASE, the AP is modelled and accessed as an APO with a standardized and predefined identifier.

8.7.1.1.2 AP class PAS

8.7.1.1.2.1 Formal model

The AP class specifies the attributes and services defined for application processes. Its parent class “TOP” indicates the top of the FAL class tree.

ASE:			**AP ASE**
CLASS:			**AP**
CLASS ID:			**not used**
PARENT CLASS:			**TOP**
ATTRIBUTES:			
1	(m)	Attribute:	DL-address
2	(m)	Attribute:	MAC address
3	(m)	Attribute:	Port information
4	(m)	Attribute:	Protocol version
5	(m)	Attribute:	Device type
6	(m)	Attribute:	Device description
7	(m)	Attribute:	Hardware-Version
8	(m)	Attribute:	Serial Number
9	(m)	Attribute:	Software-Version
10	(m)	Attribute:	Software-Date
11	(m)	Attribute:	Vendor ID
12	(m)	Attribute:	Product Code
13	(m)	Attribute:	Device flags
14	(m)	Attribute:	Device state
15	(m)	Attribute:	tx_cnt_normal
16	(m)	Attribute:	tx_cnt_all
17	(m)	Attribute:	rx_cnt_normal
18	(m)	Attribute:	rx_cnt_all
19	(m)	Attribute:	relay_cnt_normal
20	(m)	Attribute:	relay_cnt_all
SERVICES:			
1	(m)	OpsService:	Identify
2	(m)	OpsService:	Status

8.7.1.1.2.2 Attributes

DL-address
See 8.3.6.4.1

MAC address
See 8.3.6.4.7

Port information
See 8.3.6.4.8

Protocol version
See 8.3.6.4.9

Device type
See 8.3.6.4.10

Device description
See 8.3.6.4.11

Hardware-Version
This attribute contains the hardware version of the device.

Serial Number
This attribute contains the serial number of the device.

Software-Version
This attribute contains the software version of the device.

Software-Date
This attribute contains the software date of the device.

Vendor ID
This attribute contains the vendor ID of the device.

Product Code
This attribute contains the product code of the device.

Device flags
See 8.3.6.4.2

Device state
See 8.3.6.4.3

tx_cnt_normal
This attribute contains the value of the transmitted count of normal packets only.

tx_cnt_all
This attribute contains the value of the transmitted count of both error packets and normal packets.

rx_cnt_normal
This attribute contains the value of the received count of normal packets.

rx_cnt_all
This attribute contains the value of the received count of both error packets and normal packets.

relay_cnt_normal
This attribute contains the value of the relayed count of normal packets.

relay_cnt_all
This attribute contains the value of the relayed count of both error packets and normal packets.

8.7.1.1.3 AP ASE service PAS

8.7.1.1.3.1 Supported services

This subsection contains the definition of services that are unique to this ASE. The services defined for this ASE are IDENTIFY and STATUS.

8.7.1.1.3.2 IDENTIFY service

8.7.1.1.3.2.1 Service overview

This confirmed service may be used to obtain the information from the device.

8.7.1.1.3.2.2 Service primitives

The service parameters for this service are shown in Table 102.

Table 102 – Identify service

Parameter	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
InvokeID	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
InvokeID			M	M(=)
Service status			M	M(=)
DL-address			M	M(=)
MAC address			M	M(=)
Port information			M	M(=)
Protocol version			M	M(=)
Device type			M	M(=)
Device description			M	M(=)
Hardware-Version			M	M(=)
Serial number			M	M(=)
Software-Version			M	M(=)
Software-Date			M	M(=)
Vendor ID			M	M(=)
Product Code			M	M(=)
Result(-)			S	S(=)
AREP			M(=)	M(=)
InvokeID			M(=)	M(=)
Service status			M	M(=)
Status code			M	M(=)

Argument
The argument conveys the service specific parameters of the service request.

AREP
This parameter contains information sufficient for local identification of the AREP to be used to convey the service. This parameter may use a key attribute of the AREP to identify the application relationship. When an AREP supports multiple contexts simultaneously, the AREP parameter is extended to identify the context as well as the AREP.

Service status
This parameter provides information on the result of service execution. It is returned in all confirmed service response primitives (+ and -). It is composed of the following elements.

Status code
This parameter indicates whether or not the service was processed successfully. If an error occurred, it indicates the type of error. Available status codes are listed in 9.10.

Result(+)
This selection type parameter indicates that the service request succeeded.

DL-address
See 8.3.6.4.1

MAC address
See 8.3.6.4.7

Port information
See 8.3.6.4.8

Protocol version
See 8.3.6.4.9

Device type
See 8.3.6.4.10

Device description
See 8.3.6.4.11

Hardware-Version
This parameter contains the hardware version of the device.

Serial number
This parameter contains the serial number of the device.

Software-Version
This parameter contains the software version of the device.

Software-Date
This parameter contains the software date of the device.

Vendor ID
This parameter contains the vendor ID of the device.

Product code
This parameter contains the product code of the device.

Result(-)
This selection type parameter indicates that the service request failed. The detailed error information is appended in the status code field.

8.7.1.1.3.2.3 Service procedure

This service procedure is a sequence of two successive confirmed services (as specified in 9.5.1.6) in opposite directions.

8.7.1.1.3.3 STATUS service

8.7.1.1.3.3.1 Service overview

This confirmed service may be used to request the network-visible state from the AP.

8.7.1.1.3.3.2 Service primitives

The service parameters for this service are shown in Table 103.

Table 103 – Status service

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
InvokeID	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
InvokeID			M	M(=)
Service status			M	M(=)
Device flags			M	M(=)
Device state			M	M(=)
tx_cnt_normal			M	M(=)
tx_cnt_all			M	M(=)
rx_cnt_normal			M	M(=)
rx_cnt_all			M	M(=)
relay_cnt_normal			M	M(=)
relay_cnt_all			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
InvokeID			M	M(=)
Service status			M	M(=)
Status code			M	M(=)

Argument
The argument conveys the service specific parameters of the service request.

AREP
This parameter contains information sufficient for local identification of the AREP to be used to convey the service. This parameter may use a key attribute of the AREP to identify the application relationship. When an AREP supports multiple contexts simultaneously, the AREP parameter is extended to identify the context as well as the AREP.

Service status
This parameter provides information on the result of service execution. It is returned in all confirmed service response primitives (+ and -). It is composed of the following elements.

Status code
This parameter indicates whether or not the service was processed successfully. If an error occurred, it indicates the type of error. For some errors, further identification of the error may be provided in the Extended Status parameter below. Available status codes are listed in 9.10.

Result(+)
This selection type parameter indicates that the service request succeeded.

Device flags
See 8.3.6.4.2

Device state
See 8.3.6.4.3

tx_cnt_normal
This parameter contains the value of the transmitted count of normal packets.

tx_cnt_all
This parameter contains the value of the transmitted count of both error packets and normal packets.

rx_cnt_normal
This parameter contains the value of the received count of normal packets.

rx_cnt_all
This parameter contains the value of the received count of both error packets and normal packets.

relay_cnt_normal
This parameter contains the value of the relayed count of normal packets.

relay_cnt_all
This parameter contains the value of the relayed count of both error packets and normal packets.

Result(-)
This selection type parameter indicates that the service request failed. The detailed error information is appended in the status code field.

8.7.1.1.3.3.3 Service procedure

This service procedure is a sequence of two successive confirmed services (as specified in 9.5.1.6) in opposite directions.

8.7.1.2 Service data object ASE

8.7.1.2.1 Overview

For all transfer types, the client takes the initiative for a transfer. The owner of the accessed object dictionary is the server of the service data object (SDO). Either the client or the server can take the initiative to abort the transfer of a SDO. All commands are confirmed. The remote result parameter indicates the success of the request. In case of a failure, an abort transfer request must be executed.

8.7.1.2.2 SDO class PAS

8.7.1.2.2.1 Formal model

The AP class specifies the attributes and services defined for application processes. Its parent class “TOP” indicates the top of the FAL class tree.

ASE:			**SDO ASE**
CLASS:			**SDO**
CLASS ID:			**not used**
PARENT CLASS:			**TOP**
ATTRIBUTES:			
1	(m)	Key Attribute:	Object identifier
2	(m)	Attribute:	Size
3	(m)	Attribute:	Data type
4	(m)	Attribute:	Access right
SERVICES:			
1	(m)	OpsService:	Read
2	(m)	OpsService:	Write

8.7.1.2.2.2 Attributes

Object identifier
This attribute contains the numeric identifier of the object to be accessed.

Size
This attribute specifies the actual size of the object in octets.

Data type
This attribute contains the numeric identifier of the data type.

Access right
This attribute contains the type of access defined for the object, as shown in Table 104.

Table 104 – Access rights for object

Name	Signification
R	Right to read for the registered password
W	Right to write for the registered password
U	Right to use the registered password

8.7.1.2.3 SDO ASE service PAS

8.7.1.2.3.1 Supported services

This subsection contains the definition of services that are unique to this ASE. The services defined for this ASE are READ and WRITE.

8.7.1.2.3.2 READ service

8.7.1.2.3.2.1 Service overview

This confirmed read service may be used to read the value of an SDO.

8.7.1.2.3.2.2 Service primitives

The service parameters for this service are shown in Table 105.

Table 105 – Read service

Parameter	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
InvokeID	M	M(=)		
Object List Count	M	M(=)		
Object List	M	M(=)		
Object-identifier	M	M(=)		
Data type	M	M(=)		
Offset	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)

Parameter	Req	Ind	Rsp	Cnf
Invoke ID			M	M(=)
Service status			M	M(=)
OD List Count			M	M(=)
Object List			M	M(=)
Object-identifier			M	M(=)
Data type			M	M(=)
Offset			M	M(=)
Length			M	M(=)
Data			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Service status			M	M(=)
Status code			M	M(=)

Argument
The argument conveys the service specific parameters of the service request.

AREP
This parameter contains information sufficient for local identification of the AREP to be used to convey the service. This parameter may use a key attribute of the AREP to identify the application relationship. When an AREP supports multiple contexts simultaneously, the AREP parameter is extended to identify the context as well as the AREP.

InvokeID
This parameter identifies this invocation of the service. InvokeID is used to associate service requests with responses. Therefore, no two outstanding service invocations can be identified by the same InvokeID value.

Object List Count
This parameter indicates the number of objects listed in the service request. Each object is designated with object list arguments described below.

Object List
This parameter contains a stream of information containing the actual list of objects to be read.

Object identifier

This parameter identifies the entry of the target object to be read by this service.

Data type

This parameter identifies the data type.

Offset

This parameter identifies the local offset address of the data to be read from the object.

Length

The parameter indicates the number of octets to be read.

Service status

This parameter provides information on the result of service execution. It is returned in all confirmed service response primitives (+ and -). It is composed of the following elements.

Status code

This parameter indicates whether or not the service was processed successfully. If an error occurred, it indicates the type of error. Available status codes are listed in Chapter 10.

Result(+)
This selection type parameter indicates that the service request succeeded.

Data

This parameter contains the value of the object that has been read and consists of the number of octets indicated in the length of the response primitive.

Result(-)
This selection type parameter indicates that the service request failed.

8.7.1.2.3.2.3 Service procedure

This service procedure is a sequence of two successive confirmed services (as specified in 9.5.1.6) in opposite directions.

8.7.1.2.3.3 Write service

8.7.1.2.3.3.1 Service overview

This confirmed write service may be used to write the value of an SDO.

8.7.1.2.3.3.2 Service primitives

The service parameters for this service are shown in Table 106.

Table 106 – Write service

Parameter	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
InvokeID	M	M(=)		
Object List Count	M	M(=)		
Object List	M	M(=)		
Object identifier	M	M(=)		
Data type	M	M(=)		
Offset	M	M(=)		
Length	M	M(=)		
Data	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
InvokeID			M	M(=)
Service status			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
InvokeID			M	M(=)
Service status			M	M(=)
Status code			M	M(=)

Argument
The argument conveys the service specific parameters of the service request.

AREP
This parameter contains information sufficient for local identification of the AREP to be used to convey the service. This parameter may use a key attribute of the AREP to identify the application relationship. When an AREP supports multiple contexts simultaneously, the AREP parameter is extended to identify the context as well as the AREP.

InvokeID
This parameter identifies this invocation of the service. InvokeID is used to associate service requests with responses. Therefore, no two outstanding service invocations can be identified by the same InvokeID value.

Object List Count
This parameter identifies the number of objects listed in the service request. Each object is designated with the object list argument described below.

Object List
This parameter contains a stream of information containing the actual list of objects to be written.

Object identifier

This parameter identifies the entry of the target object to be written by this service.

Data type

This parameter identifies the data type.

Offset

This parameter identifies the local offset address of the data to be written to the object.

Length

The parameter indicates the number of octets to be written.

Data

This parameter contains the value of the object that has been written, and consists of the number of octets indicated in the length of the request primitive.

Service status

This parameter provides information on the result of service execution. It is returned in all confirmed service response primitives (+ and -). It is composed of the following elements.

Status code

This parameter indicates whether or not the service was processed successfully. If an error occurred, it indicates the type of error. Available status codes are listed in Chapter 10.

Result(+)
This selection type parameter indicates that the service request succeeded.

Result(-)
This selection type parameter indicates that the service request failed.

8.7.1.2.3.3.3 Service procedure

This service procedure is a sequence of two successive confirmed services (as specified in 9.5.1.6) in opposite directions.

8.7.1.3 Process data object ASE

8.7.1.3.1 Overview

The time-critical data are transferred by the process data objects (PDO) using the publisher-subscriber communication model. The periodically scheduled timer-based (TB) or change-of-state (COS) event-triggered PDO data are multicast to subscribers using the unconfirmed send service.

From the point of view of the device, there are two types of PDO usage: data transmission and data reception. The transmission PDOs (TPDOs) and reception PDOs (RPDOs) shall be distinguished from each other. Devices supporting TPDOs are called PDO publishers and devices that are able to receive PDOs are called PDO subscribers.

The numbers of supported channels between AREPs may be preconfigured by the application, and the method to configure the relationships between AREPs outside the scope of this PAS.

8.7.1.3.2 PDO class PAS

8.7.1.3.2.1 Formal model

ASE:			**PDO ASE**
CLASS:			**PDO**
CLASS ID:			**not used**
PARENT CLASS:			**TOP**
ATTRIBUTES:			
1	(m)	Attribute:	list of RPDO channels
1.1	(m)	Attribute:	channel info
1.2	(m)	Attribute:	list of mapped object entries
1.2.1	(m)	Attribute:	object identifier
1.2.2	(m)	Attribute:	data type
1.2.3	(m)	Attribute:	length
2	(m)	Attribute:	list of TPDO channels
2.1	(m)	Attribute:	channel info
2.2	(m)	Attribute:	list of mapped object entries
2.2.1	(m)	Attribute:	object identifier
2.2.2	(m)	Attribute:	data type
2.2.3	(m)	Attribute:	length
SERVICES:			
1	(m)	OpsService:	TB-transfer
2	(m)	OpsService:	COS-transfer

8.7.1.3.2.2 Attributes

List of RPDO channels
This attribute specifies the number of logical channels to receive TB/COS-transfer services. The maximum permissible value for the RPDO channel is 254.

Channel info
This attribute specifies the AREP relationship between publisher and subscriber.

List of mapped OD entries
The following attributes specify the OD entries to be mapped to the PDO of the respective channel. A PDO consists of up to 256 mapped object entries.

Object identifier

This parameter identifies the entry of the target object to be written by this service.

Data type

This parameter identifies the data type.

Length
This attribute provides the length of the mapped object in octets.

List of TPDO channels
The following attributes specify the logical channels for outgoing PDOs.

All the following attributes correspond to the equivalents in “list-of-RPDO-channels” and are therefore not repeated here.

8.7.1.3.2.3 Services

TB-transfer
This service is used to transfer the TB PDO between devices.

COS-transfer
This service is used to transfer the COS PDO between devices.

8.7.1.3.3 PDO ASE service PAS

8.7.1.3.3.1 Supported services

This subsection contains the definition of services that are unique to this ASE. The services defined for this ASE are TB-transfer and COS-transfer.

