PD CLC/TS 52056-8-4:2015

BSI Standards Publication

Electricity metering data exchange — The DLMS/COSEM suite

Part 8-4: Narrow-band OFDM PRIME PLC communication profile for neighbourhood networks

bsi.

...making excellence a habit.™

National foreword

This Published Document is the UK implementation of CLC/TS 52056-8-4:2015.

The UK participation in its preparation was entrusted to Technical Committee PEL/13, Electricity Meters.

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.

Published by BSI Standards Limited 2015

ISBN 978 0 580 85916 8
ICS 35.240.60; 91.140.50

Compliance with a British Standard cannot confer immunity from legal obligations.

This Published Document was published under the authority of the Standards Policy and Strategy Committee on 31 May 2015.

Amendments/corrigenda issued since publication

Date	Text affected

TECHNICAL SPECIFICATION

SPÉCIFICATION TECHNIQUE

TECHNISCHE SPEZIFIKATION

CLC/TS 52056-8-4

April 2015

ICS 35.240.60; 91.140.50

English Version

Electricity metering data exchange - The DLMS/COSEM suite - Part 8-4: Narrow-band OFDM PRIME PLC communication profile for neighbourhood networks

This Technical Specification was approved by CENELEC on 2014-11-11.

CENELEC members are required to announce the existence of this TS in the same way as for an EN and to make the TS available promptly at national level in an appropriate form. It is permissible to keep conflicting national standards in force.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Bulgaria, Croatia, Cyprus, the Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

CEN-CENELEC Management Centre: Avenue Marnix 17, B-1000 Brussels

Ref. No. CLC/TS 52056-8-4:2015 E

CONTENTS

List of Figures

List of tables

Foreword

This document (CLC/TS 52056-8-4:2015) has been prepared by CLC/TC 13 "Equipment for electrical energy measurement and load control".

The following date is fixed:

- latest date by which the existence of this document has to be announced at national level (doa) 2015-07-24

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CENELEC [and/or CEN] shall not be held responsible for identifying any or all such patent rights.

This document has been prepared under a mandate given to CENELEC by the European Commission and the European Free Trade Association.

Introduction

This Technical Specification is based on the results of the European OPEN Meter project, Topic Energy 2008.7.1.1, Project no.: 226369, www.openmeter.com, and has been prepared by the PRIME Alliance Technical Working Group, www.prime-alliance.org .

1 Scope

This Technical Specification is part of the EN 62056 / 52056 DLMS/COSEM suite and it specifies the DLMS/COSEM communication profiles for power line carrier neighbourhood networks using the modulation specified in ITU-T G.9904:2012.

There are three profiles specified:

- a profile using the EN 61334-4-32:1996 LLC layer;
- a profile using TCP-UDP/IPv4;
- a profile using TCP-UDP/IPv6.

2 Normative references

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

EN 50065-1, *Signalling on low-voltage electrical installations in the frequency range 3 kHz to 148.5 kHz - Part 1: General requirements, frequency bands and electromagnetic disturbances*

EN 61334-4-1:1996, *Distribution automation using distribution line carrier systems – Part 4: Data communication protocols – Section 1: Reference model of the communication system (IEC 61334-4-1:1996)*

EN 61334-4-32:1996, *Distribution automation using distribution line carrier systems – Part 4: Data communication protocols – Section 32: Data link layer – Logical link control (LLC) (IEC 61334-4-32:1996)*

EN 61334-4-511:2000, *Distribution automation using distribution line carrier systems – Part 4-511: Data communication protocols – Systems management – CIASE protocol (IEC 61334-4-511:2000)*

FprEN 62056-4-7:2014, *Electricity metering data exchange - The DLMS/COSEM suite – Part 4-7: DLMS/COSEM transport layer for IP networks (IEC 62056-4-7:2015)*

EN 62056-5-3, *Electricity metering data exchange – The DLMS/COSEM suite – Part 5-3: DLMS/COSEM application layer (IEC 62056-5-3)*

EN 62056-6-1, *Electricity metering data exchange – The DLMS/COSEM suite – Part 6-1: Object identification system (OBIS) (IEC 62056-6-1)*

EN 62056-6-2, *Electricity metering data exchange – The DLMS/COSEM suite – Part 6-2: COSEM interface classes (IEC 62056-6-2)*

EN 62056-9-7:2013, *Electricity metering data exchange – the DLMS/COSEM suite – Part 9-7: Communication profile for TCP-UDP/IP networks (IEC 62056-9-7:2013)*

Recommendation ITU-T G.9904:2012, *SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS Access networks – In premises networks. Narrowband orthogonal frequency division multiplexing power line communication transceivers for PRIME networks*

RFC 2460 *Internet Protocol, Version 6 (IPv6) Specification*
Authors: S. Deering, Cisco, R. Hinden Nokia
Date: December 1998
Available from: http://www.ietf.org/rfc/rfc2460.txt

RFC 2464 *Transmission of IPv6 Packets over Ethernet Networks*
Authors M. Crawford Fermilab
Date: December 1998

Available from: http://www.ietf.org/rfc/rfc2464.txt

RFC 4291 *IP Version 6 Addressing Architecture*
Authors R. Hinden Nokia, S. Deering Cisco Systems
Date: February 2006.
Available from: http://www.ietf.org/rfc/rfc4291.txt

RFC 6282 *Compression Format for IPv6 Datagrams over IEEE 802.15.4–Based Networks*
Authors J. Hui, Ed. Arch Rock Corporation P. Thubert Cisco
Date: September 2011.
Available from: http://www.ietf.org/rfc/rfc6282.txt

RFC 4862 *IPv6 Stateless Address Configuration*
Authors S. Thomson, Cisco, T. Narten IBM, T. Jinmei, Toshiba
Date: September 2007.
Available from: www.ietf.org/rfc/rfc4862.txt

RFC 3315 *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*
Authors R. Droms, E J. Bound, B. Volz, T. Lemon, C. Perkins, M. Carney
Date: July 2003
Available from: www.ietf.org/rfc/rfc3315.txt

3 Abbreviations

AA	Application Association
AARE	Application Association Response
AARQ	Application Association Request
ACSE	Application Control Service Element
AL	Application Layer
AP	Application Process
APDU	Application Protocol Data Unit
ARQ	Automatic Repeat Request
CENELEC	European Committee for Electrotechnical Standardization
CL	Convergence Layer
.cnf	Confirm service primitive
COSEM	Companion Specification for Energy Metering
CPCS	Common Part Convergence Sublayer
CSMA/CA	Carrier Sense Multiple Access – Collision Avoidance
D8PSK	Differential Eight-Phase Shift Keying
DBPSK	Differential Binary Phase Shift Keying
DGW	Default Gateway
DHCP	Dynamic Host Configuration Protocol
DLMS	Device Language Message Specification
DQPSK	Differential Quaternary Phase Shift Keying
EUI-48	48-bit Extended Unique Identifier
FU	Firmware Upgrade
FW	Firmware
IANA	Internet Assigned Numbers Authority
IGMP	Internet Group Management Protocol
.ind	Indication service primitive
IP	Internet Protocol
IPv4	Internet Protocol, version 4
IPv6	Internet Protocol version 6
LCID	Local Connection Identifier
LD	Logical Device
LLC	Logical Link Control (sub-layer)
LNID	Local Node Identifier
MAC	Medium Access Control, MAC sublayer entity
MLME	MAC Layer Management Entity
MPDU	MAC Protocol Data Unit
NAT	Network Address Translation
NHC	Next Header Compression

NL	Noise Level
OBIS	OBject Identification System
OFDM	Orthogonal Frequency Division Multiplexing
OSI	Open System Interconnection
PHY	Physical Layer entity
PLC	Power Line Communication
PIB	PLC Information Base
PLME	Physical Layer Management Entity
PPDU	PHY Protocol Data Unit
.req	Request service primitive
RFC	Request For Comment
.rsp	Response service primitive
SDU	Service Data Unit
SID	Switch Identifier
SNA	Subnetwork Address
SNR	Signal-to-Noise Ratio
SSCS	Service Specific Convergence Sublayer
TCP	Transmission Control Protocol
TOS	Type Of Service
UDP	User Datagram Protocol
xDLMS_ASE	extended DLMS Application Service Element
ZCT	Zero Crossing Time

4 Targeted communication environments

The *DLMS/COSEM narrow-band OFDM PLC profiles for PRIME networks* are intended for remote data exchange on Neighbourhood Networks (NN) between *Neighbourhood Network Access Points* (NNAP) and *Local Network Access Points* (LNAPs) or *End Devices* using OFDM technology over the low voltage electricity distribution network as a communication medium. The functional reference architecture is shown in Figure 1.

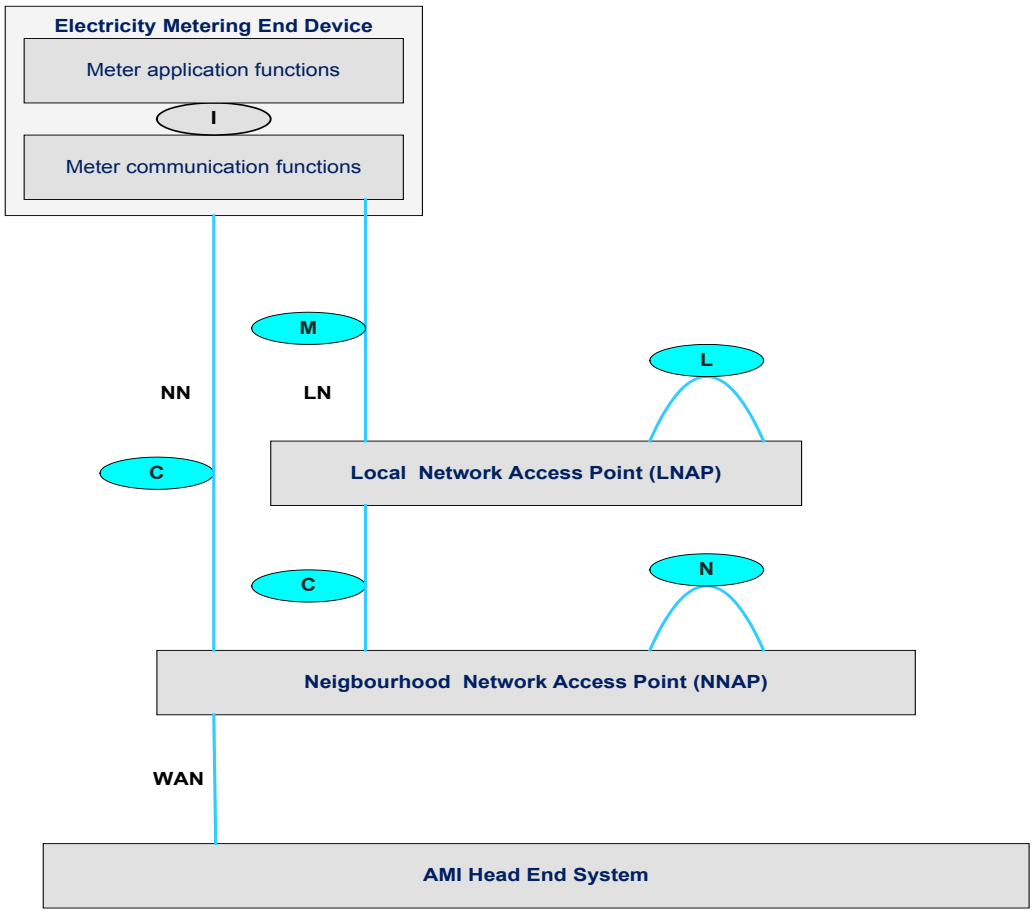


Figure 1 – Communication architecture

End devices – typically electricity meters – comprise application functions and communication functions. They may be connected directly to the NNAP via the C interface, or to an LNAP via an M interface, while the LNAP is connected to the NNAP via the C interface. The LNAP function may be co-located with the metering functions.

A NNAP comprises gateway functions and it may comprise concentrator functions. Upstream, it is connected to the Metering Head End System (HES) using suitable communication media and protocols.

End devices and LNAPs may communicate to different NNAPs, but to one NNAP only at a time. From the PLC communication point of view, the NNAP acts as the Base Node while end devices and LNAPs act as Service Nodes.

NNAPs and similarly LNAPs may communicate to each other, but this is out of the scope of this Technical Specification, which covers the C interface only.

When the NNAP has concentrator functions, it acts as a DLMS/COSEM client. When the NNAP has gateway functionality only, then the HES plays the role of a DLMS/COSEM client. The end devices or the LNAPs play the role of DLMS/COSEM servers.

A mixed architecture is also possible, i.e. both the HES and the NNAP can act as a client.

5 Reference model

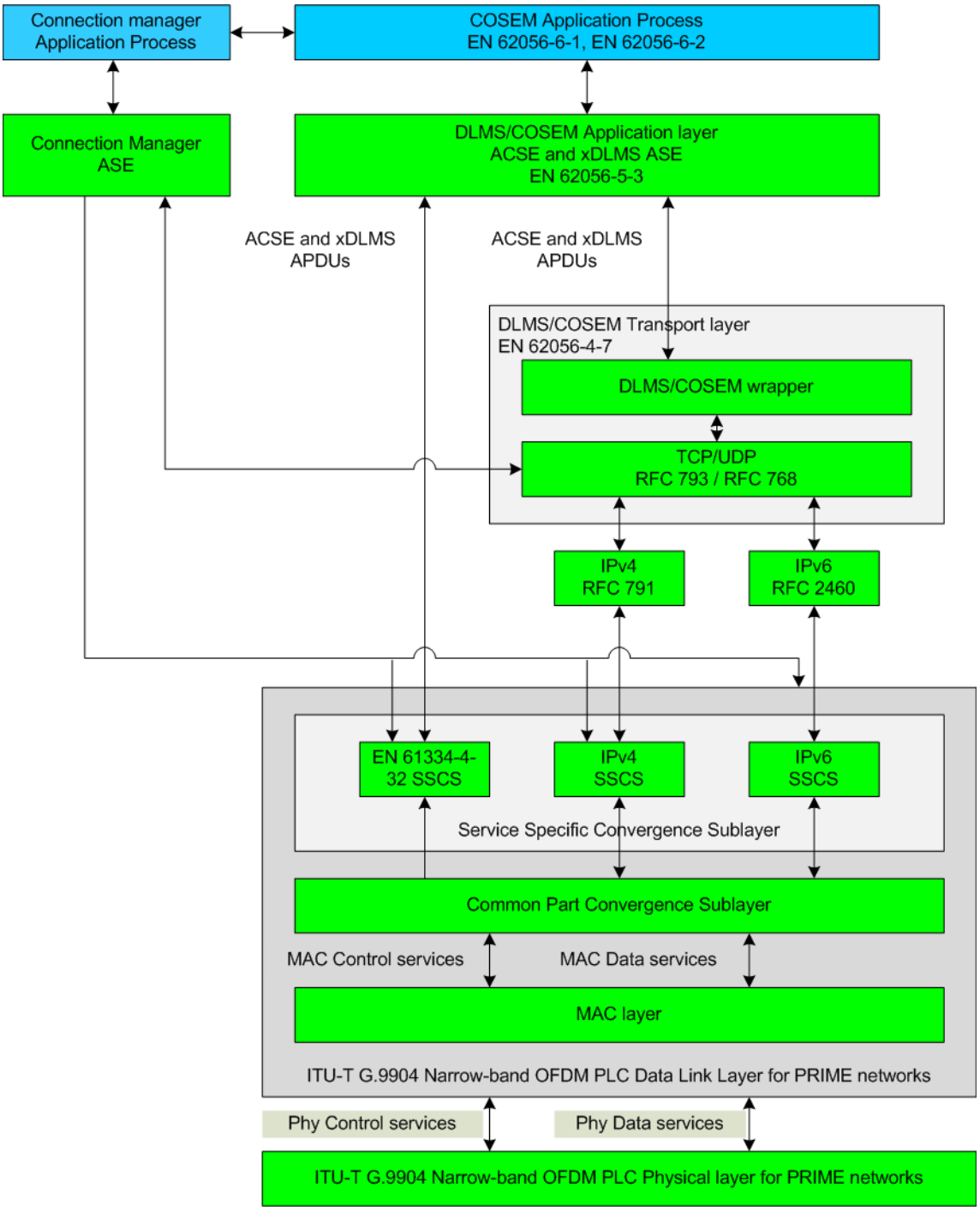
5.1 Overview

The proposed protocol stacks use the following OSI layers as shown in Figure 2.