8.7.1.3.3.2 TB-transfer

8.7.1.3.3.2.1 Service overview

This service is used to transfer process data objects between devices. At the requesting device, the outgoing TB is composed of the appropriate TB attributes. At the receiving device, the incoming TB is handled according to the TB attributes.

8.7.1.3.3.2.2 Service primitives

The service parameters for this service are shown in Table 107.

Table 107 – TB-transfer

Parameter name	Req	Ind
Argument	M	M(=)
AREP	M	M(=)
TB PDO	M	M(=)

Argument
The argument conveys the service specific parameters of the service request.

TB PDO
This parameter contains the TB PDO data.

8.7.1.3.3.2.3 Service procedure

This service is performed either as one unconfirmed service or as a sequence of two successive unconfirmed services in opposite directions.

8.7.1.3.3.3 COS-transfer

8.7.1.3.3.3.1 Service overview

This service is used to transfer process data objects between devices. At the requesting device, the outgoing COS is composed of the appropriate COS attributes. At the receiving device, the incoming COS is handled according to the COS attributes.

8.7.1.3.3.3.2 Service primitives

The service parameters for this service are shown in Table 108.

Table 108 – COS-transfer

Parameter	Req	Ind
Argument	M	M(=)
AREP	M	M(=)
COS PDO	M	M(=)

Argument
The argument conveys the service specific parameters of the service request.

COS PDO
This parameter contains the COS data.

8.7.1.3.3.3.3 Service procedure

This service is performed either as one unconfirmed service or as a sequence of two successive unconfirmed services in opposite directions.

8.7.1.4 Application relationship ASE

8.7.1.4.1 Overview

8.7.1.4.1.1 General

In a distributed system, application processes communicate with each other by exchanging application layer messages across well-defined application layer communications channels.

These communication channels are modeled in the FAL as ARs.

ARs are responsible for conveying messages between applications according to specific communications characteristics required by time-critical systems. Different combinations of these characteristics lead to the definition of different types of ARs. The characteristics of ARs are defined formally as attributes of AR endpoint classes.

The messages that are conveyed by ARs are FAL service requests and responses. Each is submitted to the AR ASE for transfer by an FAL ASE that represents the class of the APO being accessed. Figure 60 illustrates this concept.

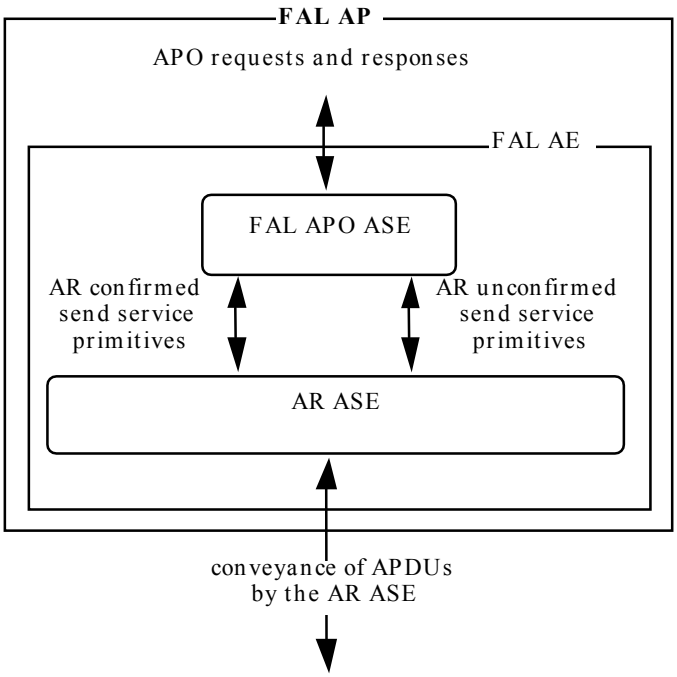


Figure 60 – The AR ASE conveys APDUs between APs

Depending on the type of AR, APDUs may be sent to one or more destination application processes connected by the AR. Other characteristics of the AR determine how APDUs are to be transferred. These characteristics are described below.

8.7.1.4.1.2 Endpoint context

Each AP involved in an AR contains an endpoint of the AR. Each AR endpoint is defined within the AE of the AP. Its definition defines an AR when combined with the definitions of the other endpoints. To ensure communications compatibility among or between endpoints, each endpoint definition contains a set of compatibility-related characteristics. These characteristics must be configured appropriately for each endpoint for the AR to operate properly.

Endpoint definitions also contain a set of characteristics that describe the operation of the AR. These characteristics define the context of the endpoint when combined with those used to specify compatibility. The endpoint context is used by the AR ASE to manage the operation of the endpoint and the conveyance of APDUs. The characteristics that comprise the endpoint context are described below.

8.7.1.4.1.2.1 Endpoint role

The role of an AREP determines the permissible behaviour of an AP at the AREP. An AREP may have the role of client, server, peer (client and/or server), publisher, or subscriber.

Tables 109 and 110 summarize the characteristics and combinations of each of the AREP roles.

Table 109 – Conveyance of service primitives by AREP role

	Client	Server	Peer	Publisher	Subscriber
CS Req	X		X		
CS Rsp		X	X		
UCS Req				X	

Table 110 – Valid combinations of AREP roles involved in an AR

	Client	Server	Peer	Publisher	Subscriber
Client		X	X		
Server	X		X		
Peer	X	X	X		
Publisher					X
Subscriber				X	

8.7.1.4.1.2.2 Cardinality

From the point of view of a client or publisher endpoint, the cardinality of an AR specifies how many remote application processes are involved in an AR. Cardinality is never expressed from the viewpoint of a server or a subscriber.

When expressed from the viewpoint of a client or peer endpoint, ARs are always 1:1. Clients are never capable of issuing a request and waiting for responses from multiple servers.

When expressed from the viewpoint of a publisher endpoint, ARs support multiple subscribers. These ARs are 1:many. Such ARs provide for communications between one application and a group of one or more applications. They are often referred to as multicast.

8.7.1.4.1.3 Conveyance model

8.7.1.4.1.3.1 General

The conveyance model defines how APDUs are sent between endpoints of an AR. The following three characteristics are used to define these transfers.

a) Conveyance paths

b) Trigger policy

c) Conveyance policy

8.7.1.4.1.3.2 Conveyance paths

The purpose of AR ASEs is to transfer information between AR endpoints. This information transfer occurs over the conveyance paths of an AR. A conveyance path represents a one-way communication path used by an endpoint for input or output.

The endpoint is configured with either one or two conveyance paths to support the role of the application process. Endpoints that only send or only receive are configured with either a send or receive conveyance path, respectively, and those that do both are configured with both. ARs with a single conveyance path are called unidirectional, while those with two conveyance paths are called bidirectional.

Unidirectional ARs are capable of conveying service requests only. To convey service responses, a bidirectional AR is necessary. Therefore, unidirectional ARs support the transfer of unconfirmed services in one direction only, while bidirectional ARs support the transfer of unconfirmed and confirmed services initiated by only one endpoint, or by both endpoints.

8.7.1.4.1.3.3 Trigger policy

Trigger policy indicates when APDUs are transmitted by the DLL over the network.

The first type is referred to as user-triggered. User-triggered AREPs submit FAL APDUs to the DLL for transmission at the earliest opportunity.

The second type is referred to as network-scheduled. Network-scheduled AREPs submit FAL APDUs to the DLL for transmission according to a schedule determined by management.

This network scheduling mechanism is cyclic.

8.7.1.4.1.3.4 Conveyance policy

Conveyance policy indicates whether APDUs are transferred according to a buffer model or a queue model. These models describe the method of conveying APDUs from sender to receiver.

Buffered ARs contain conveyance paths that have a single buffer at each endpoint. Updates to the source buffer are conveyed to the destination buffer according to the trigger policy of the AR. Updates to either buffer overwrite the contents with new data. In buffered ARs, unconveyed or undelivered data are lost once they are overwritten. Data in a buffer may be read multiple times without the contents being destroyed.

Queued ARs contain conveyance paths that are represented as a queue between endpoints. Queued ARs convey data using a FIFO queue. Queued ARs are not overwritten; new entries are queued until they can be conveyed and delivered.

If a queue is full, new messages will not be added.

NOTE The AR conveyance services are described abstractly in such a way that they are capable of being implemented to operate using buffers or queues. These services may be implemented in a number of ways. For example, they may be implemented in such a way that the capability is provided to load the buffer/queue, and subsequently post it for transfer by the underlying DLL. Alternatively, these services may be implemented such that these capabilities are combined so that the buffer/queue may be loaded and transferred in a single request. On the receiving side, these services may be implemented by delivering the data when received, or by indicating receipt and allowing the user to retrieve the data in a separate operation. Another option is to require the user to detect that the buffer or queue has been updated.

8.7.1.4.1.4 Underlying communications services

8.7.1.4.1.4.1 General

The AR ASE conveys FAL APDUs using the capabilities of the underlying DLL. Several characteristics are used to describe these capabilities. This subclause provides a description of each. These characteristics are specific to the data link mapping defined in 9.10. Their precise specification can be found there.

8.7.1.4.1.4.2 Connection-oriented services

The underlying layer does not support AR endpoints by providing connection-oriented services. Thus, connections have to be preconfigured by means of the application layer.

8.7.1.4.1.4.3 Buffered and queued services

The underlying layer may support AR endpoints by providing buffered or queued services. These services can be used to implement buffers or queues required by certain classes of endpoint.

8.7.1.4.1.4.4 Cyclic and acyclic transfers

The underlying layer may support AR endpoints by providing cyclic or acyclic services.

8.7.1.4.1.5 AR establishment

For an AR endpoint to be used by an application process, the corresponding AR must be active. When an AR is activated, it is referred to as being “established.”

ARs can be pre-established. Pre-established means that the AE that maintains the endpoint context is created before the AP is connected to the network. In this case, communications among the applications involved in the AR may take place without first having to establish the AR explicitly.

8.7.1.4.1.6 Application relationship classes

AREPs are defined with a combination of characteristics to form different classes of ARs.

8.7.1.4.2 Application relationship class PAS

8.7.1.4.2.1 Formal model

The formal AR endpoint model defines the characteristics common to all AR endpoints. This class is not capable of being instantiated. It is present only for the inheritance of its attributes and services by its subclasses, each specified in a separate section of this PAS.All AR endpoint attributes are accessible through system management.

FAL ASE:			**AR ASE**
CLASS:			**AR ENDPOINT**
CLASS ID:			**not used**
PARENT CLASS:			**TOP**
ATTRIBUTES:			
1	(m)	Attribute:	FAL Revision
2	(m)	Attribute:	Dedicated (TRUE, FALSE)
3	(m)	Attribute:	Cardinality (1:1, 1:many)
4	(m)	Attribute:	Conveyance policy (queued, buffered)
5	(m)	Attribute:	Conveyance path (unidirectional, bidirectional)
6	(m)	Attribute:	Trigger policy (user-triggered, network-scheduled)
7	(o)	Attribute:	Transfer Syntax
SERVICES:			
1	(o)	OpsService:	AR-unconfirmed send
2	(o)	OpsService:	AR-Confirmed Send

8.7.1.4.2.2 Attributes

FAL revision
This specifies the revision level of the FAL protocol used by this endpoint. The revision level is in the AR header of all FAL PDUs transmitted.

Dedicated
This attribute specifies whether the endpoint is dedicated or not. When TRUE, the services of the AR ASE are accessed directly by the FAL-user.

Cardinality
This attribute specifies the cardinality of the AR described in 9.5.1.3.7.

Conveyance policy
This attribute specifies the conveyance policy of the AR described in 9.5.1.3.7.

Conveyance path
This attribute specifies the conveyance path of the AR described in 9.5.1.3.7.

Trigger policy
This attribute specifies the trigger policy of the AR described in 9.5.1.3.7.

Transfer syntax
This optional attribute identifies the encoding rules to be used on the AR. When not present, the default FAL transfer syntax of this PAS is used.

8.7.1.4.2.3 Services

All services defined for this class are optional. When an instance of the class is defined, at least one shall be selected.

AR-unconfirmed send
This optional service is used to send an unconfirmed service.

AR-confirmed send
This optional service is used to send a confirmed service.

8.7.1.4.3 Application relationship ASE service specification

8.7.1.4.3.1 Supported services

This section contains the definitions of services that are unique to this ASE. The services defined for this ASE are AR-unconfirmed send and AR-confirmed send.

The AR-confirmed send service contains the FAL PDU body as part of the Result parameter in the response and confirmation primitives. The FAL PDU body may contain either a positive or negative response returned by the FAL-user transparently to the AR ASE. Therefore, these services have a single Result parameter instead of the separate Result(+) and Result(-) parameters commonly used to convey the positive and negative responses returned by the FAL-user.

8.7.1.4.3.2 AR-unconfirmed send service

8.7.1.4.3.2.1 Service overview

This service is used to send AR-unconfirmed send request APDUs for FAL APO ASEs. The AR-unconfirmed send service may be requested at either endpoint of a 1:1 bidirectional AR, at the server endpoint of a 1:1 unidirectional AR, or at the publisher endpoint of a 1:many AR.

8.7.1.4.3.2.2 Service primitives

The service parameters for each primitive are shown in Table 111.

Table 111 – AR-unconfirmed send

Parameter	Req	Ind
Argument	M	M(=)
AREP	M	M(=)
Remote DL-address	M	M(=)
FAL Service Type	M	M(=)
FAL APDU Body	M	M(=)

Remote DL-address
This parameter contains the destination DLSAP address in the request and the source DLSAP address in the indication.

FAL service type
This parameter contains the type of service being conveyed.

FAL APDU body
This parameter contains the service-dependent body for the APDU.

8.7.1.4.3.2.3 Service procedure

The AR-unconfirmed send service is a service that operates through a queue.

The requesting FAL ASE submits an AR-unconfirmed send request primitive to its AR ASE. The AR ASE builds an AR-unconfirmed send request APDU.

The AR ASE queues the APDU for submission to the lower layer.

If the AREP is user-triggered, the AR ASE immediately requests the lower layer to transfer the APDU. If the AR is network-scheduled, the AR ASE requests the DLL to transfer the data at the scheduled time. The data link mapping indicates how the AR ASE coordinates its requests to transmit the data with the DLL.

Upon receipt of the AR-unconfirmed send request APDU, the receiving AR ASE delivers an AR-unconfirmed send indication primitive to the appropriate FAL ASE as indicated by the FAL service type parameter.

8.7.1.4.3.3 AR-confirmed send service

8.7.1.4.3.3.1 Service overview

This service is used to send confirmed request and response APDUs for FAL APO ASEs. The AR-confirmed send service may be requested at client and peer endpoints of 1:1 bidirectional ARs.

8.7.1.4.3.3.2 Service primitives

The service parameters for each primitive are shown in Table 112.

Table 112 – AR-confirmed send

Parameter	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
InvokeID	M	M(=)		
Server DL-address	M	M(=)		
FAL service type	M	M(=)		
FAL APDU body	M	M(=)		
Result			M	M(=)
AREP			M	M(=)
InvokeID			M	M(=)
Client DL-address			M	M(=)
FAL service type			M	M(=)
FAL APDU body			M	M(=)

Server DL-address
This parameter contains the DL-address of the server.

FAL service type
This parameter contains the type of service being conveyed.

FAL APDU body
This parameter contains the service-dependent body for the APDU.

Result
This parameter indicates that the service request either succeeded or failed.

Client DL-address
This parameter contains the DL-address of the client.

Service procedure

The AR-confirmed send service is a service that operates through a queue.

The requesting FAL ASE submits an AR-confirmed send request primitive to its AR ASE. The AR ASE creates a transaction state machine to control the invocation of the service.

The AR ASE builds an AR-confirmed send request APDU and queues it for submission to the lower layer. Then, the AR ASE immediately requests the lower layer to transfer the APDU.

Upon receipt of the AR-confirmed send request APDU, the receiving AR ASE delivers an AR-confirmed send indication primitive to the appropriate FAL ASE as indicated by the FAL service type parameter.

The responding FAL ASE submits a confirmed send response primitive to its AR ASE. The AR ASE builds a confirmed send response APDU.

The AR ASE queues the APDU for submission to the lower layer. Then, the AR ASE immediately requests the lower layer to transfer the APDU.

Upon receipt of the confirmed send response APDU, the receiving AR ASE uses the InvokeID contained in the response APDU to associate the response with the appropriate request and cancel the associated transaction state machine. The AR ASE delivers an AR-confirmed send confirmation primitive to the requesting FAL ASE.

If the timer expires before the sending AR ASE receives the response APDU, the AR ASE cancels the associated transaction state machine and delivers an AR-confirmed send confirmation(-) primitive to the requesting FAL ASE.

8.7.2 ARs

8.7.2.1 Point-to-point user-triggered confirmed client/server AREP (PTC-AR)

8.7.2.1.1 Class overview

This class is defined to support the on-demand exchange of confirmed services between two or more application processes. It uses connectionless-mode data link services for the exchanges. The behaviour of this class is described below.

An AR ASE user submits a request APDU as an AR ASE service data unit to its AREP. The AREP sending the request APDU queues it to its underlying layer for transfer at the next available opportunity.

The AREP receiving the request APDU from its underlying layer queues it for delivery to its AR ASE user in the order in which it was received.

For a confirmed service request, the AREP receiving the request APDU accepts the corresponding response APDU from its AR ASE user and queues it to the underlying layer for transfer.

The AREP that issued the request APDU receives the response APDU from its underlying layer and queues it for delivery to its AR ASE user in the order in which it was received.

The following summarizes the characteristics of this AREP class.

Roles	Client
	Server

Cardinality	1:1
Conveyance paths	Bidirectional
Trigger policy	User-triggered
Conveyance policy	Queued

Formal model

FAL ASE:			**AR ASE**
CLASS:			**PTC-AR**
CLASS ID:			**not used**
PARENT CLASS:			**AR ENDPOINT**
ATTRIBUTES:			
1	(m)	Attribute:	Role (CLIENT, SERVER)
2	(m)	Attribute:	AREP State
3	(m)	Attribute:	Transmit DL Mapping Reference
4	(m)	Attribute:	Receive DL Mapping Reference
SERVICES:			
1	(m)	OpsService:	Confirmed Send

8.7.2.1.2 Network management attributes

Role
This attribute specifies the role of the AREP. Valid values are as follows.

a) Client. Endpoints of this type issue confirmed service request-APDUs to servers and receive confirmed service response-APDUs.

b) Server. Endpoints of this type receive confirmed service request-APDUs from clients and issue confirmed service response-APDUs to them.

AREP-state
This attribute specifies the state of the AREP. The values for this attribute are specified in 9.10.

Transmit data link mapping reference
This attribute provides a reference to the underlying DLL mapping for the transmit conveyance path for this AREP. Data link mappings for the DLL are specified in Chapter 10.

Receive data link mapping reference
This attribute provides a reference to the underlying DLL mapping for the receive conveyance path for this AREP. Data link mappings for the DLL are specified in 9.10.

8.7.2.1.3 Services

Confirmed send
This optional service is used to send a confirmed service on an AR.

8.7.2.2 Multipoint network-scheduled unconfirmed publisher-subscriber AREP (MSU-AR)

8.7.2.2.1 Class overview

This class is defined to support the push model for scheduled and buffered distribution of unconfirmed services to one or more application processes.

The behaviour of this type of AR can be described as follows.

An AR ASE user submits a request APDU as an AR ASE service data unit to its AREP for distribution. The sending AREP writes the APDU into the internal buffer, overwriting the existing buffer contents.

The AREP transfers the buffer contents at the next scheduled transfer opportunity.

If the AREP receives another APDU before the buffer contents are transmitted, the buffer contents will be overwritten with the new APDU, and the previous APDU will be lost. When the buffer contents are transmitted, the AR ASE notifies the user of transmission.

At the receiving endpoint, the APDU is received from the network and is written immediately into the buffer, overwriting the existing contents of the buffer. The endpoint notifies the user that the APDU has arrived and delivers it to the user according to the local user interface. If the APDU has not been delivered before the next APDU arrives, it will be overwritten by the next APDU and lost.

An FAL-user receiving the buffered transmission may request to receive the currently buffered APDU later.

The following summarizes the characteristics of this AREP class.

Roles	Publisher
	Subscriber
Cardinality	1:n
Conveyance paths	Unidirectional
Trigger policy	Network-scheduled
Conveyance policy	Buffered

Formal model

FAL ASE:			**AR ASE**
CLASS:			**MSU-AR**
CLASS ID:			**not used**
PARENT CLASS:			**AR ENDPOINT**
ATTRIBUTES:			
1	(m)	Attribute:	Role (PUBLISHER, SUBSCRIBER)
2	(m)	Attribute:	AREP State
3	(m)	Attribute:	DL Mapping Reference
SERVICES:			
1	(m)	OpsService:	Unconfirmed Send

8.7.2.2.2 Network management attributes

Role
This attribute specifies the role of the AREP. Valid values are as follows.

a) Push-publisher. Endpoints of this type publish their data issuing unconfirmed service request-APDUs.

b) Push-subscriber. Endpoints of this type receive data from service request-APDUs released by a push-publisher AREP.

AREP-state
This attribute specifies the state of the AREP. The values for this attribute are specified in 9.10.

DL-mapping-reference
For publisher AREPs, this attribute specifies the mapping to the transmit conveyance path. For subscriber AREPs, this attribute specifies the mapping to the receive conveyance path. Data link mapping attributes for the DLL are specified in 9.10.

8.7.2.2.3 Services

Unconfirmed Send
This optional service is used to send an unconfirmed service on an AR.

8.7.2.3 Multipoint user-triggered unconfirmed publisher-subscriber AREP (MTU-AR)

8.7.2.3.1 Class overview

This class is defined to support the on-demand queued distribution of unconfirmed services to one or more application processes. The behaviour of this class is described as follows.

An AR ASE user submits a request APDU as an AR ASE service data unit to its AREP. The AREP sending the request APDU queues it to its underlying layer for transfer at the next available opportunity.

The AREP receiving the request APDU from its underlying layer queues it for delivery to its AR ASE user in the order in which it was received.

The following summarizes the characteristics of this AREP class.

Roles	Publisher
	Subscriber
Cardinality	1:n
Conveyance paths	Unidirectional
Trigger policy	User-triggered
Conveyance policy	Queued

Formal model

FAL ASE:			**AR ASE**
CLASS:			**MTU-AR**
CLASS ID:			**not used**
PARENT CLASS:			**AR ENDPOINT**
ATTRIBUTES:			
1	(m)	Attribute:	Role (PUBLISHER, SUBSCRIBER)
2	(m)	Attribute:	AREP State
3	(m)	Attribute:	DL Mapping Reference
SERVICES:			
1	(m)	OpsService:	Unconfirmed Send

8.7.2.3.2 Network management attributes

Role
This attribute specifies the role of the AREP. Valid values are as follows.

- Push-publisher: Endpoints of this type publish their data issuing unconfirmed service request-APDUs.
- Push-subscriber: Endpoints of this type receive data from service request-APDUs released by a push-publisher AREP.

AREP-state
This attribute specifies the state of the AREP. The values for this attribute are specified in 9.10.

DL-mapping-reference
For publisher AREPs, this attribute specifies the mapping to the transmit conveyance path. For subscriber AREPs, this attribute specifies the mapping to the receive conveyance path. Data link mapping attributes for the DLL are specified in 9.10.

8.7.2.3.3 Services

Unconfirmed Send
This optional service is used to send an unconfirmed service on an AR.

8.7.3 Summary of FAL classes

This section contains a summary of the defined FAL Classes. The Class ID values have been assigned to be compatible with existing standards. Table 113 provides a summary of the FAL classes.

Table 113 – FAL class summary

FAL ASE	Class
Application process	AP
Data type	Fixed-length and String
Process data object	PDO
Service data object	SDO
Application relationship	AREP
	PTC-AREP
	MSU-AREP
	MTU-AREP

8.7.4 Permitted FAL services by AREP role

Table 114 defines the valid combinations of services and AREP roles. The Unc and Cnf columns indicate whether the service listed in the left-hand column is unconfirmed (Unc) or confirmed (Cnf).

Table 114 – Services by AREP role

	Unc	Cnf	Client		Server		Push Publisher		Push Subscriber	
FAL Services			**req**	**rcv**	**req**	**rcv**	**req**	**rcv**	**req**	**rcv**
AP ASE										
Identify		x	x			x				
Status		x	x			x				
PDO ASE										
TB-transfer	x						x			x
COS-transfer	x						x			x
SDO ASE										
Read		x	x			x				
Write		x	x			x				
AR ASE										
PTC-AR		x	x			x				
MSU-AR	x						x			x
MTU-AR	x						x			x

9 Application layer protocol specification

9.1 Introduction

The application protocol provides the application service using the services available from the data link layer (DLL) or other immediately lower layer. The primary aim of this specification is to provide a set of rules for communication expressed in terms of the procedures to be carried out by peer application entities (AEs) at the time of communication. These rules for communication are intended to provide a sound basis for development to serve a variety of purposes, including

a) as a guide for implementers and designers;

b) for use in the testing and procurement of equipment;

c) as part of an agreement for the admission of systems into the open systems environment;

d) as a refinement to the understanding of time-critical communications within OSI.

In particular, this specification is concerned with the communication and interworking of sensors, effectors, and other automation devices. By using this PAS together with other standards positioned within the OSI or fieldbus reference models, systems that are otherwise incompatible may work together in any combination.

9.2 Scope

9.2.1 General

This PAS is one of a series produced to facilitate the interconnection of automation system components. It is related to other standards in the set as defined by the three-layer fieldbus reference model described in IEC/TR 61158-1:2007.

This section contains material specific to the RAPIEnet communication protocol.

9.2.2 Overview

The fieldbus application layer (FAL) provides user programmes with a means to access the fieldbus communication environment. In this respect, the FAL can be viewed as a window between corresponding application programmes.

This specification provides common elements for basic time-critical and non-time-critical messaging communications between application programmes in an automation environment, as well as material specific to RAPIEnet. The term “time-critical” is used to represent the presence of a time-window, within which one or more specified actions must to be completed with some defined level of certainty. Failure to complete specified actions within the required time risks the failure of the applications requesting the actions, with attendant risk to equipment, plant, and possibly human life.

This specification defines interactions between remote applications. It also defines the externally visible behaviour provided by the RAPIEnet application layer in terms of

a) the formal abstract syntax defining the application layer protocol data units (APDUs) conveyed between communicating application entities;

b) the transfer syntax defining encoding rules that are applied to the APDUs;

c) the application context state machine defining the application service behaviour visible between communicating application entities;

d) the application relationship state machines defining the communication behaviour visible between communicating application entities.

The purpose of this clause is to

a) describe the wire-representation of the service primitives defined in Clause 9;

b) describe the externally visible behaviour associated with their transfer.

This specification defines the protocol of the RAPIEnet application layer in conformance with the OSI Basic Reference Model (ISO/IEC 7498) and the OSI application layer structure (ISO/IEC 9545).

9.2.3 Specifications

The principal objective of this clause is to specify the syntax and behaviour of the application layer protocol that conveys the RAPIEnet application layer services.

A secondary objective is to provide migration paths from previously existing industrial communications protocols.

9.2.4 Conformance

This specification does not restrict individual implementations or products, nor does it constrain the implementations of application layer entities in industrial automation systems. Conformance is achieved through implementation of this application layer protocol specification.

9.3 Normative references

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

ISO/IEC 7498-1, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*

ISO/IEC 7498-3, *Information technology – Open Systems Interconnection – Basic Reference Model: Naming and addressing*

ISO/IEC 8822, *Information technology – Open Systems Interconnection – Presentation service definition*

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

ISO/IEC 9545, *Information technology – Open Systems Interconnection – Application Layer structure*

9.4 Terms, definitions, symbols, abbreviations, and conventions

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

9.4.1 ISO/IEC 8824 terms

For the purposes of this document, the following terms as defined in ISO/IEC 8824 apply.

9.4.1.1 any type

9.4.1.2 bitstring type

9.4.1.3 Boolean type

9.4.1.4 choice type

9.4.1.5 false

9.4.1.6 integer type

9.4.1.7 module

9.4.1.8 null type

9.4.1.9 object identifier

9.4.1.10 octetstring type

9.4.1.11 production

9.4.1.12 simple type

9.4.1.13 sequence of type

9.4.1.14 sequence type

9.4.1.15 structured type

9.4.1.16 tag

9.4.1.17 tagged type

9.4.1.18 true

9.4.1.19 type

9.4.2 ISO/IEC 10731 terms

(N)-connection
(N)-entity
(N)-layer
(N)-service
(N)-service-access-point
confirm (primitive)
indication (primitive)
request (primitive)
response (primitive)

9.4.3 Other terms and definitions

The following terms and definitions are used in this specification

9.4.3.1 receiving

Service user that receives a confirmed primitive or an unconfirmed primitive, or a service provider that receives a confirmed APDU or an unconfirmed APDU.

9.4.3.2 resource

Resource is a processing or information capability of a subsystem.

9.4.3.3 sending

Service user that sends a confirmed primitive or an unconfirmed primitive, or a service provider that sends a confirmed APDU or an unconfirmed APDU.

9.5 Conventions

9.5.1 General conventions

This PAS uses the descriptive conventions given in ISO/IEC 10731.

This PAS uses the descriptive conventions given in Clause 9 for FAL service definitions.

9.5.2 Convention for the encoding of reserved bits and octets

The term “reserved” may be used to describe bits in octets or whole octets. All bits or octets that are reserved should be set to zero on the sending side. They will not be tested on the receiving side except if explicitly stated, or if the reserved bits or octets are checked by a state machine.

The term “reserved” may also be used to indicate that certain values within the range of a parameter are reserved for future extensions. In this case the reserved values should not be used at the sending side. They shall not be tested at the receiving side except if explicitly stated, or if the reserved values are checked by a state machine.

9.5.3 Conventions for the common coding of specific field octets

APDUs may contain specific fields that carry information in a primitive and condensed way. These fields shall be coded in the order given in Figure 61.

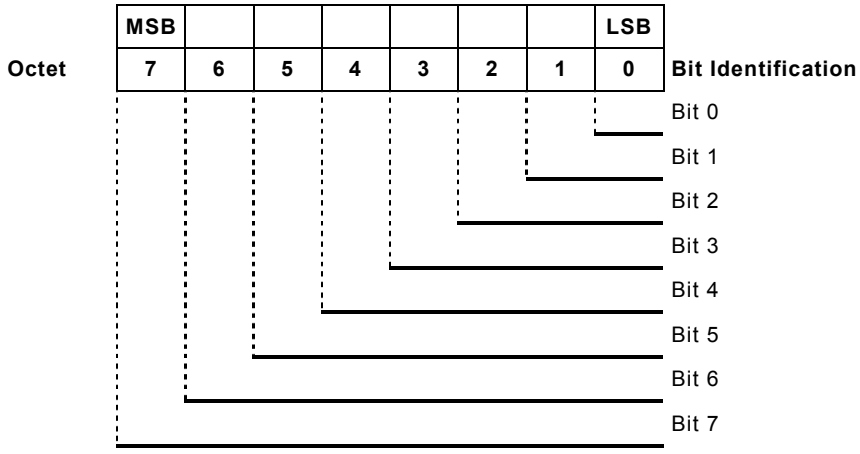


Figure 61 – Common structure of specific fields

Bits may be grouped. Each bit or group of bits shall be addressed by its bit identification (for example, Bit 0, Bits 1–4). The position within the octet shall be as shown in Figure 61. Alias names may be used for each bit or group of bits, or they may be marked as reserved. The grouping of individual bits shall be in ascending order without gaps. The values for a group of bits may be represented as binary, decimal, or hexadecimal values. This value shall only be valid for the grouped bits and can only represent the whole octet if all 8 bits are grouped. Decimal or hexadecimal values shall be transferred in binary values so that the bit with the highest group number represents the most significant bit (MSB) of the grouped bits.