- The DLMS/COSEM Application layer as specified in EN 62056-5-3 covering the Application, Presentation and Session functionalities;
- The LLC sublayer as specified in EN 61334-4-32:1996, used with the DLMS/COSEM 61334-4-32 profile over PRIME networks;
- The DLMS/COSEM transport layer as specified in FprEN 62056-4-7:2014, used with the DLMS/COSEM TCP-UDP/IPv4 and TCP-UDP/IPv6 profiles over PRIME networks;
- The PRIME Data link layer, that consists of the MAC sublayer, and the CPCS and the corresponding SSCS, according to the selected profile (EN 61334-4-32, TCP-UDP/IPv4 or TCP-UDP/IPv6);
- The PRIME Physical layer.

Following this reference model, three distinct profiles can be identified, all of them using the PRIME PHY, MAC sublayer as lower layers and the Common Part Convergence Sublayer on one hand, and the DLMS/COSEM Application layer specified in EN 62056-5-3 and the COSEM object model specified in EN 62056-6-1 and EN 62056-6-2 on the other hand. Lower layers meaning PHY and MAC are based on IEEE 802.15.4 principle.

NOTE The COSEM interface classes for setting up and managing data exchange over the narrow-band OFDM PRIME PLC network are specified in EN 62056-6-2



NOTE In the case of the TCP-UDP/IPv4 and TCP-UDP/IPv6 profiles the Connection Manager entity is the TCP and IPv4 SSCS / IPV6 SSCS Connection Manager. In this case the valid link is from the Connection Manager entity to the TCP / IPV4 SSCS / IPV6 SSCS entity.

In the case of the EN 61334-4-32 profile the Connection Manager entity is the EN 61334-4-32 connection manager. In this case the valid link is from the Connection Manager entity to the EN 61334-4-32 SSCS entity.

Figure 2 – PLC PRIME protocol architecture

5.2 The EN 61334-4-32 profile

The EN 61334-4-32 profile uses the PRIME EN 61334-4-32 SSCS making the necessary adaptation between the PRIME PHY and MAC layers and the DLMS/COSEM application layer.

5.3 The TCP-UDP/IPv4 profile

The TCP-UDP/IPv4 profile uses the PRIME IPv4 SSCS making the necessary adaptation between the PRIME PHY and MAC layers and the IPv4 layer, supporting the DLMS/COSEM transport layer and application layer.

5.4 The TCP-UDP/IPv6 profile

The TCP-UDP/IPv6 profile uses the PRIME IPv6 SSCS making the necessary adaptation between the PRIME PHY and MAC layers and the IPv6 layer, supporting the DLMS/COSEM transport layer and application layer.

6 Physical Layer (PHY)

6.1 General

This layer provides the interface between the equipment and the physical transmission medium that is the electricity distribution network. It transmits and receives MPDUs between neighbour nodes. The physical layer uses OFDM modulation in the CENELEC A-band. See EN 50065-1. This band covers the frequency range from 3 kHz up to 95 kHz and its use is restricted to electricity suppliers and their licensees. The OFDM signal uses a frequency bandwidth of 47,363 kHz located on the high end of the CENELEC A-Band.

The OFDM signal itself uses 97 (96 data plus one pilot) equally spaced subcarriers with a short cyclic prefix. Differential modulation schemes are used, with three possible constellations: DBPSK, DQPSK or D8PSK. ½ rate convolutional coding, then an additive scrambler is used to avoid the occurrence of long sequences of identical bits, and finally interleaving is applied. Convolutional encoding can be disabled by the higher layers if the channel is good enough and higher throughputs are needed.

6.2 PRIME PHY data plane services

PHY DATA services are generated / used by the MAC layer entity whenever data – PPDUs – have to be transmitted to / received from (a) peer MAC entity(ies) using the PHY transmission procedures. See ITU-T G.9904:2012 clause 7.10.2.

6.3 PRIME PHY control plane services

PRIME PHY control plane services are used to control the physical layer by the MAC layer. See ITU-T G.9904:2012 clause 7.10.3. They are the following:

- PHY_AGC: allows the MAC layer entity to set or get the Automatic Gain Mode of the PHY;
- PHY_TIMER: allows the MAC layer entity to get the time at which the transmission has to be started;
- PHY_CD: allows the MAC layer entity to look for the carrier detect signal, in order to detect if the physical medium is free;
- PHY_NL: allows the MAC layer entity to get the floor noise level value present on the power line;
- PHY_SNR: allows the MAC layer entity to get the value of the signal-to-noise ratio, in order to find the appropriate degree of robustness needed for data exchange;
- PHY_ZCT: allows the MAC layer entity to get the zero crossing time of the mains and the time between the last transmission or reception and the zero crossing of the mains.

6.4 PRIME PHY management plane services

PRIME PHY management plane services are used to manage the physical layer by the MAC layer. They are described below:

- PLME_RESET: allows the MAC layer entity to request the PHY layer to reset its present functional state. As a result of this primitive, the PHY should reset all internal states and flush all buffers to clear any queued receive or transmit data;

- PLME_SLEEP: allows the MAC layer entity to request the PHY layer to suspend its present activities including all reception functions. The PHY layer should complete any pending transmission before entering into a sleep state;
- PLME_RESUME: allows the MAC layer entity to request the PHY layer to resume its suspended activities. As a result of this primitive, the PHY layer should start its normal transmission and reception functions;
- PLME_TESTMODE: allows the MAC layer entity to put the PHY layer into some non-default functional modes. Specific functional mode out of the various possible modes is provided as an input parameter. Following the reception of this primitive, the PHY layer should complete any pending transmissions in its buffer before entering the test mode requested;
- PLME_GET: allows the MAC layer entity to query information about a given attribute of the PRIME Information Base.

7 Data link layer

7.1 Overview – main features and functions

A subnetwork is a tree with two types of nodes, the Base Node and Service Nodes. The Base Node is at the root of the tree and acts as the master node that provides the subnetwork with connectivity. There is one and only one Base Node in a subnetwork. Any other subnetwork node is a Service Node.

The Base Node manages the subnetwork resources and connections. This Base Node is initially the subnetwork itself and all other nodes should follow a registration process to enrol themselves on the subnetwork.

Service Nodes start in a “Disconnected” state and they try to find a Base Node or a Switch Node to register themselves to the subnetwork. After this, they become leaves of the tree.

Service Nodes may change their state dynamically from “Terminal” functions to “Switch” functions and vice-versa. These changes occur on the basis of certain pre-defined events on the network. Service nodes in “Switch” state become branch points of the tree, capable of switching their neighbours’ data to propagate connectivity.

The three functional states of Service Nodes are:

- **Disconnected**: all nodes are in this state initially or after a restart. In this state, a node is not capable to participate in the network. The primary function of a Service Node in this state is to search for an operational network in its proximity and try to register itself on it.
- **Terminal**: In this state a Service Node is part of the network and is capable of communicating its traffic by establishing connections, but it is not capable of switching the traffic of any other node. And finally:
- **Switch**: In this state a Service Node is capable of performing all Terminal functions. Additionally, it is capable of forwarding data to and from other devices on the subnetwork.

The events and associated processes that trigger changes from one functional state to another are registration, unregistration, promotion and demotion.

Other functions of the MAC layer are:

- Address resolution and broadcast and multicast addressing;
- CSMA/CA algorithm implementation;
- Promoting Service Nodes from “Terminal” state to “Switch” state or demoting them from the “Switch” state to “Terminal” state;
- Establishing direct connections from one Service Node to another;
- Packet aggregation;
- Security functions, such as encryption and security keys management;

- PHY robustness management in order to select the best modulation schema for a given situation;
- ARQ mechanism.

7.2 Services used by Base Node and Service Nodes

- MAC_ESTABLISH: is used to manage the connection establishment at MAC layer;
- MAC_RELEASE: is used to release a connection at MAC layer;
- MAC_JOIN: is used to join to a broadcast or multicast connection and allow the reception of such packets;
- MAC_LEAVE: is used to leave a broadcast or multicast connection;
- MAC_DATA: is used to send and receiving unicast, multicast or broadcast data.

7.3 Services used by Base Node signalling

- MAC_REDIRECT: is used to answer a MAC_ESTABLISH.indication primitive. It also redirects the connection from the Base Node to another Service Node on the subnetwork.

7.4 Management services

- MLME_REGISTER: is used to perform registration and to indicate when registration has been performed;
- MLME_UNREGISTER: is used to perform deregistration and to indicate when it has been performed;
- MLME_PROMOTE: is used to perform promotion of Service Nodes from the “Terminal“ state to “Switch” state and to indicate when it has been performed;
- MLME_DEMOTE: is used to perform demotion of Service Nodes in the “Switch“ state to “Terminal” state and to indicate when it has been performed;
- MLME_RESET: is used to reset the MAC into a known good status. After the reset the MAC is in the Disconnected functional state;
- MLME_GET: is used to retrieve individual values from the MAC such as statistics;
- MLME_LIST_GET: is used to retrieve a list of values from the MAC.
- MLME_SET: is used to set configuration values in the MAC.

8 Convergence layers

8.1 Overview

NOTE The following description is based on ITU-T G.9904:2012 clause 9.1.

The convergence layer is divided into two sublayers:

The Common Part Convergence Sublayer (CPCS) provides a set of generic services. The use of CPCS services is optional in that a SSCS will configure into its protocol stack those services, which are required from the CPCS, and omit services that are not required. In the present specification, the CPCS provides the following services to the different SSCSs: segmentation and reassembly.

The Service Specific Convergence Sublayer (SSCS) contains services that are specific to a given communication profile. There are four SSCSs specified to connect the MAC layer to the upper layer:

- The EN 61334-32 LLC Convergence Sublayer, see 8.2;
- The IPv4 Convergence Sublayer, see 8.3;
- The IPv6 Convergence Sublayer, see 8.4;
- The Null Convergence Sublayer.

NOTE The Null Convergence Sublayer is outside the Scope of this Technical Specification. Information can be found in ITU-T G.9904:2012.

8.2 The EN 61334-4-32 LLC convergence sublayer

8.2.1 General

The EN 61334-4-32 LLC convergence sublayer (SSCS) provides convergence functions for the communication profile using EN 61334-4-32:1996 LLC services. Implementations conforming to this SSCS shall offer all LLC services specified in this document. Additionally, the PRIME EN 61334-4-32 SSCS shall provide services that help the mapping of the connection-less LLC protocol into the connection-oriented nature of PRIME MAC, meaning that before data exchange can take place, a connection has to be established. See ITU-T G.9904:2012, 9.5.2.

Main features:

- A Service Node can only exchange data with the Base Node and not to other Service Nodes;
- Each EN 61334-4-32 SSCS session establishes a dedicated PRIME MAC connection for exchanging unicast data with the Base Node;
- The Service Node SSCS session is responsible for initiating this connection to the Base Node. The Base Node SSCS cannot initiate a connection to a Service Node;
- Each EN 61334-4-32 SSCS listens to a PRIME broadcast MAC connection dedicated to the transfer of EN 61334-4-32 broadcast data from the Base Node to the Service Nodes. This broadcast connection is used when the Base Node application process using the EN 61334-4-32 communication profile makes a transmission request with the destination address used for broadcast or when the broadcast SAP functions are used. When there are multiple SSCS sessions within a Service Node, one PRIME broadcast MAC connection is shared by all the SSCS sessions.

The EN 61334-4-32 SSCS services are the following:

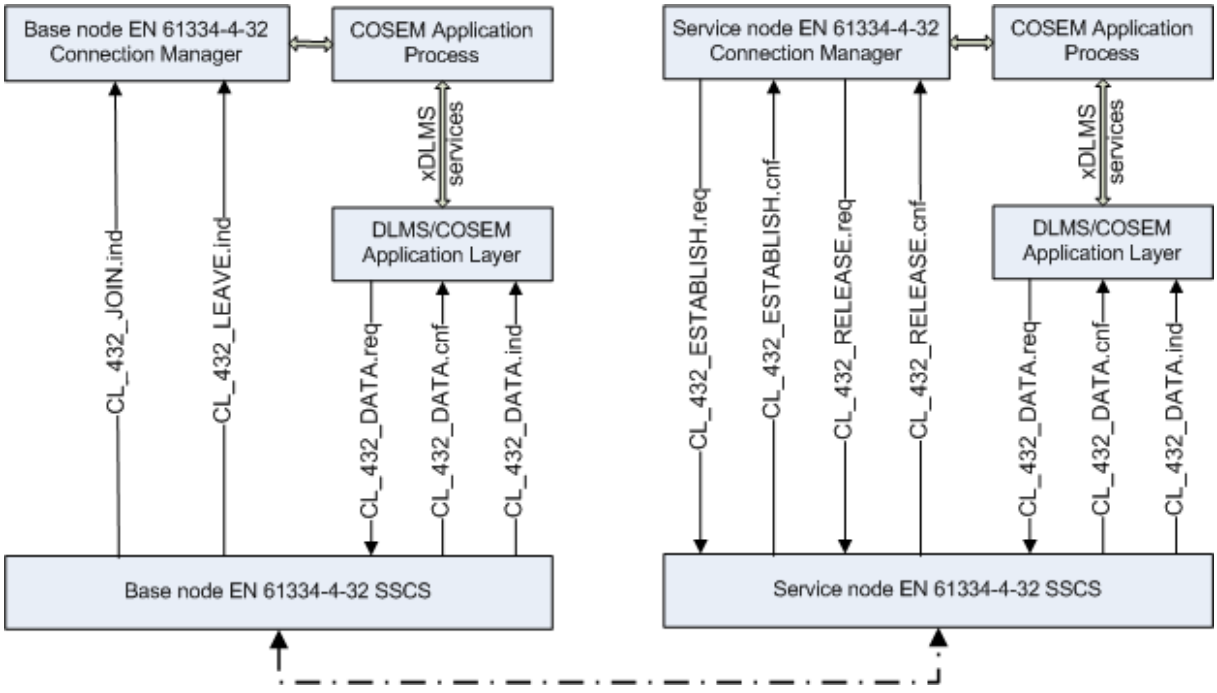


Figure 3 – EN 61334-4-32 SSCS services

8.2.2 Connection management services

8.2.2.1 CL_432_ESTABLISH.request

NOTE For details, see ITU-T G.9904:2012, 9.5.6.1.1.

Function

The purpose of this service is to request a connection opening with the Base Node.

Service parameters

The semantics of the primitive is the following:

```
CL_432_ESTABLISH.request
(
        Device_Identifier
)
```

Device_Identifier is the serial number of the Service Node.

Use

This service primitive is used by the Service Node EN 61334-4-32 Connection Manager to request a connection establishment with the Base Node.

8.2.2.2 CL_432_ESTABLISH.confirm

NOTE For details, see ITU-T G.9904:2012, 9.5.6.1.2.

Function

The purpose of this service is to inform the Service Node EN 61334-4-32 Connection Manager about the result of the previous CL_432_ESTABLISH.request

Service parameters

The semantics of the primitive is the following:

```
CL_432_ESTABLISH.confirm
(
        Device_Identifier,
        Destination_Address,
        Base_Address
)
```

Device_Identifier is the serial number of the Service Node. It is the same value than the one carried by the CL_432_ESTABLISH.req primitive.

Destination_Address is the address assigned by the Base Node to the concerned Service Node during the connection establishment process. Its scope is the subnetwork managed by the Base Node. During the registration process, the Base Node allocates a unique subnetwork destination address to the Service Nodes. For the scope of this address, refer to EN 61334-4-1:1996, 4.3.

The Base_Address is the address assigned to the Base Node.

Use

On receiving the connection establishment confirm primitive, the EN 61334-4-32 SSCS session confirms to the Service Node EN 61334-4-32 Connection Manager that the EN 61334-4-32 SSCS session has been opened and it indicates the Destination_Address allocated to the Service Node EN 61334-4-32 SSCS session, as well as the address of the Base Node.

The Base Node EN 61334-4-32 SSCS is responsible for allocating these addresses dynamically and associating the Device identifier of the Service Node EN 61334-4-32 SSCS session device with the allocated Destination_Address, according to EN 61334-4-511.

The Service Node also opens a MAC broadcast connection if no other EN 61334-4-32 SSCS session has already opened such a broadcast connection. This connection is used to receive broadcast packets sent by the Base Node EN 61334-4-32 SSCS to all Service Node EN 61334-4-32 SSCS sessions.

In the case the CL_432_ESTABLISH.request fails, a CL_432_Release.confirm primitive is sent back. The related reason of the failure is notified in the result.

8.2.2.3 CL_432_JOIN.indication

NOTE For details, see ITU-T G.9904:2012, 9.5.6.3.2.

Function

The purpose of this service is to inform the Base Node EN 61334-4-32 Connection Manager about a new connection establishment.

Service parameters

The semantics of the service is the following:

```
CL_432_JOIN.indication
(
        Device_Identifier,
        Destination_Address
)
```

Device_Identifier is the serial number of the Service Node, which joined the PRIME PLC subnetwork.

Destination_Address is the destination address provided by the Base Node SSCS to the Service Node which has requested the connection establishment.

See also 8.2.5.

Use

The Base Node EN 61334-4-32 SSCS uses this service to inform its EN 61334-4-32 Connection Manager that the concerned destination address is assigned to the Service Node identified by this Device_Identifier.

8.2.2.4 CL_432_RELEASE.request

NOTE For details, see ITU-T G.9904:2012, 9.5.6.1.3.

Function

This service primitive is passed from the Service Node EN 61334-4-32 Connection Manager to the EN 61334-4-32 SSCS. It is used to close the SSCS and to release any resource it may be holding.

Service parameters

The semantics of the service is the following:

```
CL_432_RELEASE.request
(
        Destination_Address
)
```

Destination_Address is the destination address provided by the Base Node SSCS to the Service Node during the connection establishment.

Use

This primitive is used by the Service Node EN 61334-4-32 Connection Manager to request the EN 61334-4-32 SSCS to close the connection.

8.2.2.5 CL_432_RELEASE.confirm

NOTE For details, see ITU-T G.9904:2012 9.5.6.1.4.

Function

This service primitive is passed from the Service Node EN 61334-4-32 SSCS to the EN 61334-4-32 Connection Manager to inform it that the SSCS has been closed. This could be due to a previous CL_432_RELEASE.request or due to an error occurred, forcing the closure of the SSCS.

Service parameters

The semantics of this primitive are as follows:

```
CL_432_RELEASE.confirm
(
        Destination_Address,
        Result
)
```

Destination_Address identifies the connection, which has been closed. The possible values of the Result parameter are specified in Table 1.

Use

This primitive is used by the Service Node EN 61334-4-32 SSCS to inform the EN 61334-4-32 Connection Manager about the result of the release request.

Table 1 – Result values for SSCS services

Result	Description
Success = 0	The SSCS service was successfully performed.
Reject = 1	The SSCS service failed because it was rejected by the Base Node.
Timeout = 2	A time out occurs during the SSCS service processing.
Not Registered = 6	The Service Node is not currently registered to a subnetwork.

8.2.2.6 CL_432_LEAVE.indicate

NOTE For details, see ITU-T G.9904:2012 9.5.6.3.3.

Function

This service primitive is used by the Base Node SSCS to inform the EN 61334-4-32 Connection Manager that the Service Node identified by the Destination_Address has left the subnetwork.

Service parameters

The semantics of this primitive are as follows:

```
CL_432_LEAVE.indicate
(
        Destination_Address
)
```

Destination_Address is address of the Service Node, which left the subnetwork.

Use

The Base Node EN 61334-4-32 SSCS uses this service to inform the EN 61334-4-32 Connection Manager that the Service Node identified by the Destination_Address has left the subnetwork.

8.2.3 Summary of the connection management services

The MSC of the service primitives exchanged between the EN 61334-4-32 Connection Manager and the EN 61334-4-32 SSCS is shown in Figure 4. Error cases are shown in Annex B.

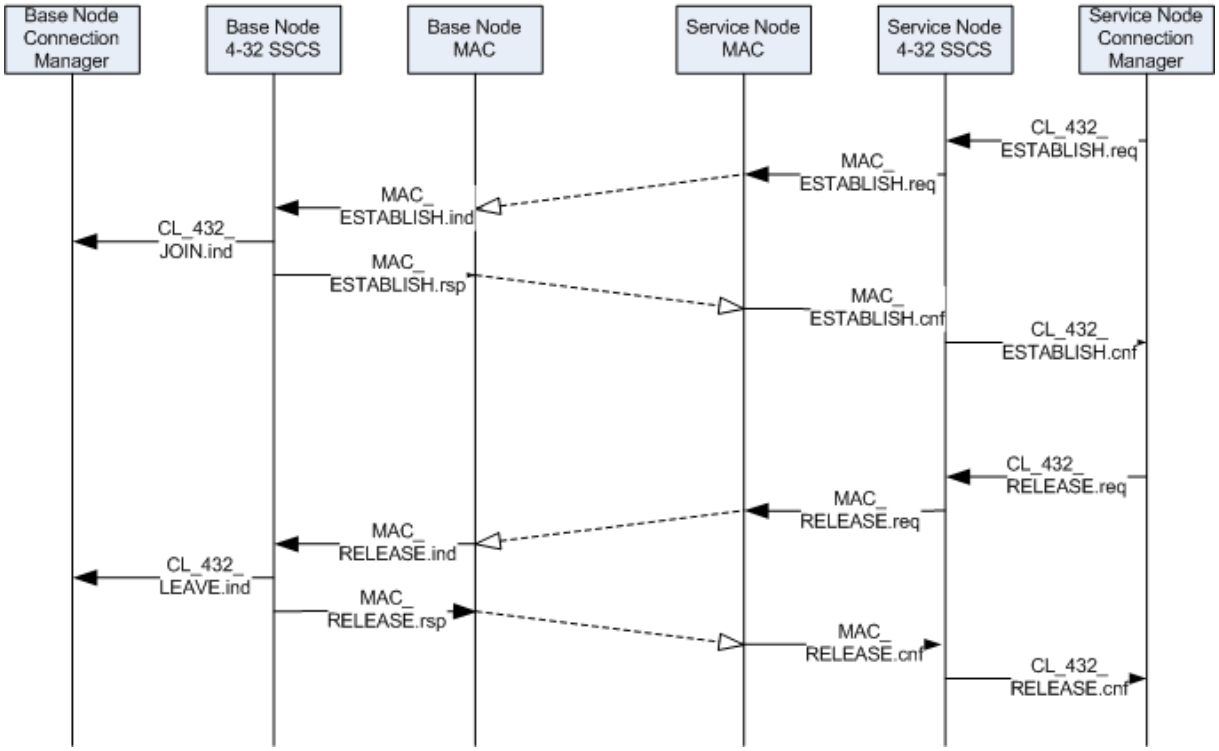


Figure 4 – MSC for EN 61334-4-32 SSCS services

8.2.4 CL_432_DATA services

8.2.4.1 Overview

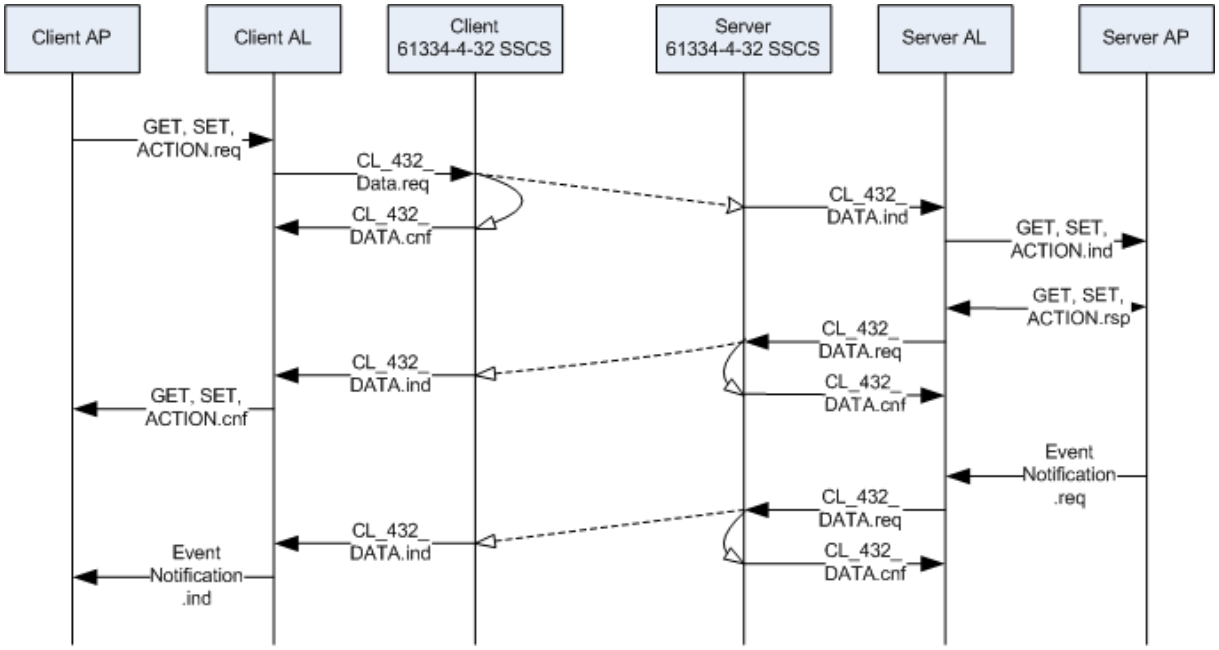
EN 61334-4-32:1996 specifies the following LLC layer data services:

- DL_Data services;
- DL_Broadcast services;
- DL_Reply services; and
- DL_Update_Reply services.

In the DLMS/COSEM PRIME EN 61334-4-32 profile, only the DL_Data services are used, mapped to CL_432_DATA services. All other services are also mapped into CL_432_DATA services as defined below.

8.2.4.2 CL_432_DATA services

For DL_Data data services, please refer to EN 61334-4-32:1996. In this Technical Specification, DL_Data services are mapped into CL_432_DATA services.



NOTE xDLMS LN services are shown as an example here. PRIME does not impose any limitation on using xDLMS services.

Figure 5 – MSC for Data services in the case of logical name referencing

8.2.4.3 DL_Broadcast services

The DL_Broadcast.request service specified in EN 61334-4-32 is not used in this profile. Instead, the CL_432_DATA.request primitive is used with a broadcast address as destination address.

8.2.4.4 DL_Reply and DL_Update_Reply services

In EN 61334-4-32:1996, the DL_Reply and Update_Reply services are used for unsolicited service management. In the DLMS/COSEM PRIME EN 61334-4-32 profile, these services are replaced by CL_432_DATA services at the SSCS level.

8.2.5 Addressing

8.2.5.1 Overview

Three levels of addresses are defined in DLMS/COSEM PRIME EN 61334-4-32 profile:

- at the MAC sublayer level, MAC addresses are processed for accessing the CL;
- at the EN 61334-4-32 SSCS level, Service Node destination addresses and Base Node addresses for accessing the LLC sublayer;
- at the LLC sublayer, source and destination addresses hold the DLMS/COSEM Application layer Client SAP and Server SAP and vice versa depending on the direction of the communication flow.

8.2.5.2 MAC address

The MAC addresses are used to identify non-ambiguously the nodes in a subnetwork participating in the data exchange.

During data transmission, this MAC address is provided by the upper layer to the MAC sublayer and it is consumed by the MAC layer after inserting it in the PPDU.

During reception, the MAC sublayer checks if the packet is for this node and if so, it consumes the address and takes into account the concerned packet.

The full MAC addressing structure is the following:

MSB LSB

48 bits	8 bits	14 bits	9 bits
SNA	LSID	LNID	LCID

Where:

- SNA is the subnetwork address, identified by the EUI-48 of the Base Node;
- LSID is the local switch identifier inside this sub network;
- LNID is the local node identifier of the related Service Node; and
- LCID is the connection identifier.

8.2.5.3 EN 61334-4-32 SSCS addresses

During connection establishment, the Base Node provides to the Service Node its address at the SSCS level. This address identifies the SSCS access point.

NOTE Due to the fact that the EN 61334-4-32 is embedded in the EN 61334-4-32 SSCS, these addresses are not part of the frames exchanged; therefore they are not visible outside the device.

8.2.5.4 LLC addresses

The LLC addresses identify the client and the server SAPs at the application layer level. For the client, this address is the client Id. For the server, the SAP identifies the logical device concerned. These addresses are of one byte each.

NOTE Service Nodes act as DLMS/COSEM servers. The Base Node acts as a DLMS/COSEM client or as a gateway for a client external to the PRIME subnetwork.