EXAMPLE 1: Description and relation for the specific field octet

Bit 0: reserved.

Bits 1–3: Reason_Code. The decimal value 2 for the Reason_Code means general error.

Bits 4–7: Always set to one.

The octet that is constructed according to the description above looks as follows:

(MSB) Bit 7 = 1,

Bit 6 = 1,

Bit 5 = 1,

Bit 4 = 1,

Bit 3 = 0,

Bit 2 = 1,

Bit 1 = 0,

(LSB) Bit 0 = 0.

This bit combination has an octet value representation of 0xf4.

9.5.4 Conventions for APDU abstract syntax definitions

This PAS uses the descriptive conventions given in ISO/IEC 8824-2 for APDU definitions.

9.5.5 Conventions for APDU transfer syntax definitions

This PAS uses the descriptive conventions given in ISO/IEC 8825-1 for transfer syntax definitions.

9.5.6 Conventions for AE state machine definitions

The conventions used for AE state machine definitions are described in Table 115.

Table 115 – Conventions used for AE state machine definitions

No.	Current state	Event/condition => action	Next state
Name of this transition	Current state to which this state transition applies	Events or conditions that trigger this state transition => The actions that are taken when the above events or conditions are met. The actions are always indented below events or conditions	The next state after the actions in this transition are taken

The conventions used in the descriptions for the events, conditions and actions are as follows.

:= The value of an item on the left is replaced by the value of an item on the right. If an item on the right is a parameter, it comes from the primitive shown as an input event.

xxx Parameter name.

Example:

Identifier := Reason

means value of the Reason parameter is assigned to the parameter called Identifier.

“xxx” Indicates a fixed value.

Example:

Identifier := “abc”

means value “abc” is assigned to a parameter named “Identifier.”

= A logical condition to indicate an item on the left is equal to an item on the right.

< A logical condition to indicate an item on the left is less than the item on the right.

> A logical condition to indicate an item on the left is greater than the item on the right.

<> A logical condition to indicate an item on the left is not equal to an item on the right.

&& Logical “AND”

|| Logical “OR”

The sequence of actions and the alternative actions can be executed using the following reserved words.

for

endfor

if

else

elseif

The following shows examples of description using the reserved words.

Example 1:

```
for (Identifier := start_value to end_value)
        actions
endfor
```

Example 2:

```
If (condition)
        actions
else
        actions
endif
```

9.6 FAL syntax description

9.6.1 General

This description of the RAPIEnet abstract syntax uses formalisms similar to ASN.1, although the encoding rules differ from that standard.

9.6.2 FAL-AR PDU abstract syntax

9.6.2.1 Top level definition

```
APDU ::= CHOICE {
    ConfirmedSend-CommandPDU
    ConfirmedSend-ResponsePDU
    UnconfirmedSend-CommandPDU
}
```

9.6.2.2 Confirmed send service

```
ConfirmedSend-CommandPDU::= SEQUENCE {
    FalArHeader
    InvokeID
    ServiceType
    ConfirmedServiceRequest
}

ConfirmedSend-ResponsePDU::= SEQUENCE {
    FalArHeader
    InvokeID
    ServiceType
    ConfirmedServiceResponse
}
```

9.6.2.3 Unconfirmed send service

```
UnconfirmedSend-CommandPDU::= SEQUENCE {
      FalArHeader
      InvokeID
      ServiceType
      UnconfirmedServiceRequest
}
```

9.6.2.4 FalArHeader

```
FalArHeader ::= Unsigned8 {
  -- bit 8-7        ProtocolVersion
  -- bit 6-4        Protocol Identifier        Identifiers abstract syntax revision, and encoding rules
   --bit 3-1        PDU Identifier             Identifies a PDU type within a Protocol Identifier
}
```

9.6.2.5 InvokeID

```
InvokeID ::= Unsigned8                        Identifies this invocation of the service
```

9.6.2.6 ServiceType

```
ServiceType ::= Unsigned16                    Contains the context specific tags for the PDU body
```

9.6.3 Abstract syntax of PDU body

9.6.3.1 ConfirmedServiceRequest PDUs

```
ConfirmedServiceRequest::= CHOICE {
      Read-Request                [0]    IMPLICIT      Read-RequestPDU,
      Write-Request               [1]    IMPLICIT      Write-RequestPDU,
      Identify-Request            [2]    IMPLICIT      Identify-RequestPDU,
      Status-Request              [3]    IMPLICIT      Status-RequestPDU,
}
```

9.6.3.2 ConfirmedServiceResponse PDUs

```
ConfirmedServiceRequest::= CHOICE {
      Read-Response               [0]    IMPLICIT      Read-ResponsePDU,
      Write-Response              [1]    IMPLICIT      Write-ResponsePDU,
      Identify-Response           [2]    IMPLICIT      Identify-ResponsePDU,
      Status-Response             [3]    IMPLICIT      Status-ResponsePDU,
}
```

9.6.3.3 UnconfirmedServiceRequest PDUs

```
UnconfirmedServiceRequest::= CHOICE  {
      TB-Request                  [0]    IMPLICIT      TB-transferPDU
      COS-Request                 [1]    IMPLICIT      COS-transferPDU

}
```

9.6.3.4 Error information

9.6.3.4.1 Error type

```
ConfirmedServiceRequest::= CHOICE  {
      Service status              [0]    IMPLICIT      ErrorClass,
      Status code                 [1]    IMPLICIT      Integer16 OPTIONAL,
}
```

9.6.3.4.2 Error class

```
ErrorClass ::= CHOICE {
    noError          [0]  IMPLICIT  Integer8 {
                            normal                  (0),
                            Other                   (1)
    }
    Definition       [1]  IMPLICIT  Integer8 {
                            other                   (0),
                            object-undefined        (1),
    }
    Resource         [2]  IMPLICIT  Integer8 {
                            other                   (0),
                            memory-unavailable      (1)
    }
    Service          [3]  IMPLICIT  Integer8 {
                            other                   (0),
                            pdu-size                (1),
                            illegal-parameter       (2)
    }
    Access           [4]  IMPLICIT  Integer8 {
                            other                   (0),
                            object-access-denied    (1),
                            invalid-address         (2),
                            object-access-unsupported (3),
                            object-non-existent     (4),
    }
    Other            [5]  IMPLICIT  Integer8 {
                            other                   (0)
    }
}
```

9.6.3.4.3 Status code

The status codes for the confirmed response primitive are listed in Table 116.

Table 116 – Status code for the confirmed response primitive

Error code	Error type	Error details and causes
0x01	Service type error	Unknown service type
0x02	Object identifier error	Object-access-unsupported
0x03	Data type error	Other data type than supported
0x04	Object offset error	Request exceeds the area each object supports
0x05	Data length error	Request exceeds the maximum range of Ethernet rules to read or write at a time

9.6.4 Protocol data units (PDUs) for application service elements (ASEs)

9.6.4.1 PDUs for application process ASE

9.6.4.1.1 Identify-Request PDUs

Identify-Request PDU::= NULL

9.6.4.1.2 Identify-Response PDUs

Identify-ResponsePDU ::= SEQUENCE {
  DL-address
  MAC address
  Port information
  Protocol version
  Device type
  Device description
  Hardware-Version
  Serial Number
  Software-Version
  Software-Date
  Vendor ID
  Product Code
}

9.6.4.1.3 Status-Request PDUs

Status-Request PDU::= NULL

9.6.4.1.4 Status-Response PDUs

Status-ResponsePDU ::= SEQUENCE {
  Device flags
  Device state
  tx_cnt_normal
  tx_cnt_all
  rx_cnt_normal
  rx_cnt_all
  relay_cnt_normal
  relay_cnt_all
}

9.6.4.1.5 DL-address

dl-address ::= Unsigned16 — See 8.3.6.4.1

9.6.4.1.6 MAC address

MAC address::= Unsigned48 — See 8.3.6.4.7

9.6.4.1.7 Port information

Port information::= Unsigned16 — See 8.3.6.4.8

9.6.4.1.8 Protocol version

Protocol version ::= Unsigned8 — See 8.3.6.4.9

9.6.4.1.9 Device type

Device type::= Unsigned16 — See 8.3.6.4.10

9.6.4.1.10 Device description

Device description ::= VISIBLE_STRING[16] — See 8.3.6.4.11

9.6.4.1.11 Hardware-Version

Hardware-Version ::= Unsigned16 — Contains the hardware version of the device

9.6.4.1.12 Serial number

Serial Number ::= Unsigned16 — Contains the serial number of the device

9.6.4.1.13 Software-Version

Software-Version ::= Unsigned16 — Contains the software version of the device

9.6.4.1.14 Vendor ID

Vendor ID ::= Unsigned16 — Contains the vendor ID of the device

9.6.4.1.15 Product Code

Product Code ::= Unsigned16 — Contains the product code of the device

9.6.4.1.16 Device flags

Device flags ::= Unsigned16 — See 8.3.6.4.2

9.6.4.1.17 Device state

Device state ::= Unsigned16 — See 8.3.6.4.3

9.6.4.1.18 tx_cnt_normal

tx_cnt_normal ::= Unsigned32 — The value of the transmit count of normal packets

9.6.4.1.19 tx_cnt_all

tx_cnt_all ::= Unsigned32 — The value of the transmit count of both error packets and normal packets

9.6.4.1.20 rx_cnt_normal

rx_cnt_normal ::= Unsigned32 — The value of the receive count of normal packets

9.6.4.1.21 rx_cnt_all

rx_cnt_all ::= Unsigned32 — The value of the receive count of both error packets and normal packets

9.6.4.1.22 relay_cnt_normal

relay_cnt_normal ::= Unsigned32 — The value of the relay count of normal packets

9.6.4.1.23 relay_cnt_all

relay_cnt_all ::= Unsigned32 — The value of the relay count of both error packets and normal packets

9.6.4.2 PDUs for service data object ASE

9.6.4.2.1 Read service PDUs

```
Read-RequestPDU::= SEQUENCE {
   Object List Count
   Object identifier (k)
   Data type
   offset
   length
   Object identifier (m)
   Data type
   offset
   length
   …                                   — (further read requests)
}
Read-ResponsePDU ::= SEQUENCE {
   Service status
   Object List Count
   Object identifier (k)
   Data type
   offset
   length
   data
   Object identifier (m)
   Data type
   offset
   length
   data
   …                                   — (further read responses)
}
```

9.6.4.2.2 Write service PDUs

```
Write-RequestPDU::= SEQUENCE {
   Object List Count
   Object identifier (k)
   Data type
   offset
   length
   data
   Object identifier (m)
   Data type
   offset
   length
   data
   ...                                  — (further write requests)
}

Write-ResponsePDU ::= SEQUENCE {
   Service status
   Status code                          — (optional)
}
```

9.6.4.2.3 Object List Count

Object list count ::= Unsigned16 — Number of objects

9.6.4.2.4 Object identifier

Object identifier ::= Unsigned16 — Specifies an entry of the device object

9.6.4.2.5 Data type

Data type ::= Unsigned16 — Contains the numeric identifier of the data type

9.6.4.2.6 Offset

offset ::= Unsigned32 — Provides the offset related to the start of the object

9.6.4.2.7 Length

length ::= Unsigned32 — The number of octets of the service which are to be read or written

9.6.4.2.8 data

data ::= Any — Application-dependent type and length;
total frame length must comply with Ethernet rules

9.6.4.3 PDUs for Process data object ASE

9.6.4.3.1 TB-transfer service PDUs

```
TB-transfer PDU ::= SEQUENCE {
      TBArep,      -- Block number
      TBData       -- content of TB segment
}
```

9.6.4.3.2 TBArep

TBArep ::= Unsigned16 — Block number

9.6.4.3.3 TBData

```
TBData ::= SEQUENCE {
   blen     Unsigned16,                 — TB byte length
   payload-data::=Any                   — TB content
```

9.6.4.3.4 COS-transfer service PDUs

```
COS-transfer PDU ::= SEQUENCE {
      COSArep,    -- Block number
      COSData     -- content of COS segment
}
```

9.6.4.3.5 COSArep

```
COSArep ::= Unsigned16                    — Block number
```

9.6.4.3.6 COSData

```
COSData ::= SEQUENCE {
   blen     Unsigned16,                   — COS byte length
   payload-data::=Any                     — COS content
```

9.7 Transfer syntax

9.7.1 Overview of encoding

The encoded FAL-PDUs encoded shall have a uniform format. They shall consist of two major parts: the APDU header part and the APDU body part as shown in Figure 62.

(1)	(1)	(2)	(n) --- octets
FalArHeader Field	(InvokeID)	ServiceType	Service Specific Parameters
<---------- APDU header ---------->			<---------- APDU body ---------->

NOTE The presence of the InvokeID Field depends on the APDU type.

Figure 62 – APDU overview

To realize an efficient APDU while maintaining flexible encoding, different encoding rules are used for the APDU header part and the APDU body part.

NOTE The DLL service provides a DLSDU parameter that implies the length of the APDU. Thus, the APDU length information is not included in the APDU.

9.7.2 APDU header encoding

The APDU header part is always present in all APDUs that conform to this specification. It consists of three fields: the FalArHeader Field, the InvokeID Field, and the ServiceType Field, as shown in Figure 62.

9.7.2.1 Encoding of FalArHeader field

All the FAL PDUs have the common PDU-header called FalArHeader. The FalArHeader identifies abstract syntax, transfer syntax, and each of the PDUs. Table 117 defines how this header shall be used.

Table 117 – Encoding of FalArHeader field

Bit position of the FalArHeader			PDU type	Protocol version
8 7	6 5 4	3 2 1		
01	001	000	ConfirmedSend-CommandPDU	Version 1
01	001	100	ConfirmedSend-ResponsePDU	Version 1
01	010	000	UnconfirmedSend-CommandPDU	Version 1

NOTE All other code points are reserved for additional protocols and future revisions.

9.7.2.2 Encoding of InvokeID Field

The InvokeID Field shall be present if it is indicated in the abstract syntax. Otherwise, this field shall not be present. If present, the InvokeID parameter supplied by a service primitive shall be placed in this field.

9.7.2.3 Encoding of Type field

The service type of an APDU is encoded in the Type field that is always the third octet of the APDU.

All bits of the Type field are used to encode the service type, as follows.

a) The service types shall be encoded in bits 16 to 1 of the Type field, with bit 16 the MSB and bit 1 the LSB. The range of service type shall be in the range 0–65534.

b) The value of 65535 is reserved for future extensions to this specification.

c) The service type is specified in the abstract syntax as a positive integer value.

Figure 63 illustrates the encoding of the Type field.

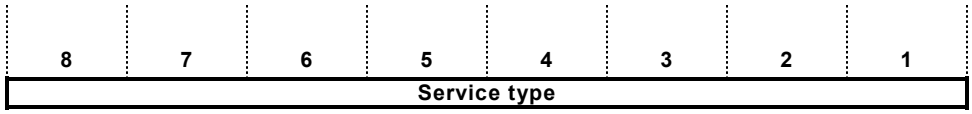


Figure 63 – Type field

9.7.3 APDU body encoding

9.7.3.1 General

The FAL encoding rules are based on the terms and conventions defined in ISO/IEC 8825-1. The encoding consists of three components in the following order:

a) identifier octet;
b) length octet(s);
c) contents octet(s).

NOTE Identification octet and content length octets do not exist in RAPIEnet.

9.7.4 Encoding of Data types

9.7.4.1 General description of data types and encoding rules

The format of this data and its meaning must be known by the producer and consumer(s) to be able to exchange meaningful data. This specification model uses the concept of data types to achieve this.

The encoding rules define the representation of values of data types and the transfer syntax for the representations. Values are represented as bit sequences. Bit sequences are transferred in sequences of octets (bytes). For numerical data types, the encoding is little-endian style as shown in Table 118.

9.7.4.2 Transfer syntax for bit sequences

A bit sequence is reordered into a sequence of octets for transmission. Hexadecimal notation is used for octets as specified in ISO/IEC 9899. Let $b = b_0 \ldots b_{n-1}$ be a bit sequence and k be a non-negative integer such that $8(k-1) < n \leq 8k$. Then, b is transferred in k octets assembled as shown in Table 118. The bits b_i, $i \geq n$ of the highest numbered octet are do-not-care bits.