Table 2 – Client service access point values

Value	Purpose
0	No station
1	Management client
2… 0x0F	
0x10	Public client
0x11… 0xFF	

Table 3 – Server service access point values

Value	Purpose
0	No station
1	Management logical device
0x02… 0x0F	Reserved for future use
0x10… 0x0FE	
0xFF	All station (Broadcast)

8.3 The TCP-UDP/IPv4 based convergence sublayer

8.3.1 Overview

8.3.1.1 General architecture

The TCP-UDP/IPv4 based communication profile is fully in line with the DLMS/COSEM communication profile for TCP-UDP/IP, as specified in EN 62056-9-7:2013. Please refer to that standard for more information. The present clause provides information related to the binding of TCP-UDP/IPv4 layers with the IPv4 SSCS only.

The IPv4 SSCS provides an efficient method for transporting IPv4 packets over the PRIME network.

Although IPv4 is a connectionless protocol, the IPv4 SSCS is connection-oriented meaning that a connection is established between the source and destination Service Nodes for the transfer of IP packets. This connection is maintained while traffic is being transferred and may be removed after a period of inactivity. The SSCS has two connection types:

- for address resolution: there is an address resolution connection to the Base Node see 8.3.2;
- for IPv4 data transfer there is a connection per destination node. The destination may be the Base Node, another Service Node on the same subnetwork, or a destination outside of the subnetwork.

Optionally TCP/IPv4 headers may be compressed. The compression is negotiated as part of the connection establishment phase.

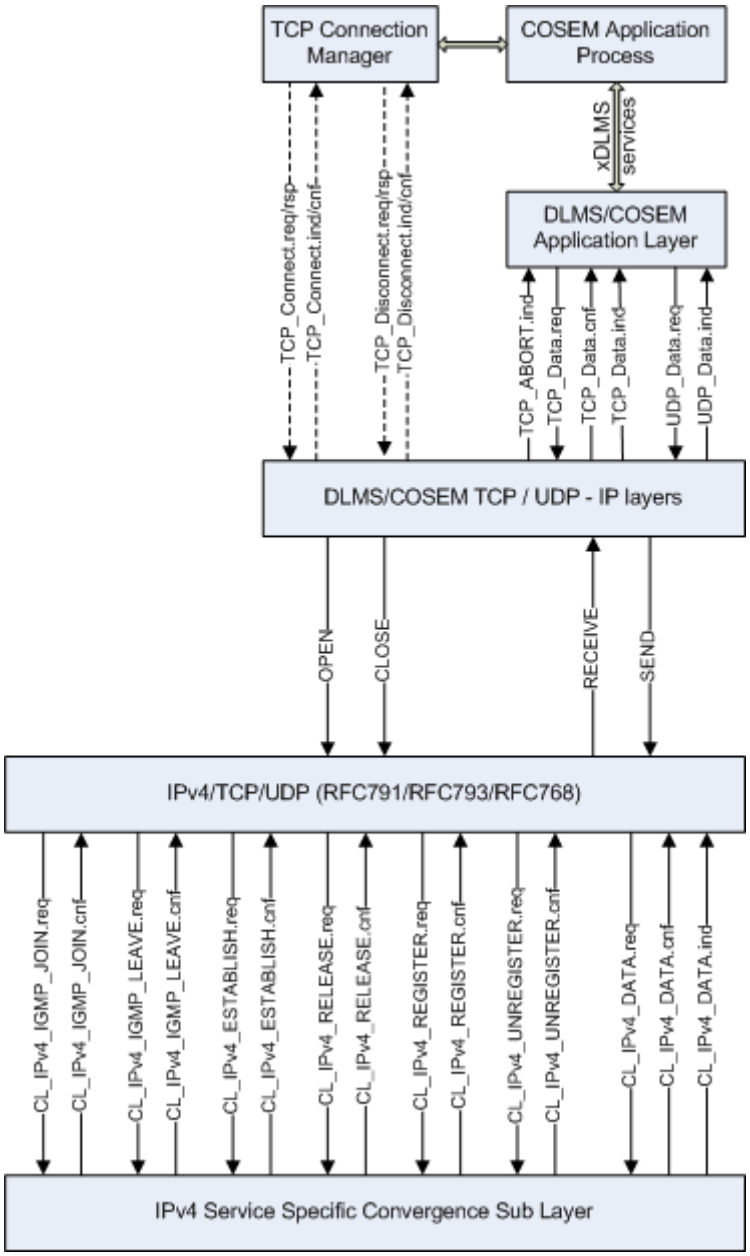
Broadcasting of IPv4 packets is supported using the MAC broadcast mechanism.

Multicasting of IPv4 packets is supported using the MAC multicast mechanism.

Segmentation and reassembly services are provided by the CPCS.

For more information see ITU-T G.9904:2012.

The general architecture is the following:



NOTE Details related to the node which issues the primitives are provided in the service description. See 8.3.2 and 8.3.6.3.

Figure 6 – The TCP-UDP/IPv4 communication profile architecture

On this figure, only upper layers are shown. Lower layers and their relationship are described in ITU-T G.9904:2012.

8.3.1.2 The TCP connection manager

The TCP Connection Manager is specified in EN 62056-9-7:2013.

8.3.1.3 TCP-UDP/IPv4

The IPv4, TCP, and UDP layers implement the standard IPv4, TCP and UDP services respectively as specified in RFC0791, RFC0793 and RFC0768. Specifying how the IPv4 layer is bound to the IPv4 SSCS is the purpose of the following clauses.

8.3.1.4 Subnetwork gateway

The Base Node usually acts as a gateway to the PRIME subnetwork providing IP connectivity to the Service Nodes. Its main features are the following:

- It can exchange IPv4 packets with Service Nodes, and it can also forward packets from one Service Node to another Service Node;
- It may perform NAT functions;
- It may perform DHCP functions;

  NOTE IPv4 addresses can be statically preconfigured or dynamic. When the IPv4 address is dynamically configured, then a DHCP server is used. When using DHCP, it is performed using the broadcast/multicast services. Then the IPv4 address is registered using CL_IPv4_REGISTER service. It is not the scope of the present Technical Specification to describe the DHCP or NAT process.

- It performs IPv4 to EUI-48 address resolution, maintaining a database of the EUI-48 address and IPv4 addresses of each Service Node registered to it. Service nodes – provided that they are registered with the Base Node – can then query the Base Node to resolve an IPv4 address to an EUI-48 address;
- It checks in each packet the IPv4 address to determine if the packet should be processed by a node in the subnetwork or should be sent to a destination outside the subnetwork.

In order to keep the implementation simple, only one single route is supported per local IPv4 address.

8.3.2 Opening and closing the IPv4 SSCS

8.3.2.1 Introduction

The following services are used to open and close the IPv4 SSCS. At a given moment, only one SSCS connection can exist, therefore the SSCS shall be opened once only. The IPv4 layer may close the SSCS when the IPv4 interface is brought down. The SSCS will also close when the underlying MAC connection to the Base Node is lost.

For MSC of IPv4 SSCS services please refer to Annex A.

8.3.2.2 CL_IPv4_ESTABLISH.request

NOTE For details, see ITU-T G.9904:2012 9.4.8.2.2.

Function

This service primitive is issued by the Service Node IPv4 layer to the IPv4 SSCS. This is done whenever the IPv4 layer brings the interface up.

Service parameters

The semantics of this primitive are as follows:

CL_IPv4_ESTABLISH.request()

Use

On receiving this service primitive, the Service Node IPv4 SSCS will form the address resolution connection to the Base Node and join the broadcast group used for receiving/transmitting broadcast packets. See Figure A. 1.

8.3.2.3 CL_IPv4_ESTABLISH.confirm

NOTE For details, see ITU-T G.9904:2012 9.4.8.2.3.

Function

This service primitive is passed from the Service Node IPv4 SSCS to the IPv4 layer. In case of success, it signals that the IPv4 SSCS is ready to accept IPv4 packets to be sent to peers.

Service parameters

The semantics of this primitive are as follows:

CL_IPv4_ESTABLISH.confirm()

Use

This service primitive is used to provide the result of the CL_IPv4_ESTABLISH.request service. Once the IPv4 SSCS has established all the necessary connections and is ready to transmit and receive IPv4 packets, this primitive is passed to the IPv4 layer.

If the IPv4 SSCS encounters an error while opening, it responds with a CL_IPv4_RELEASE.confirm primitive, instead of a CL_IPv4_ESTABLISH.confirm primitive.

8.3.2.4 CL_IPv4_RELEASE.request

NOTE For details, see ITU-T G.9904:2012 9.4.8.2.4.

Function

This service primitive is used by the Service Node IPv4 layer to the IPv4 SSCS to request closing the IPv4 SSCS.

Service parameters

The semantics of this primitive are as follows:

CL_IPv4_RELEASE.request()

Use

This primitive is used by the Service Node when the SSCS has to be closed for any reason. The IPv4 SSCS closes all connections. All resources are released and no more IPv4 packets can be received or sent. When the IPv4 SSCS has released all its connections and resources, it returns a CL_IPv4_RELEASE.confirm service primitive.

8.3.2.5 CL_IPv4_RELEASE.confirm

NOTE For details, see ITU-T G.9904:2012 9.4.8.2.5.

Function

This service primitive is used by the Service Node IPv4 SSCS to indicate to the IPv4 layer that the IPv4 SSCS has been closed. This can be the result of a CL_IPv4_RELEASE.request primitive, a failed CL_IPv4_ESTABLISH.request primitive, or because the MAC layer indicates that the address resolution connection has been lost (i.e., this may happen because of an occurrence of a communication error, such as corrupted messages, too many ARQ retransmissions...), or the Service Node itself is no longer registered, meaning no longer IP connectivity.

Service parameters

The semantics of this primitive are as follows:

CL_IPv4_RELEASE.confirm(Result)

The possible values of Result parameter are specified in Table 1.

Use

The IPv4 SSCS uses this service to inform the IPv4 layer that the Service Node has left the subnetwork at the IP level.

8.3.3 Unicast address management

8.3.3.1 Introduction

The services defined here are used for unicast address management, i.e. the registration and unregistration of unicast IPv4 addresses with the IPv4 SSCS.

When there are no unicast IPv4 addresses registered with the IPv4 SSCS, the IPv4 SSCS can only send and receive broadcast and multicast packets. However, this is sufficient for BOOTP/DHCP operation to allow the device to obtain an IPv4 address. Once an IPv4 address has been registered, the IPv4 layer can transmit unicast packets that have a source address equal to one of its registered IPv4 addresses.

8.3.3.2 CL_IPv4_REGISTER.request

NOTE 1 For details, see ITU-T G.9904:2012 9.4.8.3.2.

Function

This primitive is passed from the Service Node IPv4 layer to the IPv4 SSCS to request registering an IPv4 address.

Service parameters

The semantics of this primitive are as follows:

```
CL_IPv4_REGISTER.req(
                IPv4_Address,
                Netmask,
                Gateway)
```

Where:

- The IPv4_Address is the address to be registered.

NOTE 2 A Service Node can register several unicast IPv4 addresses.

- The Netmask is the network mask. It is used by the IPv4 SSCS to determine whether the packet should be delivered directly or the gateway should be used.
- The Gateway is an IPv4 address of the gateway to be used when the destination address is outside the subnetwork.

See also 8.3.1.4.

NOTE 3 As each PRIME subnetwork may have only one Base Node, the Gateway address can generally be the IPv4 address of the Base Node, but this is not mandatory.

NOTE 4 The IPv4 Address, the Netmask and the Gateway address are provided by the IP layer and are configured inside the device by using an "IPv4 setup" object. See EN 62056-6-2[GK1].

Use

From the Service Node side this registration is remote up to the Base Node.

If the Service Node has successfully registered its IPv4 address with the Base Node, a CL_IPv4_REGISTER.cnf primitive is used. If the registration fails, the CL_IPv4_RELEASE.cnf primitive will be used.

From the Base Node side, the registration is only local and used for notifying to the IPv4 SSCS which packets are to be passed to the upper layer.

8.3.3.3 CL_IPv4_REGISTER.confirm

NOTE For details, see ITU-T G.9904:2012 9.4.8.3.3.

Function

This service primitive is passed from the Service Node IPv4 SSCS to the IPv4 layer to indicate that the IPv4 address registration succeeded.

Service parameters

The semantics of this primitive are as follows:

CL_IPv4_REGISTER.cnf(IPv4_Address)

The IPv4_Address is the address that has been successfully registered.

Use

The CL_IPv4_REGISTER.cnf service primitive is used by the Service Node IPv4 SSCS once the registration has been completed. The IPv4 layer can then make use of this address for sending IPv4 packets.

8.3.3.4 CL_IPv4_UNREGISTER.req

NOTE For details, see ITU-T G.9904:2012 9.4.8.3.4.

Function

This service primitive is passed from the Service Node IPv4 layer to the IPv4 SSCS to request unregistering an IPv4 address.

Service parameters

The semantics of this primitive are as follows:

CL_IPv4_UNREGISTER.req(IPv4_Address)

The IPv4_Address is the address to be unregistered.

Use

Once the IPv4 address has been unregistered with the Base Node, a CL_IPv4_UNREGISTER.confirm primitive is used. In the case the CL_IPv4_UNREGISTER.request fails, a CL_IPv4_RELEASE.confirm primitive is sent back. The related reason of the failure is notified in the result.

Because several IPv4 addresses can be registered by the Service Node, therefore only the one concerned is unregistered by this request. The addresses not unregistered are still valid, until their unregistration or until the SSCS is closed.

8.3.3.5 CL_IPv4_UNREGISTER.confirm

NOTE For details, see ITU-T G.9904:2012 9.4.8.3.5.

Function

This service primitive is passed from the Service Node IPv4 SSCS to the IPv4 layer to indicate that unregistering of the IPv4 address has been successful.

Service parameters

The semantics of this primitive are as follows:

CL_IPv4_UNREGISTER.confirm(IPv4_Address)

The IPv4_Address is the unicast IPv4 address that has been unregistered.

Use

Once the unregistration has been completed, the IPv4 layer cannot send IPv4 packets using this source address.

8.3.4 Multicast group management

8.3.4.1 General

This clause describes the service primitives used to manage multicast groups.

NOTE Notice the different nature of CL_432_JOIN primitives and the CL_IPv4_IGMP_JOIN primitives. The first ones inform the Base Node of the new Service Nodes that have joined the network. The second group will manage the subscription to a certain IP multicast group. Also note that this set of primitives have the IGMP in its name so the distinction is clear.

8.3.4.2 CL_IPv4_IGMP_JOIN.req

NOTE For details, see ITU-T G.9904:2012 9.4.8.4.2.

Function

This service primitive is passed from the Service Node or Base Node IPv4 layer to the IPv4 SSCS. It contains an IPv4 multicast group address that is to be joined. The Base Node and the Service Node can use then multicast services.

Service parameters

The semantics of this primitive are as follows:

CL_IPv4_IGMP_JOIN.req(IPv4_Address)

The IPv4_Address is the IPv4 multicast group to be joined.

Use

When the IPv4 SSCS receives this primitive, it will arrange for IPv4 packets sent to this group to be multicast in the subnetwork and to receive packets using this address to be passed to the IPv4 layer. If the SSCS can join the group, it shall use CL_IPv4_IGMP_JOIN.confirm primitive to indicate success. Otherwise, it shall use the CL_IPv4_IGMP_LEAVE.confirm service primitive.

8.3.4.3 CL_IPv4_IGMP_JOIN.confirm

NOTE For details, see ITU-T G.9904:2012 9.4.8.4.3.

Function

This service primitive is passed from the Service Node or the Base Node IPv4 SSCS to the IPv4 layer.

Service parameters

The semantics of this primitive are as follows:

CL_IPv4_IGMP_JOIN.confirm(IPv4_Address)

The IPv4_Address is the IPv4 multicast group that has been joined.

Use

The SSCS will start forwarding IPv4 multicast packets for the given multicast group.

8.3.4.4 CL_IPv4_IGMP_LEAVE.request

NOTE For details, see ITU-T G.9904:2012 9.4.8.4.4.

Function

This service primitive is passed from the Service Node or Base Node IPv4 layer to the IPv4 SSCS. It contains an IPv4 multicast address to be left.

Service parameters

The semantics of this primitive are as follows:

CL_IPv4_IGMP_LEAVE.req (IPv4_Address)

The IPv4_Address is the IPv4 multicast group to be left.

Use

The SSCS will stop forwarding IPv4 multicast packets for this group and may leave the MAC multicast group.

8.3.4.5 CL_IPv4_IGMP_LEAVE.confirm

NOTE For details, see ITU-T G.9904:2012 9.4.8.4.5.

Function

This service primitive is passed from the Service Node or the Base Node IPv4 SSCS to the IPv4 layer to indicate the result of the CL_IPv4_IGMP_LEAVE.request.

Service parameters

The semantics of this primitive are as follows:

CL_IPv4_IGMP_LEAVE.confirm(IPv4_address, Result)

The IPv4_Address is the address of IPv4 multicast group that has been left. The IPv4 SSCS will stop forwarding IPv4 multicast packets for the given multicast group.

For the Result values, see Table 1.

Use

This primitive can be used by the Service Node IPv4 SSCS as a result CL_IPv4_IGMP_LEAVE.request, a CL_IPv4_IGMP_JOIN.request in the case of failure or due to an error condition resulting in the loss of the MAC multicast connection.

8.3.5 Data transfer

8.3.5.1 General

The following primitives are used for sending and receiving IPv4 packets.

8.3.5.2 CL_IPv4_DATA.request

NOTE For details, see ITU-T G.9904:2012 9.4.8.5.2.

Function

This primitive is passed from the IPv4 layer to the IPv4 SSCS. It contains one IPv4 packet to be sent.

Service parameters

The semantics of this primitive are as follows:

CL_IPv4_DATA.request(IPv4_PDU)

The IPv4_PDU is the IPv4 packet to be sent.

Use

This primitive is used when an IPv4 packet has to be transmitted to the peer IPv4 layer.

8.3.5.3 CL_IPv4_DATA.confirm

NOTE For details, see ITU-T G.9904:2012 9.4.8.5.3.

Function

This primitive is passed from the IPv4 SSCS to the IPv4 layer. It contains a status indication and an IPv4 packet that has just been sent.

Service parameters

The semantics of this primitive are as follows:

CL_IPv4_DATA.confirm(IPv4_PDU, Result)

The IPv4_PDU is the IPv4 packet that was to be sent.

The Result value indicates whether the packet was sent or an error occurred. It takes a value from Table 1.

Use

This primitive is generated locally for signalling to the requestor, the result of the data transfer request.

8.3.5.4 CL_IPv4_DATA.indicate

NOTE For details, see ITU-T G.9904:2012 9.4.8.5.4.

Function

This primitive is passed from the IPv4 SSCS to the IPv4 layer. It contains an IPv4 packet that has just been received.

Service parameters

The semantics of this primitive are as follows:

CL_IPv4_DATA.indicate(IPv4_PDU)

The IPv4_PDU is the IPv4 packet that was received.

Use

Upon the arrival of a new IPv4 packet from the peer, the primitive is generated.

8.3.6 IPv4 SSCS PDUs

8.3.6.1 General

The following subclauses specify the IPv4 SSCS PDUs. The services which carry them are MAC sub layer services as shown on Figure 6 and Figure A. 1.

8.3.6.2 Address resolution PDUs

8.3.6.2.1 Overview

The following messages are used over the address resolution connection between the Service Node and the Base Node. See ITU-T G.9904:2012 clause 9.4.7.

AR.MSG values from 6 to 7 are reserved.

8.3.6.2.2 AR_REGISTER_S

The table below shows the address resolution register message built by the Service Node SSCS and sent from the Service Node to the Base Node.

Table 4 – AR_REGISTER_S message format

Name	Length	Description
AR.MSG	8-bits	Address Resolution Message Type For AR_REGISTER_S = 0
AR.IPv4	32-bits	IPv4 address to be registered
AR.EUI-48	48-bits	EUI-48 to be registered

8.3.6.2.3 AR_REGISTER_B

The table below shows the address resolution register acknowledgement message built by the Base Node SSCS and sent from the Base Node to the Service Node.

Table 5 – AR_REGISTER B message format

Name	Length	Description
AR.MSG	8-bits	Address Resolution Message Type For AR_REGISTER_B = 1
AR.IPv4	32-bits	IPv4 address registered
AR.EUI-48	48-bits	EUI-48 registered

8.3.6.2.4 AR_UNREGISTER_S

The table below shows the address resolution unregister message sent from the Service Node to the Base Node.

Table 6 – AR_UNREGISTER_S message format

Name	Length	Description
AR.MSG	8-bits	Address Resolution Message Type For AR_UNREGISTER_S = 2
AR.IPv4	32-bits	IPv4 address to be unregistered
AR.EUI-48	48-bits	EUI-48 of the device which requested the IPv4 to be unregistered

8.3.6.2.5 AR_UNREGISTER_B

The table below shows the address resolution unregister acknowledgement message sent from the Base Node to the Service Node.

Table 7 – AR_UNREGISTER_B message format

Name	Length	Description
AR.MSG	8-bits	Address Resolution Message Type For AR_UNREGISTER_B = 3
AR.IPv4	32-bits	IPv4 address unregistered
AR.EUI-48	48-bits	EUI-48 of the device which IPv4 address is unregistered

8.3.6.2.6 AR_MCAST_REG_S

The table below shows the multicast address resolution register message sent from the Service Node to the Base Node

Table 8 – AR_MCAST_REG_S message format

Name	Length	Description
AR.MSG	8-bits	Address Resolution Message Type For AR_MCAST_REG_S = 8
AR.IPv4	32-bits	IPv4 multicast address to be registered

8.3.6.2.7 AR_MCAST_REG_B

The table below shows the multicast address resolution register acknowledgment message sent from the Base Node to the Service Node.

Table 9 – AR_MCAST_REG_B message format

Name	Length	Description
AR.MSG	8-bits	Address Resolution Message Type For AR_MCAST_REG_B = 9
AR.IPv4	32-bits	IPv4 multicast address registered
Reserved	2-bits	Reserved. Shall be encoded as 0
AR.LCID	6-bits	LCID assigned to this IPv4 multicast address

The AR.IPv4 field is included in the AR_MCAST_REG_B message so that the Service Node can perform multiple overlapping registrations.

8.3.6.2.8 AR_MCAST_UNREG_S

The table below shows the multicast address resolution unregister message sent from the Service Node to the Base Node.

Table 10 – AR_MCAST_UNREG_S message format

Name	Length	Description
AR.MSG	8-bits	Address Resolution Message Type For AR_MCAST_UNREG_S = 10
AR.IPv4	32-bits	IPv4 multicast address to be unregistered

8.3.6.2.9 AR_MCAST_UNREG_B

The table below shows the multicast address resolution unregister acknowledgement message sent from the Base Node to the Service Node

Table 11 – AR_MCAST_UNREG_B message format

Name	Length	Description
AR.MSG	8-bits	Address Resolution Message Type For AR_MCAST_UNREG_B = 11
AR.IPv4	32-bits	IPv4 multicast address unregistered

The AR.IPv4 field is included in the AR_MCAST_UNREG_B message so that the Service Node can perform multiple overlapping unregistrations.

8.3.6.3 Data connection establishment

8.3.6.3.1 Overview

The following PDU are sent during the data connection establishment.

8.3.6.3.2 AR_LOOKUP_S

The table below shows the address resolution lookup message sent from the Service Node to the Base Node.

Table 12 – AR_LOOKUP_S message format

Name	Length	Description
AR.MSG	8-bits	Address Resolution Message Type For AR_LOOKUP_S = 4
AR.IPv4	32-bits	IPv4 address to lookup

8.3.6.3.3 AR_LOOKUP_B

The table below shows the address resolution response message sent from the Base Node to the Service Node.

Table 13 – AR_LOOKUP_B message format

Name	Length	Description
AR.MSG	8-bits	Address Resolution Message Type For AR_LOOKUP_B = 5
AR.IPv4	32-bits	IPv4 address looked up
AR.EUI-48	48-bits	EUI-48 for IPv4 address
AR.Status	8-bits	Lookup status, indicating if the address was found or an error occurred. 0 = found, AR.EUI-48 valid. 1 = unknown, AR.EUI-48 undefined

The lookup may fail if the requested address has not been registered. In that case, AR.Status will have a value other than zero and the contents of AR.EUI-48 will be undefined. The lookup is only successful when AR.Status is zero. In that case, the EUI-48 field contains the resolved address.

8.3.7 IPv4 SSCS packet format

8.3.7.1 General

The following PDU formats are used for transferring IPv4 packets between Service Nodes. Two formats are defined. The first format is used without header compression. The second format is used with Van Jacobsen header compression.

8.3.7.2 IPv4 packet format without header compression

When no header compression has been negotiated, the IP packet is simply sent as it is, without any header compression. For compression negotiation, please see clause 8.3.8.2.

Table 14 – IPv4 packet format without header compression negotiated

Name	Length	Description
IPv4PKT	n-octets	The IPv4 Packet

8.3.7.3 IPv4 packet format with Van Jacobsen header compression

The table below shows the IPv4 packet format when Van Jacobsen header compression has been negotiated. See RFC 1144 clause 3.2.

Table 15 – IPv4 packet format with VJ header compression

Name	Length	Description
IPv4.Type	2-bits	Type of compressed packet. IPv4.Type = 0 – TYPE_IP IPv4.Type = 1 – UNCOMPRESSED_TCP IPv4.Type = 2 – COMPRESSED_TCP IPv4.Type = 3 – TYPE_ERROR
IPv4.Seq	6-bits	Packet sequence number
IPv4.PKT	n-octets	The IPv4 Packet (with the header compressed)

The IPv4.Type value TYPE_ERROR is never sent. It is a pseudo packet type used to tell the decompressor that a packet has been lost.

8.3.8 Connection data

8.3.8.1 General

The following PDUs are exchanged during the data connection establishment phase. When a connection is established between Service Nodes for the transfer of IPv4 packets, data is also transferred in the connection request packets. This data allow the negotiation of compression and notification of the IPv4 address. See 8.4.2.1 and ITU-T G.9904:2012 clause 8.6.6.1.

8.3.8.2 Connection data from the initiator

The table below shows the connection data as it has to be sent by the connection initiator.

Table 16 – Connection data sent by the initiator

Name	Length	Description
Reserved	6-bits	Should be encoded as zero in this version of the SSCS protocol
Data.HC	2-bits	Header Compression Data.HC = 0 – No compression requested Data.HC = 1 – VJ Compression requested Data.HC = 2, 3 – Reserved for future versions of this Technical Specification
Data.IPv4	32-bits	IPv4 address of the initiator

If the device accepts the connection, it should copy the Data.IPv4 address into a new table entry along with the negotiated Data.HC value.

8.3.8.3 Connection data from the responder

The table below shows the connection data sent in response to the connection request.

Table 17 – Connection data sent by the responder

Name	Length	Description
Reserved	6-bits	Should be encoded as zero in this version of the SSCS protocol
Data.HC	2-bit	Header Compression negotiated Data.HC = 0 – No compression permitted Data.HC = 1 – VJ Compression negotiated Data.HC = 2,3 – Reserved

A header compression scheme can only be used when it is supported by both Service Nodes. The responder may only set Data.HC to 0 or the same value as that received from the initiator. When the same value is used, it indicates that the requested compression scheme has been negotiated and will be used for the connection. Setting Data.HC to 0 allows the responder to deny the request for that header compression scheme or force the use of no header compression.

8.4 The TCP-UDP/IPv6 based convergence sublayer

8.4.1 Overview

8.4.1.1 General architecture

The general TCP-UDP/IPv6 architecture is the same as the TCP-UDP/IPv4 one, meaning according to the EN 62056-9-7:2013. The lower layers are according to ITU-T G.9904:2012. The present clause provides information related to the binding of TCP-UDP/IPv6 layers with the IPv6 SSCS. See Figure 2.

The IPv6 SSCS provides an efficient method for transferring IPv6 packets over the PRIME network.

A Service Node can pass IPv6 packets to the Base Node or directly to other Service Nodes.

By default, the Base Node acts as a router between the PRIME subnet and the backbone network. A Base Node shall have at least this connectivity capability. Any other node inside the subnetwork can also act as a gateway. The Base Node can also act as a NAT router. However given the abundance of IPv6 addresses this is not expected. How the Base Node connects to the backbone is beyond the scope of this Technical Specification.

8.4.1.2 IPv6 unicast addressing assignment

IPv6 Service Nodes (and Base Nodes) shall support the standard IPv6 protocol, as described in RFC 2460.

IPv6 Service Nodes (and Base Nodes) shall support the standard IPv6 addressing architecture, as described in RFC 4291.

IPv6 Service Nodes (and Base Nodes) shall support global unicast IPv6 addresses, link-local IPv6 addresses and multicast IPv6 addresses, as described in RFC 4291.

IPv6 Service Nodes (and Base Nodes) shall support automatic address configuration using stateless address configuration as specified in RFC 4862. They may also support automatic address configuration using stateful address configuration as specified in RFC 3315 and they may support manual configuration of IPv6 addresses. The decision of which address configuration scheme to use is deployment specific.

Service nodes shall support DHCPv6 client, when Base nodes support DHCPv6 server as specified in RFC 3315 for stateless address configuration.

8.4.1.3 Role of the Base Node

At the IPv6 SSCS level, the Base Node maintains a table containing all the IPv6 unicast addresses and the EUI-48 addresses related to them. One of the roles of the Base Node is to perform IPv6 to EUI-48 address resolution. Each Service Node belonging to the subnetwork managed by the Base Node registers its IPv6 address and EUI-48 address with the Base Node. Other Service Nodes can then query the Base Node to resolve an IPv6 address into a EUI-48 address. This requires the establishment of a dedicated connection to the Base Node for address resolution.

Optionally UDP/IPv6 headers may be compressed. Compression is negotiated as part of the connection establishment phase. Currently one header compression technique is described in the present Technical Specification that is used for transmission of IPv6 packets over IEEE 802.15.4-based networks, as defined in RFC 6282. This is also known as LOWPAN_IPHC.

The multicasting of IPv6 packets is supported using the MAC multicast mechanism.

8.4.2 IPv6 SSCS

8.4.2.1 General

The IPv6 SSCS has two types of connection types. For address resolution, there is a connection to the Base Node. For IPv6 data transfer, there is one connection per destination node. The Base Node may act as the IPv6 gateway to the outside world or another node in the same subnetwork.