Table 118 – Transfer syntax for bit sequences

Octet number	1.	2.	k.
	b_7 .. b_0	b_{15} .. b_8	b_{8k-1} .. b_{8k-8}

Octet 1 is transmitted first and octet k is transmitted last. The bit sequence is transferred as follows across the network (transmission order within an octet is determined by ISO/IEC 8802-3):

$b_7, b_6, ..., b_0, b_{15}, ..., b_8, ...$

EXAMPLE

Bit 9	...	Bit 0
10b	0001b	1100b
2h	1h	Ch
		= 21Ch

The bit sequence $b = b_0 .. b_9 = 0011\ 1000\ 01_b$ represents an Unsigned10 with the value 21Ch, and is transferred in two octets: first 1Ch and then 02h.

9.7.4.3 Encoding of a Boolean value

Data of basic data type BOOLEAN can have the values TRUE or FALSE.

The values are represented as bit sequences of length 1. The value TRUE is represented by 1, and FALSE by 0.

A BOOLEAN shall be transferred over the network as UNSIGNED8 of value 1 (TRUE) or 0 (FALSE). Sequences of BOOLEANs may be packed into one UNSIGNED8. Sequences of BOOLEAN and BIT type items may be also packed into one UNSIGNED8.

9.7.4.4 Encoding of an unsigned integer value

Data of basic data type UNSIGNEDn has values in the non-negative integers. The value range is 0, ..., 2^n-1. The data are represented as bit sequences of length n.

The bit sequence

$b = b_0 ... b_{n-1}$

is assigned the value

$$UNSIGNEDn(b) = b_{n-1}\times 2^{n-1} + ... + b_1\times 2^1 + b_0\times 2^0$$

It should be noted that the bit sequence starts on the left with the least significant byte.

Example: The value $266_d = 10A_h$ with data type UNSIGNED16 is transferred in two octets across the bus, first $0A_h$ and then 01_h.

The UNSIGNEDn data types are transferred as shown in Table 119.

Table 119 – Transfer syntax for data type UNSIGNEDn

Octet number	0	1	2	3	4	5	6	7
UNSIGNED8	$b_7..b_0$							
UNSIGNED16	$b_7..b_0$	$b_{15}..b_8$						
UNSIGNED32	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$				
UNSIGNED64	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	$b_{63}..b_{56}$

9.7.4.5 Encoding of a signed integer

The data of basic data type INTEGERn has values in the integers. The value range is from -2^{n-1} to $2^{n-1}-1$. The data are represented as bit sequences of length n. The bit sequence

$$b = b_0 .. b_{n-1}$$

is assigned the value

$$\text{INTEGERn}(b) = b_{n-2} \times 2^{n-2} + ... + b_1 \times 2^1 + b_0 \times 2^0 \text{ if } b_{n-1} = 0$$

and, performing two's complement arithmetic,

$$\text{INTEGERn}(b) = - \text{INTEGERn}(\hat{}b) - 1 \text{ if } b_{n-1} = 1$$

It should be noted that the bit sequence starts on the left with the least significant bit.

Example: The value $-266_d = 0xFEF6_h$ with data type Integer16 is transferred in two octets, first 0xF6 and then 0xFE.

The INTEGERn data types are transferred as specified in Table 120.

Table 120 – Transfer syntax for data type INTEGERn

Octet number	0	1	2	3	4	5	6	7
INTEGER8	$b_7..b_0$							
INTEGER16	$b_7..b_0$	$b_{15}..b_8$						
INTEGER32	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$				
INTEGER64	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	$b_{63}..b_{56}$

9.7.4.6 Encoding of a floating point value

Data of basic data types REAL32 and REAL64 have values in real numbers.

The data type REAL32 is represented as a bit sequence of length 32. The encoding of values follows the IEEE 754-1985 standard for single-precision floating point.

The data type REAL64 is represented as a bit sequence of length 64. The encoding of values follows the IEEE 754:1985 standard for double-precision floating point numbers.

A bit sequence of length 32 either has a value (finite non-zero real number, ± 0, ± _) or is not a number (NaN).

The bit sequence

$b = b_0 \ldots b_{31}$

is assigned the value (finite non-zero number)

$REAL32(b) = (-1)^S \times 2^{E-127} \times (1+ F)$

where

$S = b_{31}$ is the sign.

$E = b_{30} \times 2^7 + \ldots + b_{23} \times 2^0$, $0 < E < 255$, is the un-biased exponent.

$F = 2^{-23} \times (b_{22} \times 2^{22} + \ldots + b_1 \times 2^1 + b_0 \times 2^0)$ is the fractional part of the number.

E = 0 is used to represent ± 0. E = 255 is used to represent infinities and NaNs.

It should be noted that the bit sequence starts on the left with the least significant bit.

9.7.4.7 Encoding of an octet string value

The data type OCTET_STRINGlength is defined as follows where “length” is the length of the octet string.

ARRAY [length] OF UNSIGNED8 OCTET_STRINGlength

9.7.4.8 Encoding of a visible string value

VISIBLE_CHAR are 0_h and the range from 20_h to $7E_h$. The data are interpreted as ISO 646:1973(E) 7-bit coded characters, and “length” is the length of the visible string.

UNSIGNED8 VISIBLE_CHAR

ARRAY [length] OF VISIBLE_CHAR VISIBLE_STRINGlength

There is no 0_h necessary to terminate the string.

9.7.4.9 Encoding of a Unicode string value

The data type UNICODE_STRINGlength is defined below where “length” is the length of the Unicode string.

ARRAY [length] OF UNSIGNED16 UNICODE_STRINGlength

9.7.4.10 Encoding of a time of day value

The data type TimeOfDay represents absolute time. TimeOfDay is represented as a bit sequence of length 48.

Component “ms” is the time in milliseconds after midnight. Component “days” is the number of days since January 1, 1984.

```
STRUCT OF
   UNSIGNED28      ms,
   VOID4           reserved,
   UNSIGNED16      days
TIME_OF_DAY
```

The encoding is as shown in Figure 64.

bits	7	6	5	4	3	2	1	0	
octets									
1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	number of milliseconds since midnight
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	number of days since 01.01.84
6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
	msb								

Figure 64 – Encoding of time-of-day value

9.7.4.11 Encoding of a time difference value

The data type TimeDifference represents a time difference or a period of time. TimeDifference is represented as a bit sequence of length 48.

Time differences are sums of numbers of days and milliseconds. Component “ms” is the number of milliseconds. Component “days” is the number of days.

```
STRUCT OF
   UNSIGNED28      ms,
   VOID4           reserved,
   UNSIGNED16      days
TIME_DIFFERENCE
```

The encoding is as shown in Figure 65.

bits	7	6	5	4	3	2	1	0	
octets									
1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	milliseconds
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	days
6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
	msb								

Figure 65 – Encoding of time difference value

9.8 FAL protocol state machines

Interfaces to FAL services and protocol machines are specified in this section.

NOTE The state machines specified in this section and Application Relationship Protocol Machines defined below only define the valid events for each. Handling invalid events is a local responsibility.

The behaviour of the FAL is described by the protocol machines shown in Figure 66. The three types of protocol machine are: FAL service protocol machines (FSPMs), application relationship protocol machines (ARPMs), and DLL mapping protocol machines (DMPMs). Specific sets of these protocol machines are defined for different types of application relationship endpoint (AREP). Figure 66 also shows the primitives exchanged between the protocol machines.

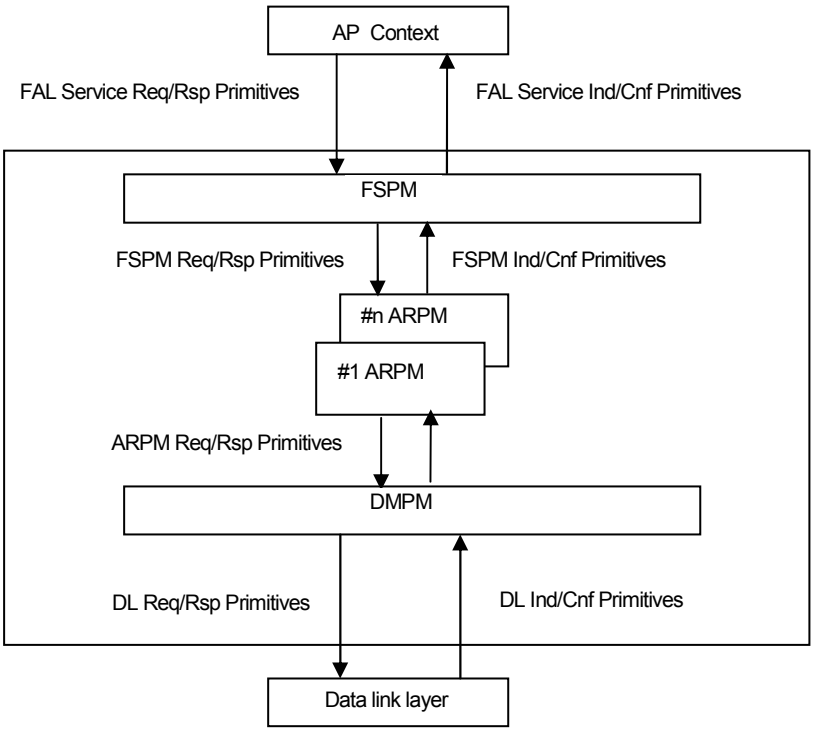


Figure 66 – Primitives exchanged between protocol machines

The FSPM is responsible for the following.

a) Accepting service primitives from the FAL service user and converting them into FAL internal primitives.

b) Selecting an appropriate ARPM state machine based on the AREP Identifier parameter supplied by the AP-Context and sending FAL internal primitives to the selected ARPM.

c) Accepting FAL internal primitives from the ARPM and converting them into service primitives for the AP context.

d) Delivering the FAL service primitives to the AP context based on the AREP Identifier parameter associated with the primitives.

The ARPM is responsible for the following.

a) Accepting FAL internal primitives from the FSPM, and creating and sending other FAL internal primitives to either the FSPM or the DMPM, based on the AREP and primitive types.

b) Accepting FAL internal primitives from the DMPM and sending them to the FSPM in a converted form for the FSPM.

c) To establish or release the specified AR if the primitives are for the Establish or Abort services, respectively.

The DMPM describes the mapping between the FAL and the DLL. It is common to all the AREP types and does not have any state changes. The DMPM is responsible for the following activities.

a) Accepting FAL internal primitives from the ARPM, preparing DLL service primitives, and sending them to the DLL.

b) Receiving DLL indication or confirmation primitives from the DLL and sending them to the ARPM in a converted form for the ARPM.

9.9 AP context state machine

There is no AP context state machine defined for this protocol.

9.10 FAL service protocol machine

9.10.1 General

These FSPMs are defined as follows.

a) Application process ASE Protocol Machine (APAM)

b) Service data object ASE Protocol Machine (SDOM)

c) Process data object ASE Protocol Machine (PDOM)

9.10.2 Common parameters of the primitives

Many services share a group of common parameters. Instead of defining them repeatedly with each individual service, the following common definitions are provided once below.

AREP
This parameter contains sufficient information for local identification of the AREP to be used to convey the service. This parameter may use a key attribute of the AREP to identify the application relationship.

InvokeID
This parameter identifies this invocation of the service. It is used to associate a service request with its response. Therefore, no two outstanding service invocations can be identified by the same InvokeID value.

Service status
This parameter provides information on the result of service execution. It is returned in all confirmed service response primitives (+ and -).

9.10.3 AP ASE protocol machine

9.10.3.1 Primitive definitions

9.10.3.1.1 Primitives exchanged

Table 121 shows the service primitives, including their associated parameters exchanged between the FAL-user and the APAM.

Table 121 – Primitives exchanged between FAL-user and APAM

Primitive	Source	Associated parameters	Functions
Identify.req	FAL-user	AREP InvokeID Service type	This primitive is used to request device information
Status.req	FAL-user	AREP InvokeID Service type	This primitive is used to request device status
Identify.rsp	FAL-user	AREP InvokeID Service type Service status Identify Data	This primitive is used to respond to an identify request
Status.rsp	FAL-user	AREP InvokeID Service type Service status Status Data	This primitive is used to respond to a status request
Identify.ind	APAM	AREP InvokeID Service type	This primitive is used to indicated an identify request
Status.ind	APAM	AREP InvokeID Service type	This primitive is used to indicate a status request
Identify.cnf	APAM	AREP InvokeID Service type Service status Identify Data	This primitive is used to confirm an identify request
Status.cnf	APAM	AREP InvokeID Service type Service status Status Data	This primitive is used to confirm a status request

9.10.3.1.2 Parameters of primitives

The parameters used with the primitives exchanged between the FAL-user and the APAM are listed in Table 122.

Table 122 – Parameters used with primitives exchanged FAL-user and APAM

Parameter	Description
AREP	See 10.11.2
InvokeID	See 10.11.2

Parameter	Description
Identify data	See Clause 9
Status data	See Clause 9

9.10.3.2 State machine

9.10.3.2.1 General

The APAM State Machine has only one possible state: ACTIVE.

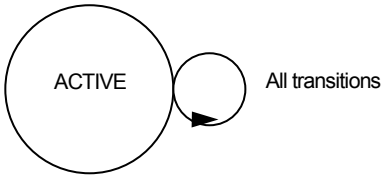


Figure 67 – State transition diagram of APAM

9.10.3.2.2 State tables

The APAM state machine is described in Figure 67 and in Table 123 and Table 124.

Table 123 – APAM state table – Sender transitions

#	Current state	Event or condition => action	Next state

S1	ACTIVE	Identify.req
		=>
		SelectArep(RemoteArep, “PTC-AR”),
		CS_req{
		user_data := Identify-RequestPDU
		}
			ACTIVE

S2	ACTIVE	Status.req
		=>
		SelectArep(RemoteArep, “PTC-AR”),
		CS_req{
		user_data := Status-RequestPDU
		}
			ACTIVE

S3	ACTIVE	Identify.rsp
		=>
		SelectArep(ArepID, “PTC-AR”),
		CS_rsp{
		user_data := Identify-ResponsePDU
		}
			ACTIVE

S4	ACTIVE	Status.rsp
		=>
		SelectArep(ArepID, “PTC-AR”),
		CS_rsp{
		user_data := Status-ResponsePDU
		}
			ACTIVE

Table 124 – APAM state table – Receiver transitions

#	Current state	Event or condition => action	Next state

R1	ACTIVE	CS_ind
		&& PDU_Type = Identify-RequestPDU
		=>
		Status.ind{
		ArepID := arep_id
		Data := user_data
		}
			ACTIVE

R2	ACTIVE	CS_ind
		&& PDU_Type = Status-RequestPDU
		=>
		Status.ind{
		ArepID := arep_id
		Data := user_data
		}
			ACTIVE

R3	ACTIVE	CS_ind
		&& PDU_Type = Identify-ResponsePDU
		&& GetErrorInfo() = “success”
		=>
		Status.cnf(+){
		Data := user_data
		}
			ACTIVE

R4	ACTIVE	CS_ind
		&& PDU_Type = Identify-ResponsePDU
		&& GetErrorInfo() <> “success”
		=>
		Status.cnf(-){
		Status := GetErrorInfo()
		}
			ACTIVE

R5	ACTIVE	CS_ind
		&& PDU_Type = Status-ResponsePDU
		&& GetErrorInfo() = “success”
		=>
		Status.cnf(+){
		Data := user_data
		}
			ACTIVE

R6	ACTIVE	CS_ind
		&& PDU_Type = Status-ResponsePDU
		&& GetErrorInfo() <> “success”
		=>
		Status.cnf(-){
		Status := GetErrorInfo()
		}
			ACTIVE

9.10.3.2.3 Functions

Table 125 lists the functions used by the APAM, their arguments, and their descriptions.

Table 125 – Functions used by the APAM

Function name	Parameter	Description
SelectArep	ArepID ARtype	Looks for the AREP entry that is specified by the ArepID and AR type
GetErrorInfo		Gets error information from the APDU
GetService	InvokeID	Gets service name from the InvokeID

9.10.4 Service data object ASE protocol machine (SDOM)

9.10.4.1 Primitive definitions

9.10.4.1.1 Primitives exchanged

Table 126 shows the service primitives and their associated parameters that are exchanged between the FAL-user and the SDOM.

Table 126 – Primitives exchanged between FAL-user and SDOM

Primitive	Source	Associated parameters	Functions
Read.req	FAL-user	AREP InvokeID Object List Count Object List(n)	This primitive is used to read values of a service data object
Write.req	FAL-user	AREP InvokeID Object List Count Object List(n) Data(n)	This primitive is used to write values of a service data object
Read.rsp	FAL-user	AREP InvokeID Object List Count Data(n)	This primitive is used to convey requested values of a service data object
Write.rsp	FAL-user	AREP InvokeID Service status	This primitive is used to report result of writing requested
Read.ind	SDOM	AREP InvokeID Object List Count Object List(n)	This primitive is used to convey a read request
Write.ind	SDOM	AREP InvokeID Object List Count Object List(n) Data(n)	This primitive is used to convey a write request
Read.cnf	SDOM	AREP InvokeID Object List Count Data(n)	This primitive is used to convey values of data requested and result of reading
Write.cnf	SDOM	AREP InvokeID Service status	This primitive is used to report result of writing requested

9.10.4.1.2 Parameters of primitives

The parameters used with the primitives exchanged between the FAL-user and the SDOM are listed in Table 127.

Table 127 – Parameters used with primitives exchanged FAL-user and SDOM

Parameter	Description
AREP	See 10.11.2
InvokeID	See 10.11.2
Object List Count	This parameter specifies the object to which the data are to be read or written
Object List	This parameter specifies object list
Data	This parameter specifies object related data

9.10.4.2 State machine

9.10.4.2.1 General

The SDOM State Machine has only one possible state: ACTIVE.

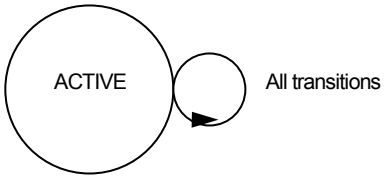


Figure 68 – State transition diagram of SDOM

9.10.4.2.2 State tables

The SDOM state machine is described in Figure 68, and in Tables 128 and 129.

Table 128 – SDOM state table – Sender transitions

#	Current state	Event or condition => action	Next state

S1	ACTIVE	Read.req
		=>
		ArepID := GetArep(Object Specifier)
		SelectArep(ArepID, “PTC-AR”),
		CS_req{
		user_data := Read-RequestPDU
		}
			ACTIVE

S2	ACTIVE	Write.req
		=>
		ArepID := GetArep(OD Specifier)
		SelectArep(ArepID, “PTC-AR”),
		CS_req{
		user_data := Write-RequestPDU
		}
			ACTIVE

S3	ACTIVE	Read.rsp
		=>
		SelectArep(ArepID, “PTC-AR”),
		CS_rsp{
		user_data := Read-ResponsePDU
		}
			ACTIVE

S4	ACTIVE	Write.rsp
		=>
		SelectArep(ArepID, “PTC-AR”),
		CS_rsp{
		user_data := Write-ResponsePDU
		}
			ACTIVE

Table 129 – SDOM state table – Receiver transitions

#	Current state	Event or condition => action	Next state

R1	ACTIVE	CS_ind
		&& PDU_Type = Read-RequestPDU
		=>
		Read.ind{
		ArepID := arep_id
		Data := user_data
		}
			ACTIVE

R2	ACTIVE	CS_ind
		&& PDU_Type = Write-RequestPDU
		=>
		Write.ind{
		ArepID := arep_id
		Data := user_data,
		}
			ACTIVE

R3	ACTIVE	CS_ind
		&& PDU_Type = Read-ResponsePDU
		&& GetErrorInfo() = “success”
		=>
		Read.cnf(+){
		Service status := 0
		Data := user_data
		}
			ACTIVE

R4	ACTIVE	CS_ind
		&& PDU_Type = Read-ResponsePDU
		&& GetErrorInfo() <> “success”
		=>
		Read.cnf(-){
		Service status := GetErrorInfo()
		}
			ACTIVE

R5	ACTIVE	CS_ind
		&& PDU_Type =Write-ResponsePDU
		&& GetErrorInfo() = “success”
		=>
		Write.cnf(+){
		Service status := 0
		Data := user_data
		}
			ACTIVE

R6	ACTIVE	CS_ind
		&& PDU_Type = Write-ResponsePDU
		&& GetErrorInfo() <> “success”
		=>
		Write.cnf(-){
		Service status := GetErrorInfo()
		}
			ACTIVE

9.10.4.2.3 Functions

Table 130 lists the functions used by the SDOM, their arguments and their descriptions.

Table 130 – Functions used by the SDOM

Function	Parameter	Description
SelectArep	ArepID ARtype	Looks for the AREP entry that is specified by the ArepID and AR type
GetArep	Object specifier	Look for the ArepID based on the specified object specifier
GetErrorInfo		Gets error information from the APDU
GetService	InvokeID	Gets service name from the InvokeID

9.10.5 Process data object ASE protocol machine (PDOM)

9.10.5.1 Primitive definitions

9.10.5.1.1 Primitives exchanged

Table 131 shows the service primitives and their associated parameters that exchanged between the FAL-user and the PDOM.

Table 131 – Primitives exchanged between FAL-user and PDOM

Primitive	Source	Associated parameters	Functions
TB.req	FAL-user	AREP TB PDO	This primitive is used to publish values of a process data object
COS.req	FAL-user	AREP COS PDO	This primitive is used to publish values of a process data object
TB.ind	PDOM	AREP TB PDO	This primitive is used to report values of process data object published
COS.ind	PDOM	AREP COS PDO	This primitive is used to report values of process data object published

9.10.5.1.2 Parameters of primitives

The parameters used with the primitives exchanged between the FAL-user and the PDOM are listed in Table 132.

Table 132 – Parameters used with primitives exchanged between FAL-user and PDOM

Parameter	Description
AREP	See 10.11.2
TB PDO	This parameter conveys timer-based FAL-user data
COS PDO	This parameter conveys change-of-state FAL-user data

9.10.5.2 State machine

9.10.5.2.1 General

The PDOM State Machine has only one possible state: ACTIVE.

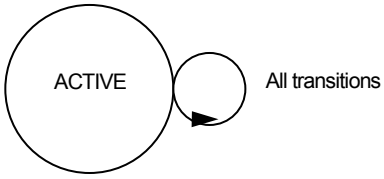


Figure 69 – State transition diagram of PDOM

9.10.5.2.2 State tables

The PDOM state machine is described in Figure 69, and in Tables 133 and 134.

Table 133 – PDOM state table – Sender transitions

#	Current state	Event or condition => action	Next state

S1	ACTIVE	TB.req
		=>
		SelectArep(ArepID, “MSU-AR”),
		UCS_req{
		user_data := TB-transferPDU
		}
			ACTIVE

S2	ACTIVE	COS.req
		=>
		SelectArep(ArepID, “MTU-AR”),
		UCS_req{
		user_data := COS-transferPDU
		}
			ACTIVE

Table 134 – PDOM state table – Receiver transitions

#	Current state	Event or condition => action	Next state

R1	ACTIVE	UCS_ind
		&& PDU_Type = TB-transferPDU
		=>
		TB.ind{
		ArepID := arep_id
		Data := user_data
		}
			ACTIVE

R2	ACTIVE	UCS_ind
		&& PDU_Type = COS-transferPDU
		=>
		COS.ind{
		ArepID := arep_id
		Data := user_data
		}
			ACTIVE

9.10.5.2.3 Functions

Table 135 lists the functions used by the SDOM, their arguments and their descriptions.

Table 135 – Functions used by the SDOM

Function name	Parameter	Description
SelectArep	ArepID ARtype	Looks for the AREP entry that is specified by the ArepID and AR type

9.11 AR protocol machine

9.11.1 General

This fieldbus has ARPMs for

a) point-to-point user-triggered confirmed client/server AREP (PTC-AR);
b) multipoint network-scheduled unconfirmed publisher/subscriber AREP (MSU-AR);
c) multipoint user-triggered unconfirmed publisher/subscriber AREP (MTU-AR).

9.11.2 Point-to-point user-triggered confirmed client/server AREP (PTC-AR) ARPM

9.11.2.1 PTC-AR Primitive definitions

9.11.2.1.1 Primitives exchanged between PTC-ARPM and user

Tables 136 and 137 list the primitives exchanged between the ARPM and the user.

Table 136 – Primitives issued by user to PTC-ARPM

Primitive name	Source	Associated parameters	Functions
CS_req	FSPM	Destination_dlsap_address InvokeID Service type User_data	This is an FAL internal primitive used to convey a Confirmed Send request primitive from the FSPM to the ARPM
CS_rsp	FSPM	Destination_dlsap_address InvokeID Service type User_data	This is an FAL internal primitive used to convey a Confirmed Send response primitive from the FSPM to the ARPM

Table 137 – Primitives issued by PTC-ARPM to user

Primitive name	Source	Associated parameters	Functions
CS_ind	ARPM	Source_dlsap_address InvokeID Service type User_data	This is an FAL internal primitive used to convey a Confirmed Send indication primitive from the ARPM to the FSPM
CS_cnf	ARPM	Source_dlsap_address InvokeID Service type User_data	This is an FAL internal primitive used to convey a Confirmed Send confirmation primitive from the ARPM to the FSPM

9.11.2.1.2 Parameters of primitives

The parameters of the primitives are described in Clause 9.

9.11.2.2 DLL mapping of PTC-AREP class

9.11.2.2.1 Formal model

This section describes the mapping of the PTC-AREP class to the RAPIEnet DLL defined in Clauses 7 and 8. It does not redefine the data link service access point (DLSAP) attributes or data link management entity (DLME) attributes that are or will be defined in the DLL PAS; rather, it defines how they are used by this AR class.

NOTE A means to configure and monitor the values of these attributes is outside the scope of this PAS.

The DLL mapping attributes and their permitted values and the DLL services used with the PTC-AR AREP class are defined in this in this section.

CLASS: **PTC-AR**
PARENT CLASS: **Point-to-point user-triggered confirmed client/server AREP**
ATTRIBUTES:
1 (m) KeyAttribute: LocalDlcepAddress
2 (m) Attribute: RemoteDlcepAddress
DLL SERVICES:
1 (m) OpsService: DL-DATA

9.11.2.2.2 Attributes

LocalDlcepAddress
This attribute specifies the local data link connection endpoint (DLCEP) address and to identify the DLCEP. The value of this attribute is used as the "DLCEP-address" parameter of the DLL.

RemoteDlcepAddress
This attribute specifies the remote DLCEP address and identifies the DLCEP.

9.11.2.2.3 DLL services

Refer to Chapter 7 for DLL service descriptions.

9.11.2.3 PTC-ARPM state machine

9.11.2.3.1 PTC-ARPM states

The PTC-ARPM state machine has only one state called "ACTIVE" (see Figure 70).

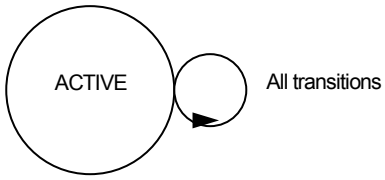


Figure 70 – State transition diagram of PTC-ARPM

9.11.2.3.2 PTC-ARPM state table

Tables 138 and 139 define the state machine of the PTC-ARPM.

Table 138 – PTC-ARPM state table – Sender transactions

#	Current state	Event or condition action	Next state

S1	ACTIVE	CS_req
		&& Role = "Client" || "Peer"
		=>
		FAL-PDU_req {
		dlsap_id := DLSAP_ID,
		called_address := Destination_dlsap_address,
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := "ConfirmedSend-CommandPDU,"
		fal_data := User_data)
		}
			ACTIVE

S2	ACTIVE	CS_rsp
		&& Role = "Server" || "Peer"
		=>
		FAL-PDU_req {
		dlsap_id := DLSAP_ID,
		called_address := Destination_dlsap_address,
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := "ConfirmedSend-ResponsePDU,"
		fal_data := User_data)
		}
			ACTIVE

Table 139 – PTC-ARPM state table – receiver transactions

#	Current state	Event or condition action	Next state

R1	ACTIVE	FAL-PDU_ind
		&& FAL_Pdu_Type (fal_pdu) = " ConfirmedSend-CommandPDU "
		&& Role = "Peer" || "Server"
		=>
		CS_ind{
		Source_dlsap_address := calling_address,
		user _data := fal_pdu
		}
			ACTIVE

R2	ACTIVE	FAL-PDU_ind
		&& FAL_Pdu_Type (fal_pdu) = " ConfirmedSend-ResponsePDU "
		&& Role = "Client" || "Peer"
		=>
		CS_cnf{
		user_data := fal_pdu
		}
			ACTIVE

9.11.2.3.3 Functions used by PTC-ARPM

Decoding to derive the relevant parameters for the state machine always follows receipt of an FAL-PDU_ind primitive. This is an implicit function and is not listed separately.

Table 140 defines the other function used by this state machine.

Table 140 – Function BuildFAL-PDU

Name	BuildFAL-PDU	Used in	ARPM
Input		Output	
Service type data additional information		DLSDU	
Function			
Builds an FAL-PDU out of the parameters given as input variables			

9.11.3 Multipoint network-scheduled unconfirmed publisher/subscriber AREP (MSU-AR) ARPM

9.11.3.1 MSU-AR primitive definitions

9.11.3.1.1 Primitives exchanged between MSU-ARPM and user

Table 141 and Table 142 list the primitives exchanged between the ARPM and the user.

Table 141 – Primitives issued by user to ARPM

Primitive name	Source	Associated parameters	Functions
UCS_req	FSPM	Remote_dlsap_address User_data	This is an FAL internal primitive used to convey an Unconfirmed Send request primitive from the FSPM to the ARPM

Table 142 – Primitives issued by ARPM to user

Primitive name	Source	Associated parameters	Functions
UCS_ind	ARPM	Remote_dlsap_address User_data	This is an FAL internal primitive used to convey an Unconfirmed Send indication primitive from the ARPM to the FSPM

9.11.3.1.2 Parameters of primitives

The parameters of the primitives are described in Clause 9.

9.11.3.2 DLL mapping of MSU-AR class

9.11.3.2.1 Formal model

This section describes the mapping of the MSU-AR AREP class to the RAPIEnet DLL defined in Clauses 7 and 8. It does not redefine the DLSAP attributes or DLME attributes that are or will be defined in the DLL PAS; rather, it defines how they are used by this AR class.

NOTE A means to configure and monitor the values of these attributes is outside the scope of this PAS.

The DLL mapping attributes with their permitted values and the DLL services used with the MTU-AR AREP class are defined in this section.

CLASS:			**MSU-AR**
PARENT CLASS:			**Multipoint network-scheduled unconfirmed publisher/subscriber AREP**
ATTRIBUTES:			
1	(m)	KeyAttribute:	LocalDlcepAddress
2	(m)	Attribute:	RemoteDlcepAddress
3	(m)	Attribute:	Role (Publisher, Subscriber)
DLL SERVICES:			
1	(m)	OpsService:	DL-DATA

9.11.3.2.2 Attributes

LocalDlcepAddress
This attribute specifies the local DLCEP address and identifies the DLCEP. The value of this attribute is used as the DLCEP-address parameter of the DLL.

RemoteDlcepAddress
This attribute specifies the remote DLCEP address and identifies the DLCEP.

Role
This attribute specifies the role of this AREP. A value of “Publisher” indicates that this AREP is used as a publisher. The value of “Subscriber” indicates that this AREP is used as a subscriber.

9.11.3.2.3 DLL services

Refer to Clause 7 for DLL service descriptions.

9.11.3.3 MSU-ARPM state machine

9.11.3.3.1 MSU-ARPM states

The MSU-ARPM state machine has only one state called “ACTIVE” (see Figure 71).