8.4.2.2 Routing in the subnetwork

Routing IPv6 packets is the scope of the IPv6 SSCS. In other words, the IPv6 SSCS will decide whether the packet should be sent directly to another Service Node or forwarded to the configured gateway depending on the IPv6 destination address.

Although IPv6 is a connectionless protocol, the IPv6 SSCS is connection-oriented. Once address resolution has been performed, a connection is established between the source and destination Service Nodes for the transfer of IPv6 packets. See 8.4.14. This connection is maintained all the time the traffic is being transferred and may be removed after a period of inactivity.

8.4.2.3 CPCS: Segmentation and reassembly

The CPCS sublayer shall always be present with the IPv6 SSCS allowing segmentation and reassembly facilities. Thus, the MSDUs generated by the IPv6 SSCS are always less than CIMTUSize bytes and application messages are expected to be no longer than CIMaxAppPktSize.

NOTE CIMTUSize is of 256 bytes. CIMaxAppPktSize is of Max(SAR.NSEGS)xCIMTUSize. See ITU-T G.9904:2012 Clause 9.2.2.2.

8.4.3 IPv6 Address Configuration

8.4.3.1 Overview

The Service Nodes may use manual configured IPv6 addresses, link local addresses, stateless auto-configuration according to RFC 4862, or DHCPv6 to obtain IPv6 addresses. All nodes shall support the unicast link local address, in addition to other configured addresses below, and multicast addresses, if ever the node belongs to multicast groups.

8.4.3.2 Interface identifier

In order to make use of stateless address auto configuration and link local addresses it is necessary to define how the Interface identifier, as defined in RFC 4291, is derived. Each PRIME node has a unique EUI-48. This EUI-48 is converted into a EUI-64 in the same way as for Ethernet networks as defined in RFC 2464. This EUI-64 is then used as the Interface Identifier.

8.4.3.3 IPv6 Link local address configuration

The IPv6 Link local address of a PRIME interface is formed by appending the Interface Identifier as defined above to the Prefix FE80::/64.

8.4.3.4 Stateless address auto configuration

An IPv6 address prefix of a PRIME interface used for stateless auto configuration, as defined in RFC 4862, shall have a length of 64 bits. The IPv6 prefix is obtained by the Service Nodes from the Base Node via Router Advertisement messages, which are sent periodically by routers or on request by the Base Node.

8.4.3.5 Stateful address configuration

Alternatively, IPv6 addresses can be configured using DHCPv6, as described in RFC 3315. DHCPv6 can provide a device with addresses assigned by a DHCPv6 server and other configuration information, which are carried in options.

8.4.3.6 Multicast address

IPv6 Service Nodes (and Base Nodes) shall support multicast IPv6 addressing, as described in RFC 4291 clause 2.7.

8.4.3.7 Address resolution

8.4.3.7.1 Overview

The IPv6 layer will present the IPv6 SSCS with an IPv6 packet to be transferred. The IPv6 SSCS is responsible for determining which Service Node the packet should be delivered to, using the IPv6 addresses in the packet. The IPv6 SSCS shall then establish a connection to the destination if one does not already exist so that the packet can be transferred. Two classes of IPv6 addresses can be used and the following section describes how these addresses are resolved into PRIME EUI-48 addresses. It should be noted that IPv6 does not have a broadcast address. However broadcasting is possible using multicast to all nodes addresses.

8.4.3.7.2 Unicast address

8.4.3.7.2.1 General

IPv6 unicast addresses shall be resolved into PRIME unicast EUI-48 addresses. The Base Node maintains a central database of IPv6 addresses and EUI-48 addresses. Address resolution functions are performed by querying this database. The Service Node shall establish a connection to the address resolution service running on the Base Node, using the TYPE value TYPE_CL_IPv6_AR (see ITU-T G.9904:2012 Table C.1). No data should be passed in the connection establishment signalling.

8.4.3.7.2.2 Address registration and deregistration

A Service Node uses the AR_REGISTERv6_S message to register an IPv6 address and the corresponding EUI-48 address. The Base Node will acknowledge an AR_REGISTERv6_B message. The Service Node may register multiple IPv6 addresses for the same EUI-48.

A Service Node uses the AR_UNREGISTERv6_S message to unregister an IPv6 address and the corresponding EUI-48 address. The Base Node will acknowledge with an AR_UNREGISTERv6_B message.

When the address resolution connection between the Service Node and the Base Node is closed, the Base Node should remove all addresses associated with that connection.

8.4.3.7.2.3 Address lookup

A Service Node uses the AR_LOOKUPv6_S message to perform a lookup. The message contains the IPv6 address to be resolved. The Base Node will respond with an AR_LOOKUPv6_B message that contains an error code and, if there is no error, the EUI-48 associated with the IPv6 address. If the Base Node has multiple entries in its database Node for the same IPv6 address, the possible EUI-48 returned is undefined.

It should be noted that, for the link local addresses, due to the fact that the EUI-48 can be obtained from the IPv6 address, the lookup can simply return this value by extracting it from the IPv6 address.

8.4.3.7.3 Multicast address

Multicast IPv6 addresses are mapped to connection handles (ConnHandle) by the IPv6 SSCS.

To join a multicast group, the IPv6 SSCS uses the MAC_JOIN.request primitive with the IPv6 address specified in the data field. A corresponding MAC_JOIN.confirm primitive will be generated by the MAC after completion of the join process. The MAC_Join.confirm primitive will contain the result (success/failure) and the corresponding ConnHandle to be used by the IPv6 SSCS. The MAC layer will handle the transfer of data for this connection using the appropriate LCIDs. To leave the multicast group, the IPv6 SSCS at the Service Node shall use the MAC_LEAVE.request(ConnHandle) primitive.

To send an IPv6 multicast packet, the IPv6 SSCS will simply send the packet to the group, using the allocated ConnHandle. The ConnHandle is maintained while there are more packets to be sent. However, after Tmcast_reg seconds of not sending an IPv6 multicast packet to the group, the node should release the ConnHandle by using the MAC_LEAVE.request primitive. The nominal value of Tmcast_reg is 10 minutes; however, other values may be used.

NOTE Tmcast_reg is the timeout for unregistering from the multicast group. See ITU-T G.9904:2012 Clause 9.4.2.4.

8.4.3.7.4 Retransmission of address resolution packets

The connection between the Service Node and the Base Node for address resolution is not reliable. The MAC ARQ is not used. The Service Node is responsible for making retransmissions if the Base Node does not respond in one second. It is not considered an error when the Base Node receives the same registration requests multiple times or is asked to remove a registration that does not exist. These conditions can be the result of retransmissions.

8.4.4 IPv6 packet transfer

For packets to be transferred a connection needs to be established between the source and destination nodes (see Figure A. 1). The IPv6 SSCS will examine each IP packet to determine the destination EUI-48 address. If a connection to the destination has already been established, the packet is simply sent. For this purpose, the IPv6 SSCS keeps a table for each connection it has with information shown in Table 18. To use this table, it is first necessary to determine if the remote address is in the local subnet or if ever a gateway has to be used. The netmask associated with the local IPv6 address is used to determine this. If the destination address is not in the local subnetwork, the address of the gateway is used instead of the destination address when the table is searched.

Table 18 – IPv6 SSCS table entry

Parameter	Description
CL_IPv6_Con.Remote_IP	Remote IP address of this connection
CL_IPv6_Con.ConHandle	MAC Connection handle for the connection
CL_IPv6_Con.LastUsed	Timestamp of last packet received/transmitted
CL_IPv6_Con.HC	Header Compression scheme being used

The IPv6 SSCS may close a connection when it has not been used for an implementation-defined time period. When the connection is closed the entry for the connection is removed at both ends of the connection.

When a connection to the destination does not exist, more work is necessary. The address resolution service is used to determine the EUI-48 address of the remote IP address if it is local or the gateway associated with the local address if the destination address is in another subnet. When the Base Node replies with the EUI-48 address of the destination Service Node, a MAC connection is established to the remote device. The TYPE value of this established connection is TYPE_CL_IPv6_UNICAST (see ITU-T G.9904:2012 Table C.1). The data passed in the request message is defined in section 8.4.13. The local IPv6 address is provided so that the remote device can add the new connection to its cache of connections for sending data in the opposite direction. The use of header compression is also negotiated as part of the connection establishment. Once the MAC connection has been established, the IPv6 packet can be sent.

8.4.5 Segmentation and reassembly

The IPv6 SSCS shall support IPv6 packets with an MTU of 1 500 bytes. This requires the use of the common part convergence sublayer segmentation and reassembly service.

8.4.6 Compression

It is assumed that any PRIME device is capable of LOWPAN_IPHC IPv6 header compression/decompression. It may also be capable of performing UDP compression/decompression. Thus UDP/IPv6 compression is negotiated.

No negotiation can take place for multicast packet. Nodes can only make use of mandatory compression capabilities.

Depending of the type of IPv6 address carried by the packet and the capabilities which are negotiated between the nodes involved in the data exchanges, IPv6 header compression is performed.

All the Service Nodes and the Base Node shall support IPv6 Header Compression using source and destination Addresses stateless compression as defined in RFC 6282. Source and destination IPv6 addresses using stateful compression and IPv6 Next header compression are negotiable.

8.4.7 Quality of Service Mapping

In the specification of the PRIME MAC layer it is specified that the contention-based access mechanism supports 4 priority levels (1-4). Level 1 is used for MAC signalling messages, but not exclusively so.

IPv6 packets include a traffic class field in the header to indicate the QoS the packet would like to receive. This traffic class can be used in the same way that IPv4 ToS, see ITU-T G.9904:2012 clause 9.4.6. That is, three bits of the ToS indicate the IP Precedence. The following table specifies how the IP Precedence is mapped into the PRIME MAC priority.

Table 19 – Mapping IPv6 precedence to PRIME MAC priority

IP Precedence	MAC Priority
000 – Routine	4
001 – Priority	4
010 – Immediate	3
011 – Flash	3
100 – Flash Override	2
101 – Critical	2
110 – Internetwork Control	1
111 – Network Control	1
NOTE At the MAC layer level the priority as set in the Packet header field is the value assigned in this table minus 1, as the range of PKT.PRIO field is from 0 to 3.	

8.4.8 Opening and closing the IPv6 SSCS

8.4.8.1 Introduction

The following services are used to open and close the IPv6 SSCS. At a given moment, only one IPv6 SSCS connection can exist, therefore the IPv6 SSCS shall be opened once only. The IPv6 layer may close the IPv6 SSCS when the IPv6 interface is brought down. The IPv6 SSCS will also close when the underlying MAC connection to the Base Node is lost.

NOTE For details, see ITU-T G.9904:2012 9.6.9.2.

8.4.8.2 CL_IPv6_ESTABLISH.request

Function

This service primitive is issued by the Service Node IPv6 layer to the IPv6 SSCS. This is done whenever the IPv6 layer brings the interface up.

Service parameters

The semantics of this primitive are as follows:

CL_IPv6_ESTABLISH.request()

Use

On receiving this service primitive, the Service Node IPv6 SSCS will form the address resolution connection to the Base Node.

8.4.8.3 CL_IPv6_ESTABLISH.confirm

Function

This service primitive is passed from the Service Node IPv6 SSCS to the IPv6 layer. In case of success, it signals that the IPv6 SSCS is ready to accept IPv6 packets to be sent to peers.

Service parameters

The semantics of this primitive are as follows:

CL_IPv6_ESTABLISH.confirm()

Use

This service primitive is used to provide the result of the CL_IPv6_ESTABLISH.request service. Once the IPv6 SSCS has established all the connection and is ready to transmit and receive IPv6 packets, this primitive is passed to the IPv6 layer.

If the IPv6 SSCS encounters an error while opening, it responds with a CL_IPv6_RELEASE.confirm primitive, instead of a CL_IPv6_ESTABLISH.confirm primitive.

8.4.8.4 CL_IPv6_RELEASE.request

Function

This service primitive is used by the Service Node IPv6 layer to the IPv6 SSCS to request closing the IPv6 SSCS connection.

Service parameters

The semantics of this primitive are as follows:

CL_IPv6_RELEASE.request()

Use

This primitive is used by the Service Node when the connection has to be closed for any reason. The IPv6 SSCS closes all connections. All resources are released and no more IPv6 packets can be received or sent. When the IPv6 SSCS has released all its connections and resources, it returns a CL_IPv6_RELEASE.confirm service primitive.

8.4.8.5 CL_IPv6_RELEASE.confirm

Function

This service primitive is used by the Service Node IPv6 SSCS to indicate to the IPv6 layer that the IPv6 SSCS has been closed. This can be the result of a CL_IPv6_RELEASE.request primitive, a failed CL_IPv6_ESTABLISH.request primitive, or because the MAC layer indicates that the address resolution connection has been lost, or the Service Node itself is no longer registered.

Service parameters

The semantics of this primitive are as follows:

CL_IPv6_RELEASE.confirm(Result)

The possible values of Result parameter are specified in Table 1.

Use

The IPv6 SSCS uses this service to inform the IPv6 layer that the Service Node has left the subnetwork.

8.4.9 Unicast address management

8.4.9.1 Introduction

The services defined here are used for address management, i.e. the registration and unregistration of unicast IPv6 addresses with the IPv6 SSCS.

NOTE For details, see ITU-T G.9904:2012 9.6.9.3.

8.4.9.2 CL_IPv6_REGISTER.request

Function

This primitive is passed from the Service Node IPv6 layer to the IPv6 SSCS to request registering an IPv6 address.

Service parameters

The semantics of this primitive are as follows:

```
CL_IPv6_REGISTER.req(
                    IPv6_Address,
                    Netmask,
                    Gateway)
```

Where:

- The IPv6_Address is the address to be registered;

NOTE 1 A Service Node can register several unicast IPv6 addresses.

- The Netmask is the network mask. It is used by the IPv6 SSCS to determine whether the packet should be delivered directly or the gateway should be used.

NOTE 2 For IPv6, the IPv6 address prefix stands for Network mask.

- The Gateway is an IPv6 address of the gateway to be used when the destination address is outside the subnetwork.

As each PRIME OFDM subnetwork may have only one Base Node, the Base Node may act as a Gateway between the PRIME subnetwork and the backbone. All the Base Nodes shall have this connectivity capability. Any other node inside the subnetwork can also act as a Gateway.

Use

From the Service Node side this registration is remote up to the Base Node.

If the Service Node has successfully registered its IPv6 address with the Base Node, a CL_IPv6_REGISTER.cnf primitive is used. If the registration fails, the CL_IPv6_RELEASE.cnf primitive will be used.

8.4.9.3 CL_IPv6_REGISTER.confirm

Function

This service primitive is passed from the Service Node IPv6 SSCS to the IPv6 layer to indicate that the IPv6 address registration succeeded.

Service parameters

The semantics of this primitive are as follows:

```
CL_IPv6_REGISTER.cnf(IPv6_Address)
```

The IPv6_Address is the address that has been successfully registered.

Use

The CL_IPv6_REGISTER.cnf service primitive is used by the Service Node IPv6 SSCS once the registration has been completed. The IPv6 layer can then make the use of this address as a source address for sending IPv6 packets.