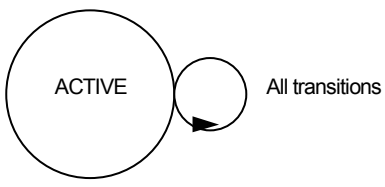


Figure 71 – State transition diagram of MSU-ARPM

9.11.3.3.2 MSU-ARPM state table

Tables 143 and 144 define the state machine of the MSU-ARPM.

Table 143 – MSU-ARPM state table – sender transactions

#	Current state	Event or condition action	Next state

S1	ACTIVE	UCS_req
		&& Role = “Publisher”
		=>
		FAL-PDU_req {
		dlsap_id := DLSAP_ID,
		called_address := Remote_dlsap_address,
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := “UnconfirmedSend-CommandPDU,”
		fal_data := user_data)
		}
			ACTIVE

Table 144 – MSU-ARPM state table – receiver transactions

#	Current state	Event or condition action	Next state

R1	ACTIVE	FAL-PDU_ind
		&& FAL_Pdu_Type (fal_pdu) = “UnconfirmedSend-CommandPDU “
		&& Role = “Subscriber”
		=>
		UCS_ind{
		remote_dlsap_address := calling_address,
		user_data := fal_pdu
		}
			ACTIVE

9.11.3.3.3 Functions used by MSU-ARPM

Decoding to derive the relevant parameters for the state machine always follows receipt of an FAL-PDU_ind primitive. This is an implicit function and is not listed separately.

Table 145 defines the other function used by this state machine.

Table 145 – Function BuildFAL-PDU

Name	BuildFAL-PDU	Used in	ARPM
Input		Output	
Service type data additional information		DLSDU	
Function			
Builds an FAL-PDU out of the parameters given as input variables.			

9.11.4 Multipoint user-triggered unconfirmed publisher/subscriber AREP (MTU-AR) ARPM

9.11.4.1 MTU-AR primitive definitions

9.11.4.1.1 Primitives exchanged between MTU-ARPM and user

Table 146 and Table 147 list the primitives exchanged between the ARPM and the user.

Table 146 – Primitives issued by user to ARPM

Primitive name	Source	Associated parameters	Functions
UCS_req	FSPM	Remote_dlsap_address User_data	This is an FAL internal primitive used to convey an Unconfirmed Send request primitive from the FSPM to the ARPM

Table 147 – Primitives issued by ARPM to user

Primitive name	Source	Associated parameters	Functions
UCS_ind	ARPM	Remote_dlsap_address User_data	This is an FAL internal primitive used to convey an Unconfirmed Send indication primitive from the ARPM to the FSPM

9.11.4.1.2 Parameters of primitives

The parameters of the primitives are described in Clause 9.

9.11.4.2 DLL mapping of MTU-AR class

9.11.4.2.1 Formal model

This subclause describes mapping of the MSU-AR AREP class to the RAPIEnet DLL defined in Clauses 7 and 8. It does not redefine the DLSAP attributes or the DLME attributes that are or will be defined in the DLL specification; rather, it defines how they are used by this AR class.

NOTE A means to configure and monitor the values of these attributes is outside the scope of this PAS.

This section defines the DLL mapping attributes with their permitted values, and the DLL services used with the MSU-AR AREP class.

CLASS: **MTU-AR**

PARENT CLASS: **Multipoint user-triggered unconfirmed publisher/subscriber AREP**

ATTRIBUTES:

1	(m)	KeyAttribute:	LocalDlcepAddress
2	(m)	Attribute:	RemoteDlcepAddress
3	(m)	Attribute:	Role (Publisher, Subscriber)

DLL SERVICES:

1	(m)	OpsService:	DL-DATA

9.11.4.2.2 Attributes

LocalDlcepAddress
This attribute specifies the local DLCEP address and identifies the DLCEP. The value of this attribute is used as the “DLCEP-address” parameter of the DLL.

RemoteDlcepAddress
This attribute specifies the remote DLCEP address and identifies the DLCEP.

Role
This attribute specifies the role of this AREP. The value of “Publisher” indicates that this AREP is used as a publisher. The value of “Subscriber” indicates that this AREP is used as a subscriber.

9.11.4.2.3 DLL services

Refer to Clause 7 for DLL service descriptions.

9.11.4.3 MTU-ARPM state machine

9.11.4.3.1 MTU-ARPM states

The MTU-ARPM state machine has only one state called “ACTIVE” (see Figure 72).

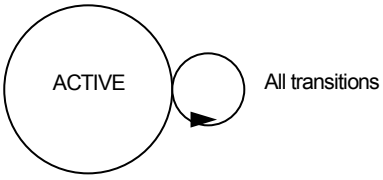


Figure 72 – State transition diagram of MTU-ARPM

9.11.4.3.2 MTU-ARPM state table

Tables 148 and 149 define the state machine of the MTU-ARPM.

Table 148 – MTU-ARPM state table – Sender transactions

#	Current state	Event or condition action	Next state

S2	ACTIVE	UCS_req
		&& Role = “Publisher”
		=>
		FAL-PDU_req {
		dlsap_id := DLSAP_ID,
		called_address := Remote_dlsap_address,
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := “UnconfirmedSend-CommandPDU,”
		fal_data := user_data)
		}
			ACTIVE

Table 149 – MTU-ARPM state table – Receiver transactions

#	Current state	Event or condition action	Next state

R2	ACTIVE	FAL-PDU_ind
		&& FAL_Pdu_Type (fal_pdu) = “UnconfirmedSend-CommandPDU “
		&& Role = “Subscriber”
		=>
		UCS_ind{
		remote_dlsap_address := calling_address,
		user_data := fal_pdu
		}
			ACTIVE

9.11.4.3.3 Functions used by MTU-ARPM

Decoding to derive the relevant parameters for the state machine always follows receipt of an FAL-PDU_ind primitive. This is an implicit function and is not listed separately.

Table 150 defines the other function used by this state machine.

Table 150 – Function BuildFAL-PDU

Name	BuildFAL-PDU	Used in	ARPM
Input		Output	
Service type data additional information		DLSDU	
Function			
Builds an FAL-PDU out of the parameters given as input variables			

9.12 DLL mapping protocol machine

9.12.1 Primitive definitions

9.12.1.1 Primitives exchanged between DMPM and ARPM

Tables 151 and 152 list the primitives exchanged between DMPM and ARPM.

Table 151 – Primitives issued by ARPM to DMPM

Primitive name	Source	Associated parameters	Functions
FAL-PDU_req	ARPM	dmpm-service-name arep-id local-dlcep-identifier reason DLSDU	This primitive is used to request the DMPM to transfer an FAL-PDU. It passes the FAL-PDU to the DMPM as a DLSDU. It also carries some of the DLL parameters that are referenced there

Table 152 – Primitives issued by DMPM to ARPM

Primitive name	Source	Associated parameters	Functions
FAL-PDU_ind	DMPM	DLSDU	This primitive is used to pass an FAL-PDU received as a DLL service data unit to a designated ARPM

9.12.1.2 Parameters of ARPM/DMPM primitives

The DLSDU parameter contains the data of the application process and all relevant information for the state machine. The DMPM state machine is able to extract this information.

9.12.1.3 Primitives exchanged between DLL and DMPM

Tables 153 and 154 list the primitives exchanged between the DLL and the DMPM.

Table 153 – Primitives issued by DMPM to DLL

Primitive name	Source	Associated parameters
DL-DATA.req	DMPM	dl_dls_user_data

Table 154 – Primitives issued by DLL to DMPM

Primitive name	Source	Associated parameters
DL-DATA.ind	DLL	dl_dls_user_data

9.12.1.4 Parameters of DMPM/DLL primitives

The parameters used with the primitives exchanged between the DMPM and the DLL are defined in the DLL Service definition (see Clause 7). They are prefixed by "dl_" to indicate that they are used by the FAL.

9.12.2 DMPM state machine

9.12.2.1 DMPM states

The DMPM state machine has only one state called "ACTIVE" (see Figure 73).

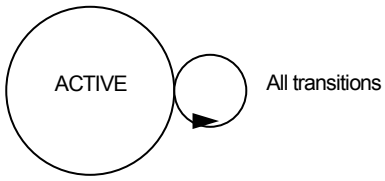


Figure 73 – State transition diagram of DMPM

9.12.2.2 DMPM state table

Tables 155 and 156 define the DMPM state machine.

Table 155 – DMPM state table – Sender transactions

#	Current state	Event or condition action	Next state

S1	ACTIVE	FAL-PDU_req
		
		DL-DATA.req {
		dl_dls_user_data := DLSDU
		}
			ACTIVE

Table 156 – DMPM state table – Receiver transactions

#	Current state	Event or condition action	Next state
R1	ACTIVE	DL-DATA.ind FAL_PDU_ind	ACTIVE

9.12.2.3 Functions used by DMPM

Decoding to derive relevant parameters for the state machine always follows receipt of a DL-DATA.ind or a DL-DATA.ind primitive. This is an implicit function and is not listed separately.

Annex A

Data link layer technical description

In this annex, the RAPIEnet network management mechanism in the data link layer (DLL) is described with detailed examples.

A.1 DL-address collision

A.1.1 General

A RAPIEnet DL-address is configured manually by hardware settings (for example, rotary switch) or set by software (for example, portable terminal). The data link address (DL-address) may be duplicated in other devices on the network because of a configuration error. When this happens, the device is unable to communicate with other devices. This type of DL-address collision event on the network is detected automatically by data link management (DLM), and the local data link management service user (DLMS-user) is notified.

A.1.2 Detecting DL-address collision

Figure A.1 shows an example of DL-address collision in a ring network. Devices1, 5, and 6 have the same DL-address value of 1. Device2 and Device7 have the same DL-address value of 2. Therefore, 3 DL-address collision events are detected by the DLM. DL-address collision events are counted by the collision counters in the network management information base (NMIB) and shared with every device on the network.

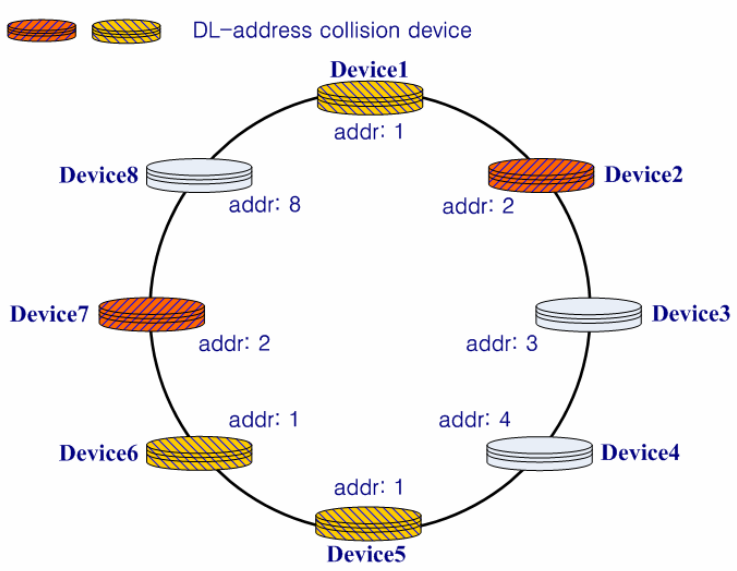


Figure A.1 – RAPIEnet DL-address collision in a ring network

Figure A.2 shows an example of DL-address collision in a line network. The DL-addresses are set the same as in Figure A.1. Three DL-address collision events are also detected and counted by the collision counters in the NMIB and shared with every device on the network.

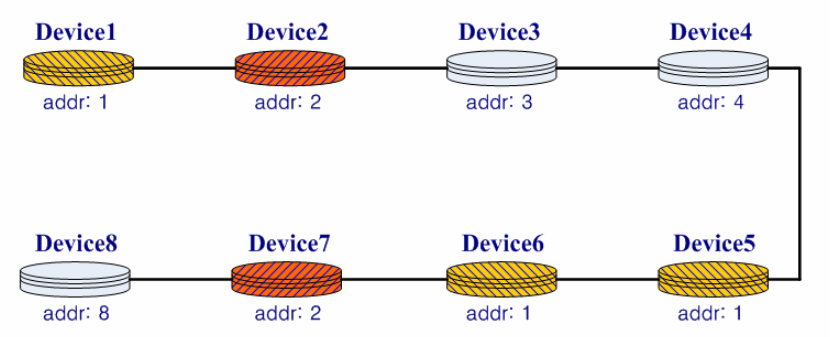


Figure A.2 – RAPIEnet DL-address collision in a line network

Table A.1 shows the DL-address collision information for the situations in Figures A.1 and A1-2 representing the DL-address collision detection mechanism using the device UIDs.

Table A.1 – DL-address collision information

Device name	DL-address	MAC address	UID	Collision
Device1	1	0x002233445511	0x0001002233445511	Collision
Device2	2	0x002233445522	0x0002002233445522	Collision
Device3	3	0x002233445533	0x0003002233445533	-
Device4	4	0x002233445544	0x0004002233445544	-
Device5	1	0x002233445555	0x0001002233445555	Collision
Device6	1	0x002233445566	0x0001002233445566	Collision
Device7	2	0x002233445577	0x0002002233445577	Collision
Device8	8	0x002233445588	0x0008002233445588	-

The DL-address is configured manually by the operator to be some value in the range of 0–255. The MAC address is a 6-byte unique ISO/IEC 8802-3 Ethernet MAC address. The RAPIEnet device UID is composed of the 2-byte RAPIEnet DL-address and the 6-byte MAC address. Therefore, the RAPIEnet Device UID has a unique 8-byte value on the network that cannot be duplicated. The RAPIEnet device UID is used to recognize a specific device on the network.

A RAPIEnet DL-address collision is detected by the DLM when devices with different device UIDs are configured with the same DL-address. If a local DL-address collision is detected, the DLM sets the DL-address Collision flag in the Device Flags and generates a DL-address Collision event. If a network DL-address Collision is detected, the DLM sets the DL-address Collision in the Network Flags and generates a Network DL-address Collision event. Figure A.3 shows the DL-address collision detection procedures of the DLM.

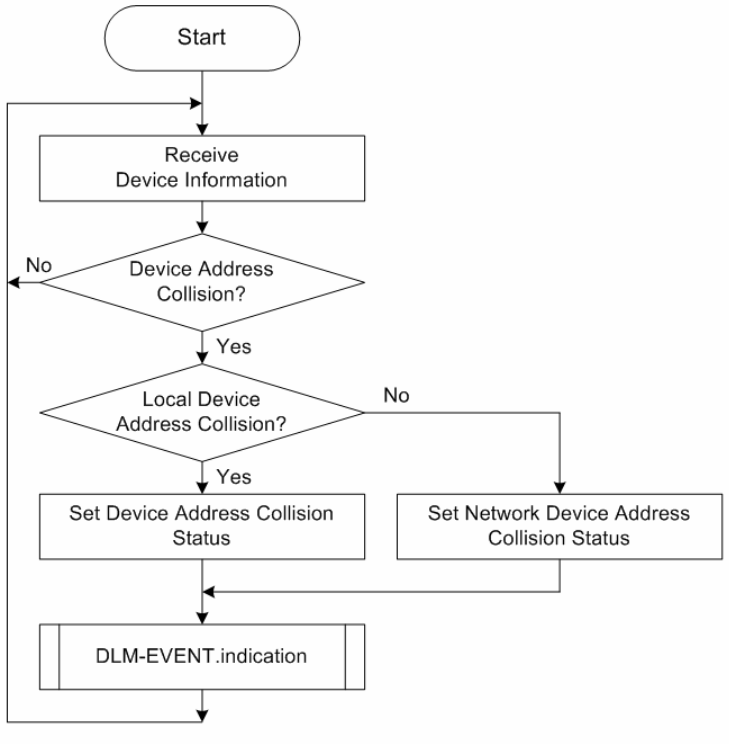


Figure A.3 – DL-address collision detection procedure

A.1.3 Clearing DL-address collision

To release the DL-address collision, the device with a duplicated DL-address should be disconnected from the network and reconnected with a new DL-address. When a device is disconnected from the network, the corresponding device's information in the path table is cleared, so when the duplicated devices are disconnected from the network, the DL-address collision is also cleared.

Figure A.4 shows a network separation example where the single line network shown in Figure A.2 is divided into two separate line networks. As the link between Device4 and Device5 is disconnected, the DL-address collision of Device1 and Device2 in Line Network1 and the DL-address collision of Device7 in Line Network2 are cleared. However, the DL-address collision of Device5 and Device6 in Line Network2 is not cleared.

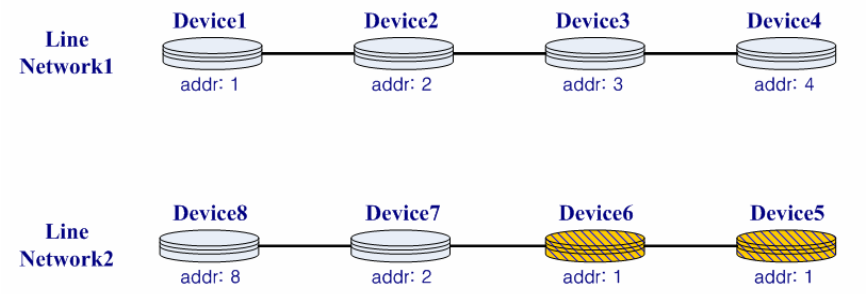


Figure A.4 – DL-address collision clearing example

Figure A.5 shows the network-triggered events and DL-address collision-clearing procedures.

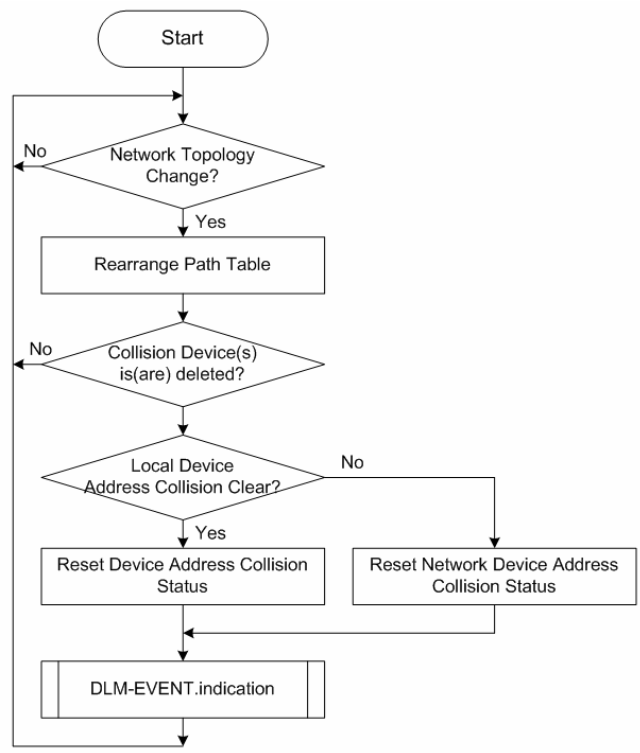


Figure A.5 – DL-address collision clearing procedure

When the device that has the same DL-address as the local device is disconnected from the network and its path table entry is cleared, the DLM clears the DL-address Collision Status and sends the DL-address Collision Clear event to the local DLMS-user using the DLM-EVENT indication service primitive. When the device that has the duplicated DL-address is disconnected from the network and its path table entry is cleared, the DLM clears the Network DL-address Collision Status and sends the Network DL-address Collision Clear event to the local DLMS-user using the DLM-EVENT indication service primitive (see 8.7.4.3).

When the device is reconnected to the network with a new DL-address, the path table and network information are updated according to the procedures in Figure A.5 (see 8.7.4.1).

A.2 Automatic Ring Network Manager (RNM) election procedure

A.2.1 General

In a ring network, a single frame can circulate continuously if the designated destination device is not found or when the frame is broadcast on the network. In a RAPIEnet ring network, the primary RNM (RNMP) and the secondary RNM (RNMS) are selected automatically to prevent infinite frame circulation using the following four steps.

a) The device with the highest Device UID value is selected automatically as the RNMP.

b) The RNMP sends an NCM_RING_START message including the RNMS assignment request to the neighbouring device connected through its R-port2.

c) The RNMS replies with an NCM_ACK_RNMS message to the RNMP.

d) Automatic ring network configuration is completed with RNMP and RNMS.

A.2.2 Primary RNM (RNMP)

A RAPIEnet device realizes that the network is configured as a ring topology when the network control message generated by the device itself is received through the other port. In a ring network, the device with the highest Device UID value is selected as the RNMP. When a RAPIEnet device detects that the network is configured as a ring network, each device tries to find the device in the path table with the highest Device UID value. Therefore, contention to select the RNMP is not necessary in RAPIEnet. If a remote device is selected as the RNMP, the other devices wait for the NCM_RING_START message from it. If a local device is selected as the RNMP, the DLM disables both frame forwarding functions and generates an NCM_RING_START message including the RNMS information of the device connected through R-port2 of the RNMP.

When the RNMP receives the NCM_ACK_RNMS message from the RNMS, RNMP disables the frame forwarding function in the RNMS direction but keeps the opposite direction enabled. The RNMS disables the frame forwarding function in the RNMP direction and enables it in the opposite direction.

NOTE Device UID has a unique value on the network. Therefore, the RNMP and RNMS are selected automatically even in the case of a DL-address collision.

A.2.3 Secondary RNM (RNMS)

The RNMS is assigned by the RNMP. When a GD state device receives an NCM_RING_START message from the RNMP, the DLM checks if the local device's Device UID is equal to the value of the RNMS Device UID in the received NCM_RING_START message. If the device is not designated as the RNMS, the DLM enables both frame forwarding functions in the GD state device. If the device is designated as the RNMS, the DLM enters RNMS state and transmits NCM_ACK_RNMS to the RNMP through the destination R-port for the RNMP in the path table. In addition, the RNMS device disables the frame forwarding function in the RNMP direction but enables it in the opposite direction.

A.3 Path management

A.3.1 General

The DLM also provides path information to the local DLS-user. Path management is calculated from the hop count. The hop count indicates how many frame forward operations are required to transfer a frame to the destination device.

A.3.2 Path of line topology network

In a line network, only one path is possible to the destination device. Figure A.6 shows an example of path management in a line network.

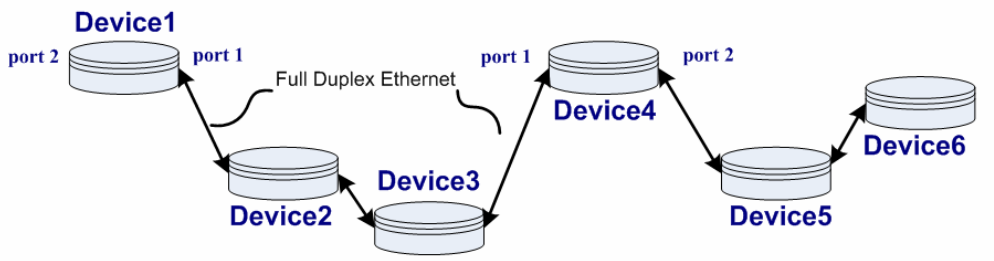


Figure A.6 – Path management in a line topology

Table A.2 shows the path table of Device1, and Table A.3 shows the path table of Device4 in Figure A.6. In a line network, only one path is possible between any two devices.

Table A.2 – Path table of Device1 in a line topology

Destination / R-port	Device2	Device3	Device4	Device5	Device6
R-port1	0 hop	1 hop	2 hops	3 hops	4 hops
R-port2	Invalid.	Invalid	Invalid	Invalid	Invalid
Preferred Port	R-port1	R-port1	R-port1	R-port1	R-port1
Destination Port	R-port1	R-port1	R-port1	R-port1	R-port1

Table A.3 – Path table of Device4 in a line topology

Destination / R-port	Device1	Device2	Device3	Device5	Device6
R-port1	2 hops	1 hop	0 hop	Invalid	Invalid
R-port2	Invalid	Invalid	Invalid	0 hop	1 hop
Preferred Port	R-port1	R-port1	R-port1	R-port2	R-port2
Destination Port	R-port1	R-port1	R-port1	R-port2	R-port2

A.3.3 Path of ring topology network

In a ring network, two paths are possible between any two devices: the clockwise path and the counter-clockwise path. However, a frame cannot be forwarded across the RNMP or RNMS. Therefore, it is impossible to transfer a frame using the path including the RNMP or RNMS. Figure A.7 shows an example of path management in a ring network.

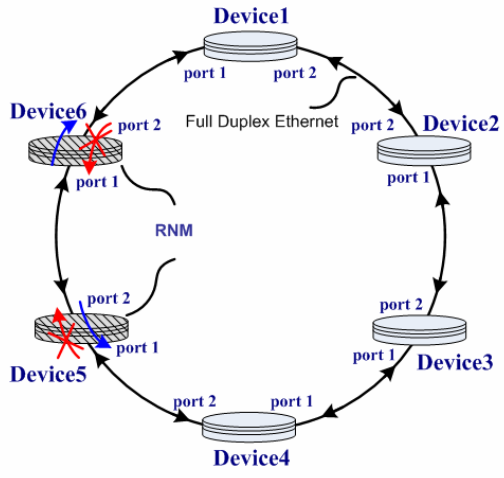


Figure A.7 – Path management in a ring network

Table A.4 shows the path table of Device1 in Figure A.7. The shortest path from Device1 to Device5 is in the R-port1 direction but this path is blocked by the RNM, Device6. In this case, the destination path is determined to be in the R-port2 direction.

Table A.4 – Path table of Device1 in a ring topology

Destination / R-port	Device2	Device3	Device4	Device5 (RNM)	Device6 (RNM)
R-port1	4 hops	3 hops	2 hops	1 hop	0 hop
R-port2	0 hop	1 hop	2 hops	3 hops	4 hops
Preferred Port	R-port2	R-port2	Don't care	R-port1 (R-port1 direction is blocked by Device6)	R-port1
Destination Port	R-port2	R-port2	R-port1(NOTE)	R-port2	R-port1

NOTE If both paths have the same hop counts and they are not blocked by the RNMs, R-port1 direction is chosen as the dominant path.

Table A.5 shows the path table of Device3 in Figure A.7.

Table A.5 – Path table of Device3 in a ring topology

Destination / R-port	Device1	Device2	Device4	Device5 (RNM)	Device 6 (RNM)
R-port1	3 hops	4 hops	0 hop	1 hop	2 hops
R-port2	1 hop	0 hop	4 hops	3 hops	2 hops
Preferred Port	R-port2	R-port2	R-port1	R-port1	Don't care (R-port1 direction is blocked by Device5)
Destination Port	R-port2	R-port2	R-port1	R-port1	R-port2

A.4 Extremely fast network recovery

RAPIEnet provides extremely fast recovery time for network topology changes. When a link failure is detected in a ring network, the network is reconfigured automatically as a line network. Topology change from a ring to a line network is processed according to one of the following two cases.

a) Link fault with the neighbouring device.

b) Link fault of the remote device.

A.4.1 Link fault with neighbouring device

Figure A.8 shows an example of a link fault with the neighbouring device in a ring network. If the link between Device1 and Device2 is disconnected when Device1 tries to send a frame to Device3, the link fault event is triggered spontaneously by the hardware signal, and it is detected by the DLM in Device1. Then, Device1 realizes that the link is not available for data transmission and the ring network should be reconfigured as a line network. Device1 broadcasts an NCM_LINE_START message and modifies the destination R-port to Device3 as the other R-port. Device1 transmits the frame through the new destination R-port. A cable break or power failure is treated in the same way as a link fault. Redundancy recovery time is determined from the time when a link fault occurs to the time when the path is recovered in a new direction.

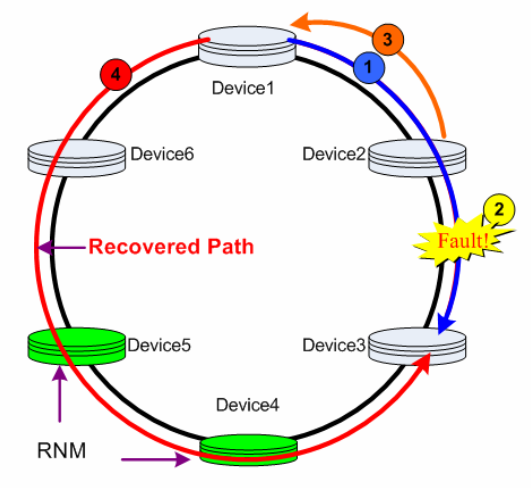


Figure A.8 – Link fault with neighbouring device

A.4.2 Link fault of remote device

Figure A.9 shows an example of a link fault of the remote device in a ring network. If the link between Device2 and Device3 is disconnected, when Device1 tries to send a frame to Device3, the link fault event is triggered spontaneously by the hardware signal, and is detected by the DLM in Device2. At this point, Device2 broadcasts an NCM_LINE_START message indicating that the link is not available, and the ring network should be reconfigured as a line network. Device1 modifies the destination R-port to Device3 as the other R-port and transmits the frame through the new R-port. Redundancy recovery time is determined from the time when a link fault occurs to the time when the path is recovered in a new direction.

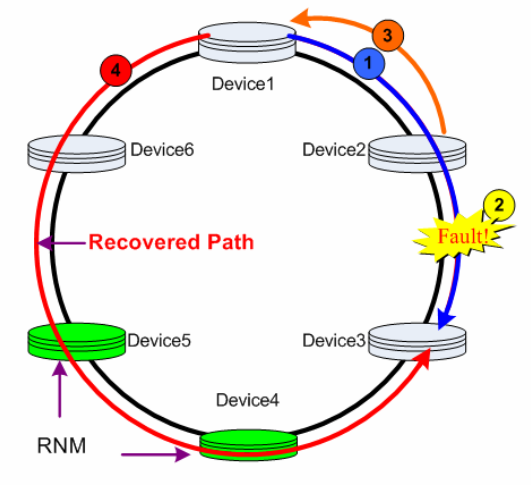


Figure A.9 – Link fault of remote device

British Standards Institute (BSI)

BSI is the independent national body responsible for preparing British Standards. It presents the UK view on standards in Europe and at the international level. It is incorporated by Royal Charter.

Revisions

British Standards are updated by amendment or revision. Users of British Standards should make sure that they possess the latest amendments or editions.

It is the constant aim of BSI to improve the quality of our products and services. We would be grateful if anyone finding an inaccuracy or ambiguity while using this British Standard would inform the Secretary of the technical committee responsible, the identity of which can be found on the inside front cover.
Tel: +44 (0)20 8996 9000 Fax: +44 (0)20 8996 7400

BSI offers members an individual updating service called PLUS which ensures that subscribers automatically receive the latest editions of standards.

Buying standards

Orders for all BSI, international and foreign standards publications should be addressed to Customer Services.
Tel: +44 (0)20 8996 9001 Fax: +44 (0)20 8996 7001
Email: orders@bsigroup.com

You may also buy directly using a debit/credit card from the BSI Shop on the Website http://www.bsigroup.com/shop.

In response to orders for international standards, it is BSI policy to supply the BSI implementation of those that have been published as British Standards, unless otherwise requested.

Information on standards

BSI provides a wide range of information on national, European and international standards through its Library and its Technical Help to Exporters Service. Various BSI electronic information services are also available which give details on all its products and services. Contact the Information Centre.
Tel: +44 (0)20 8996 7111 Fax: +44 (0)20 8996 7048
Email: info@bsigroup.com

Subscribing members of BSI are kept up to date with standards developments and receive substantial discounts on the purchase price of standards. For details of these and other benefits contact Membership Administration.
Tel: +44 (0)20 8996 7002 Fax: +44 (0)20 8996 7001
Email: membership@bsigroup.com

Information regarding online access to British Standards via British Standards Online can be found at http://www.bsigroup.com/BSOL.

Further information about BSI is available on the BSI website at http://www.bsigroup.com.

Copyright

BSI Group Headquarters
389 Chiswick High Road,
London W4 4AL, UK
Tel +44 (0)20 8996 9001
Fax +44 (0)20 8996 7001
www.bsigroup.com/standards