8.4.9.4 CL_IPv6_UNREGISTER.request

Function

This service primitive is passed from the Service Node IPv6 layer to the IPv6 SSCS to request unregistering an IPv6 address.

Service parameters

The semantics of this primitive are as follows:

CL_IPv6_UNREGISTER.req(IPv6_Address)

The IPv6_Address is the address to be unregistered.

Use

Once the IPv6 address has been unregistered with the Base Node, a CL_IPv6_UNREGISTER.confirm primitive is used. In the case the CL_IPv6_UNREGISTER.request fails, a CL_IPv6_RELEASE.confirm primitive is sent back. The related reason of the failure is notified in the result.

Because several IPv6 addresses can be registered by the Service Node, therefore only the one concerned is unregistered by this request. The addresses not unregistered are still valid, until their unregistration or a release primitive takes place.

8.4.9.5 CL_IPv6_UNREGISTER.confirm

Function

This service primitive is passed from the Service Node IPv6 SSCS to the IPv6 layer to indicate that unregistering of the IPv6 address has been successful.

Service parameters

The semantics of this primitive are as follows:

CL_IPv6_UNREGISTER.confirm(IPv6_Address)

The IPv6_Address is the unicast IPv6 address that has been unregistered.

Use

Once unregistration has been completed, the IPv6 layer will not send IPv6 packets using this source address.

8.4.10 Multicast group management

8.4.10.1 Introduction

This section describes the primitives used to manage multicast groups.

NOTE 1 For details, see ITU-T G.9904:2012 9.6.9.4.

NOTE 2 Note the different nature of CL_432_JOIN primitives and the CL_IPv6_MUL_JOIN primitives. The first ones inform the Base Node of the new Service Nodes that have joined the network. The second group will manage the subscription to a certain IP multicast group. Also note that this set of primitives has the MUL in its name so the distinction is clear.

8.4.10.2 CL_IPv6_MUL_JOIN.request

Function

This primitive is passed from the IPv6 layer to the IPv6 SSCS. It contains an IPv6 multicast address that is to be joined.

Service parameters

The semantics of this primitive are as follows:

CL_IPv6_MUL_JOIN.request(IPv6_Address)

The IPv6_Address is the IPv6 multicast group address that is to be joined.

Use

This primitive is used when the IPv6 layer want to subscribe to a multicast group.

8.4.10.3 CL_IPv6_MUL_JOIN.confirm

Function

This primitive is passed from the IPv6 SSCS to the IPv6 layer. It contains a result status and an IPv6 multicast address that was joined.

Service parameters

The semantics of this primitive are as follows:

CL_IPv6_MUL_JOIN.confirm(IPv6_Address)

The IPv6_Address is the IPv6 multicast group address that was joined.

Use

The IPv6 SSCS will start forwarding IPv6 multicast packets for the given multicast group.

When the IPv6 SSCS receives this primitive, it will arrange for IP packets sent to this group to be multicast in the PRIME network and receive packets using this address to be passed to the IPv6 stack. If the IPv6 SSCS cannot join the group, it uses the CL_IPv6_MUL_LEAVE.confirm primitive. Otherwise the CL_IPv6_MUL_JOIN.confirm primitive is used to indicate success.

8.4.10.4 CL_IPv6_MUL_LEAVE.request

Function

This primitive is passed from the IPv6 layer to the IPv6 SSCS. It contains an IPv6 multicast address to be left.

Service parameters

The semantics of this primitive are as follows:

CL_IPv6_MUL_LEAVE.request(IPv6_Address)

The IPv6_address is the IPv6 multicast group address to be left.

Use

The IPv6 SSCS will stop forwarding IPv6 multicast packets for this group and may leave the PRIME MAC multicast group.

8.4.10.5 CL_IPv6_MUL_LEAVE.confirm

Function

This primitive is passed from the IPv6 SSCS to the IPv6 layer. It contains a result status and an IPv6 multicast address that was left.

Service parameters

The semantics of this primitive are as follows:

CL_IPv6_MUL_LEAVE.confirm(IPv6_Address, Result)

The IPv6_address is the IPv6 multicast group that was left.

Use

The IPv6 SSCS will stop forwarding IPv6 multicast packets for the given multicast group.

The Result takes a value from Table 1.

This primitive can be used by the IPv6 SSCS as a result of a CL_IPv6_MUL_JOIN.request, CL_IPv6_MUL_LEAVE.request or because of an error condition resulting in the loss of the PRIME MAC multicast connection.

8.4.11 Data transfer

8.4.11.1 General

The following primitives are used to send and receive IPv6 packets.

NOTE For details, see ITU-T G.9904:2012 9.6.9.5.

8.4.11.2 CL_IPv6_DATA.request

Function

This primitive is passed from the IPv6 layer to the IPv6 SSCS. It contains one IPv6 packet to be sent.

Service parameters

The semantics of this primitive are as follows:

CL_IPv6_DATA.request(IPv6_PDU)

The IPv6_PDU is the IPv6 packet to be sent.

Use

This primitive is used when an IPv6 packet has to be transmitted to the peer application.

8.4.11.3 CL_IPv6_DATA.confirm

Function

This primitive is passed from the IPv6 SSCS. It contains a status indication and an IPv6 packet that has just been sent.

Service parameters

The semantics of this primitive are as follows:

CL_IPv6_DATA.confirm(IPv6_PDU, Result)

The IPv6_PDU is the IPv6 packet that was to be sent.

The Result value indicates whether the packet was sent or an error occurred. It takes a value from Table 1.

Use

This primitive is generated for signalling to the requestor the result of the data transfer request.

8.4.11.4 CL_IPv6_DATA.indicate

Function

This primitive is passed from the IPv6 SSCS to the IPv6 layer. It contains an IPv6 packet that has just been received.

Service parameters

The semantics of this primitive are as follows:

CL_IPv6_DATA.indicate(IPv6_PDU)

The IPv6_PDU is the IPv6 packet that was received.

Use

Upon the arrival of a new IPv6 packet from the peer, the primitive is generated.

8.4.12 IPv6 SSCS PDUs

8.4.12.1 General

This section defines the IPv6 SSCS PDUs. For the sequence, refer to Figure A. 1, and ITU-T G.9904:2012 clause 9.6.8.

8.4.12.2 Address resolution PDUs

8.4.12.2.1 Overview

The following PDUs are transferred over the address resolution connection between the Service Node and the Base Node for the purpose of IP address management upon the CL_IPv6_REGISTER.request. The following sections define a number of AR.MSG values.

NOTE AR.MSG values 22 and 23 are reserved.

8.4.12.2.2 AR_REGISTERv6_S

The table below shows the address resolution register message sent from the Service Node to the Base Node.

Table 20 – AR_REGISTERv6_S message format

Name	Length	Description
AR.MSG	8-bits	Address Resolution Message Type For AR_REGISTERv6_S = 16
AR.IPv6	128-bits	IPv6 address to be registered
AR.EUI-48	48-bits	EUI-48 to be registered

8.4.12.2.3 AR_REGISTERv6_B

The table below shows the address resolution register acknowledgment message sent from the Base Node to the Service Node, as the result of the CL_IPv6_REGISTER.confirm.

Table 21 – AR_REGISTERv6_B message format

Name	Length	Description
AR.MSG	8-bits	Address Resolution Message Type For AR_REGISTERv6_B = 17
AR.IPv6	128-bits	IPv6 address registered
AR.EUI-48	48-bits	EUI-48 registered

The AR.IPv6 and AR.EUI-48 fields are included in the AR_REGISTERv6_B message so that the Service Node can perform multiple overlapping registrations of IPv6 addresses.

8.4.12.2.4 AR_UNREGISTERv6_S

The table below shows the address resolution unregister message sent from the Service Node to the Base Node.

Table 22 – AR_UNREGISTERv6_S message format

Name	Length	Description
AR.MSG	8-bits	Address Resolution Message Type For AR_UNREGISTERv6_S = 18
AR.IPv6	128-bits	IPv6 address to be unregistered
AR.EUI-48	48-bits	EUI-48 of the device which requested the IPv6 to be unregistered

8.4.12.2.5 AR_UNREGISTERv6_B

The table below shows the address resolution unregister acknowledgment message sent from the Base Node to the Service Node.

Table 23 – AR_UNREGISTERv6_B message format

Name	Length	Description
AR.MSG	8-bits	Address Resolution Message Type For AR_UNREGISTERv6_B = 19
AR.IPv6	128-bits	IPv6 address unregistered
AR.EUI-48	48-bits	EUI-48 which IPv6 address has been unregistered

The AR.IPv6 and AR.EUI-48 fields are included in the AR_UNREGISTERv6_B message so that the Service Node can perform multiple overlapping unregistrations.

8.4.12.2.6 AR_MCAST_REGv6_S

The table below shows the multicast address resolution register message sent from the Service Node to the Base Node.

Table 24 – AR_MCAST_REGv6_S message format

Name	Length	Description
AR.MSG	8-bits	Address Resolution Message Type For AR_MCAST_REGv6_S = 24
AR.IPv6	128-bits	IPv6 multicast address to be registered

8.4.12.2.7 AR_MCAST_REGv6_B

The table below shows the multicast address resolution register acknowledgment message sent from the Base Node to the Service Node.

Table 25 – AR_MCAST_REGv6_B message format

Name	Length	Description
AR.MSG	8-bits	Address Resolution Message Type For AR_MCAST_REGv6_B = 25
AR.IPv6	128-bits	IPv6 multicast address registered
Reserved	2-bits	Reserved. Should be encoded as 0.
AR.LCID	6-bits	LCID assigned to this IPv6 multicast address

The AR.IPv6 field is included in the AR_MCAST_REGv6_B message so that the Service Node can perform multiple overlapping registrations.

8.4.12.2.8 AR_MCAST_UNREGv6_S

The table below shows the multicast address resolution unregister message sent from the Service Node to the Base Node.

Table 26 – AR_MCAST_UNREGv6_S message format

Name	Length	Description
AR.MSG	8-bits	Address Resolution Message Type For AR_MCAST_UNREGv6_S = 26
AR.IPv6	128-bits	IPv6 multicast address to be unregistered

8.4.12.2.9 AR_MCAST_UNREGv6_B

The table below shows the multicast address resolution unregister acknowledgment message sent from the Base Node to the Service Node.

Table 27 – AR_MCAST_UNREGv6_B message format

Name	Length	Description
AR.MSG	8-bits	Address Resolution Message Type For AR_MCAST_UNREGv6_B = 27
AR.IPv6	128-bits	IPv6 multicast address unregistered

The AR.IPv6 field is included in the AR_MCAST_UNREGv6_B message so that the Service Node can perform multiple overlapping unregistrations.

8.4.12.3 Data connection establishment

8.4.12.3.1 Overview

The following PDU are sent during the data connection establishment phase. Please refer to Figure A. 1.

AR.MSG values from 24 to 255 are reserved.

8.4.12.3.2 AR_LOOKUPv6_S

The table below shows the address resolution lookup message sent from the Service Node to the Base Node.

Table 28 – AR_LOOKUPv6_S message format

Name	Length	Description
AR.MSG	8-bits	Address Resolution Message Type For AR_LOOKUPv6_S = 20
AR.IPv6	128-bits	IPv6 address to lookup

8.4.12.3.3 AR_LOOKUPv6_B

The table below shows the address resolution lookup response message sent from the Base Node to the Service Node.

Table 29 – AR_LOOKUPv6_B message format

Name	Length	Description
AR.MSG	8-bits	Address Resolution Message Type For AR_LOOKUPv6_B = 21
AR.IPv6	128-bits	IPv6 address looked up
AR.EUI-48	48-bits	EUI-48 for IPv6 address
AR.Status	8-bits	Lookup status, indicating if the address was found or an error occurred. 0 = found, AR.EUI-48 valid. 1 = unknown, AR.EUI-48 undefined

The lookup may fail if the requested address has not been registered. In that case, AR.Status will have a value equal to 1, and the contents of AR.EUI-48 will be undefined. The lookup is only successful when AR.Status is zero. In that case, the EUI-48 field contains the resolved address.

8.4.13 IPv6 Packet format

8.4.13.1 General

The following PDU formats are used for transferring IPv6 packets between Service Nodes. See ITU-T G.9904:2012 clause 9.6.8.3.

8.4.13.2 No negotiated header compression

When no header compression takes place, the IPv6 packet is simply sent as it is, without any additional header.

Table 30 – IPv6 Packet format without negotiated header compression

Name	Length	Description
IPv6.PKT	n-octets	The IPv6 Packet

8.4.13.3 Header compression

When LOWPAN_IPHC header compression takes place, and the next header compression is negotiated, the UDP/IPv6 packet is sent as shown in Table 31.

Table 31 – UDP/IPv6 Packet format with LOWPAN_IPHC header compression and LOWPAN_NHC

Name	Length	Description
IPv6.IPHC	2-octet	Dispatch + LOWPAN_IPHC encoding. With bit 5=1 indicating that the next header is compressed ,using LOWPAN_NHC format
IPv6.ncIPv6	n.m-octets	Non-Compressed IPv6 fields (or elided)
IPv6.HC_UDP	1-octet	Next header encoding
IPv6.ncUDP	n.m-octets	Non-Compressed UDP fields
Padding	0.m-octets	Padding to byte boundary
IPv6.DATA	n-octets	UDP data

Note that these fields are not necessarily aligned to byte boundaries. For example the IPv6.ncIPv6 field can be any number of bits. The IPv6.HC_UDP field follows directly afterwards, without any padding. Padding is only applied at the end of the complete compressed UDP/IPv6 header such that the UDP data is byte aligned.

When the IPv6 packet contains data other than UDP the following packet format is used as shown in Table 32.

Table 32 – IPv6 Packet format with LOWPAN_IPHC negotiated header compression

Name	Length	Description
IPv6.IPHC	2-octet	HC encoding. Bits 5 contain 0 indicating the next header byte is not compressed.
IPv6.ncIPv6	n.m-octets	Non-Compressed IPv6 fields
Padding	0.m-octets	Padding to byte boundary
IPv6.DATA	n-octets	IP Data

8.4.14 Connection data

8.4.14.1 Overview

When a connection is established between Service Nodes for the transfer of IPv6 packets, data is also transferred in the connection request packets. This data allows the negotiation of compression and notification of the IPv6 address. See ITU-T G.9904:2012 clause 9.6.8.4.

8.4.14.2 Connection data from the initiator

The table below shows the connection data sent by the initiator.

Table 33 – IPv6 Connection signalling data sent by the initiator

Name	Length	Description
Reserved	6-bits	Should be encoded as zero in this version of the IPv6 SSCS protocol
Data.HCNH	2-bit	Header Compression negotiated Data.HC = 0 – No compression requested Data.HC = 1 – LOWPAN_NH Data.HC = 2 – stateful address compression. Data.HC = 3 – LOWPAN_NH and stateful address compression.
Data.IPv6	128-bits	IPv6 address of the initiator

If the device accepts the connection, it should copy the Data.IPv6 address into a new table entry along with the negotiated Data.HC value.

8.4.14.3 Connection data from the responder

The table shows the connection data sent in response to the connection request.

Table 34 – IPv6 Connection signalling data sent by the responder

Name	Length	Description
Reserved	6-bits	Should be encoded as zero in this version of the IPv6 SSCS protocol
Data.HC	2-bit	Header Compression negotiated • Data.HC = 0 – No compression requested NOTE: When stateless address compression is used all nodes shall support it. When the stateless address compression is not used then the node notify by this value, its compression capability. • Data.HC = 1 – LOWPAN_NH • Data.HC = 2 – stateful address compression. • Data.HC = 3 – LOWPAN_NH and stateful address compression.

All nodes support stateless address compression.

The next header compression scheme and stateful address compression can only be used when it is supported by both Service Nodes. The responder may only set Data.HC to the same value as that received from the initiator or a value lower than the one received. When the same value is used, it indicates that the requested compression scheme has been negotiated and will be used for the connection. Setting Data.HC to lower value allows the responder to deny the request for that header compression scheme.

Annex A
(informative)

A.1 Data exchange between two IP communication peers

This example shows for IPv4 layer and IPv4 SSCS, the exchange of service primitives between a Service Node (192.168.0.100/24) and a Base Node when the former wants to exchange IPv4 packets with a third Service Node (192.168.0.101/24) whose IP address is in the same IP subnetwork.

NOTE: For IPv6 SSCS, the principle is the same.

This example takes the following assumptions:

- Service node (192.168.0.100) IPv4 SSCS has not yet a connection established, so it needs to start a connection establishment with the Base Node and register its IP address with the Base Node prior to the exchange of IP packets;
- Service node (192.168.0.101) has already registered its IP Address with the Base Node.

The steps illustrated in Figure A.1 are:

1. The IPv4 layer of the Service Node (192.168.0.100) invokes the CL_IPv4_ESTABLISH.request primitive. To establish IPv4 SSCS connection, it is required:
    a. To establish a connection with the Base Node so all address resolution messages can be exchanged over it;
    b. To inform the Service Node MAC layer that IPv4 SSCS is ready to receive all IPv4 broadcasts packets. Note the difference between broadcast and multicast. To join a multicast group, the Service Node will need to inform the Base Node of the group it wants to join. This is illustrated in section A.2.
2. The IPv4_ layer, once the IPv4 SSCS is established, needs to register its IP address in the Base Node. To do so, it will use the already established connection.
3. Whenever the IPv4 layer needs to deliver an IPv4 packet to a new destination IP address, the following two steps are to be done (in this example, the destination IP address is 192.168.0.101).
    a. As the IPv4 destination address is new, the IPv4 SSCS needs to request the EUI-48 associated to that IPv4 address. To do so, a lookup request message is sent to the Base Node.
    b. Upon the reception of the EUI-48, a new connection (type = TYPE_CL_IPv4_UNICAST – see ITU-T G.9904:2012 Table C.1 – is established so that all IP packets to be exchanged between 192.168.0.100 and 192.168.0.101 will use that connection.

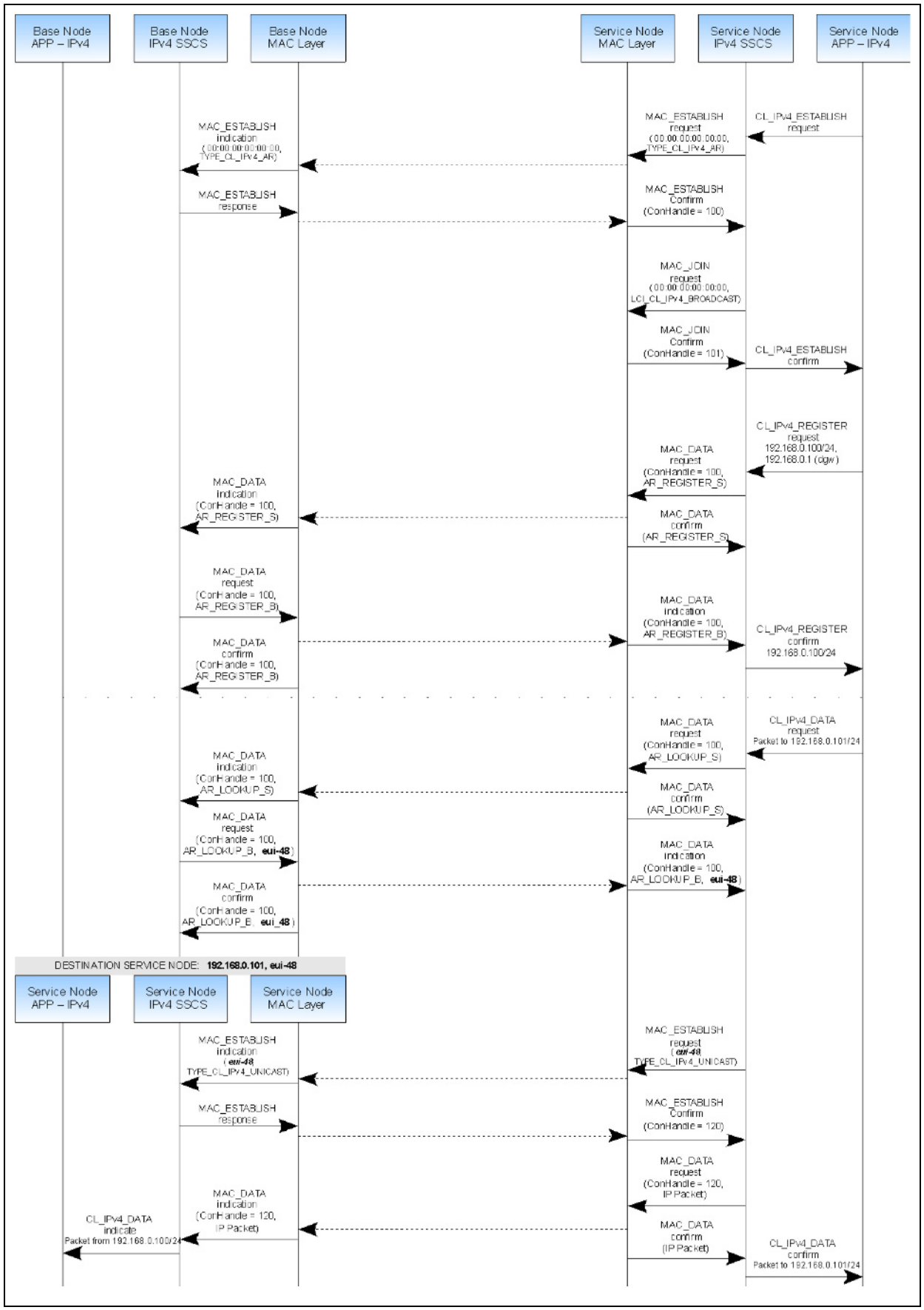


Figure A.1 – MSC of IPv4 SSCS services

A.2 Joining a multicast group

The figure below illustrates how a Service Node joins a multicast group. As mentioned before, main difference between multicast and broadcast is related to the messages exchanged. For broadcast, the MAC layer will immediately issue a MAC_JOIN.confirm primitive since it does not need to perform any end-to-end operation. For multicast, the MAC_JOIN.confirm is only sent once the signalling between the Service Node and Base Node is complete.

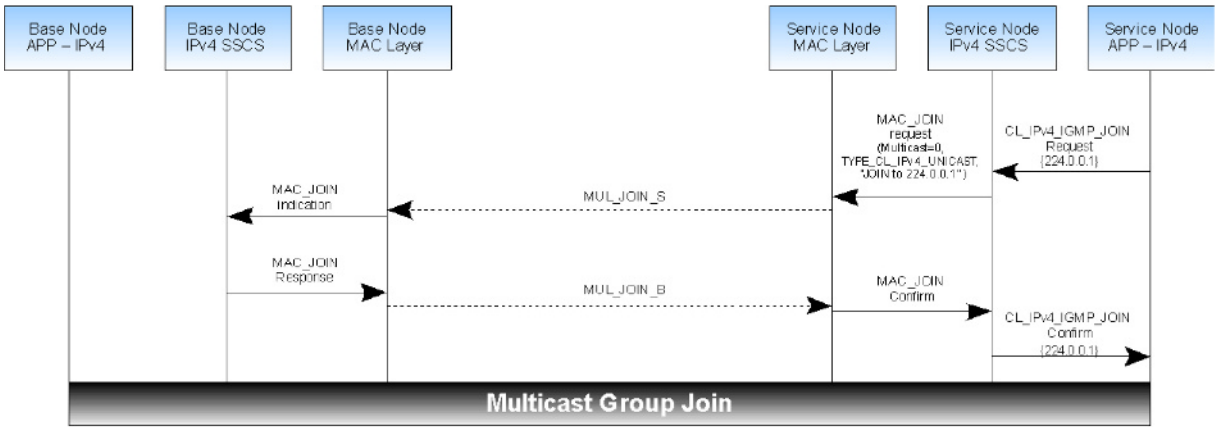


Figure A.2 – MSC for joining an IPv4 multicast group

Annex B
(informative)
EN 61334-4-32 profile: Error cases during connection establishment

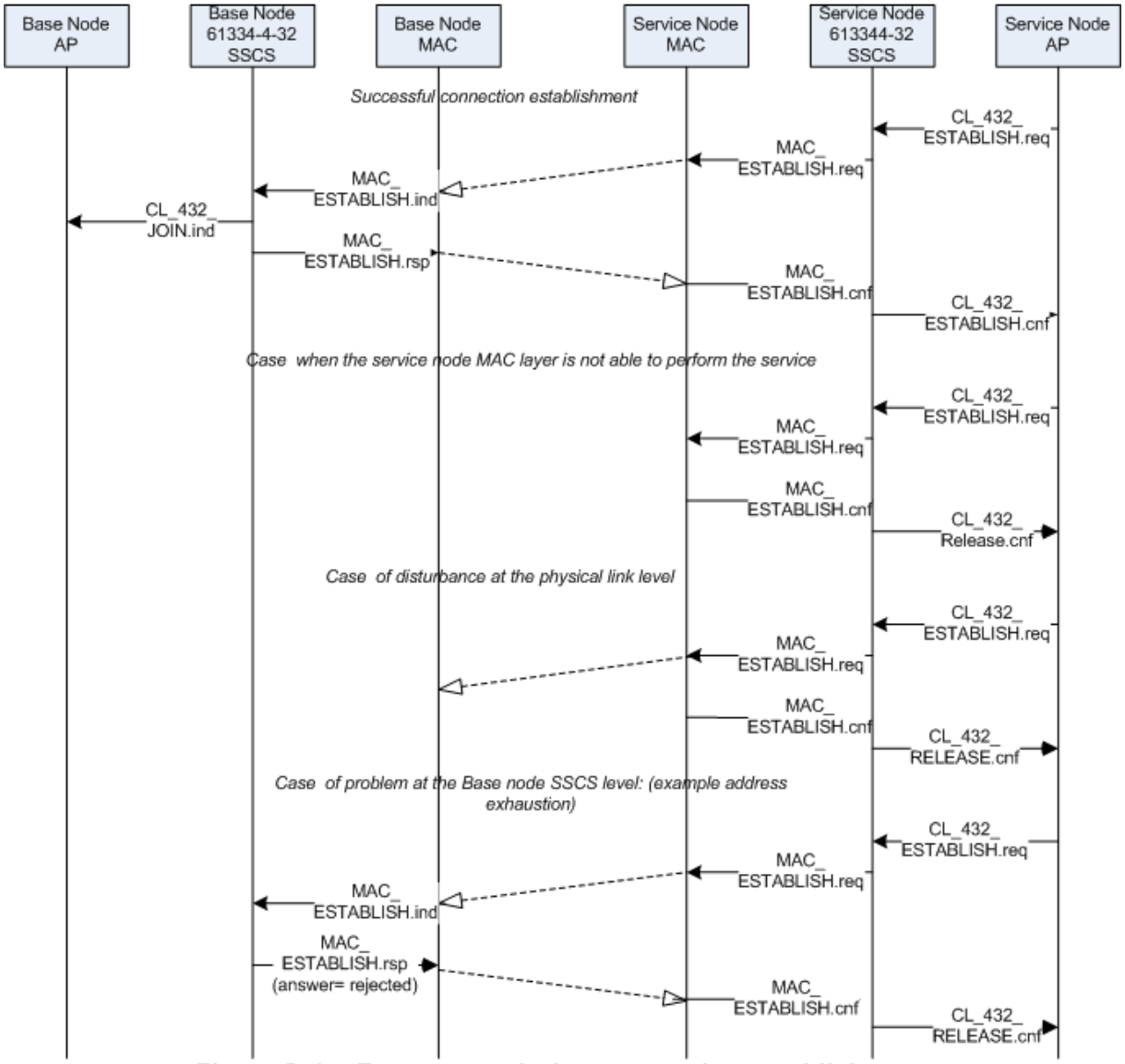


Figure B.1 – Error cases during connection establishment

Annex C
(informative)
PRIME encoding examples

C.1 ACSE APDUs and xDLMS APDUs carried by MAC frames using the EN 61334-4-32 SSCS

In these examples, the following communication sequence is shown, when the DLMS/COSEM PRIME profile is used with the EN 61334-4-32 SSCS.

- the initiator establishes an AA;
- it reads the time attribute of the Clock object;
- it reads load profile to show both PRIME segmentation and DLMS block transfer;
- the initiator releases AA.

The traces have been taken from a protocol analyser. DLMS/COSEM payload is highlighted in blue.

Open association on the Logical device LsapDest=0x01 and Management Client LsapSrc=0x01: MAC frame carrying an AARQ APDU

```
Hex:
0000    00 40 29 05 00 00 60 18 3c 87 07 00 90 01 01 60
0010    34 a1 09 06 07 60 85 74 05 08 01 01 8a 02 07 80
0020    8b 07 60 85 74 05 08 02 01 ac 08 80 06 31 32 33
0030    34 35 36 be 10 04 0e 01 00 00 00 06 5f 1f 04 00
0040    00 30 1d ff ff 63 b0 fb a5


PRIME
    00.. .... = PRIME PDU Unused: 2bits. Must be always 00
    ..00 .... = PRIME PDU Header Type: Generic MAC header format GPDU
(0)
    Generic MAC Header
        .... 0000  0... .... = PRIME Generic MAC Header Reserved: 0
        .1.. .... = PRIME Generic MAC HDR.DO field Downlink: 1
        ..00 0000 = PRIME Generic MAC HDR.LEVEL field, Header level: 0
        PRIME Generic MAC checksum: 41
    GPDU Header
        000. .... = PRIME PDU General Header reserved: 0
        ...0 .... = PRIME PDU General Header NAD: 0
        .... 01.. = PRIME PDU General Header Prio: 1
        .... ..0. = PRIME PDU General Header control: 0
        .... ...1  0000 0000 = PRIME PDU General Header LCID: 256
        PRIME PDU General Header SID: 0
        0110 0000  0001 10.. = PRIME PDU General Header LNID: 6150
        .... ..0. = PRIME PDU General Header padding: 0
        .... ...0  0011 1100 = PRIME PDU General Header length: 60
    GPDU ARQ
        1... .... = PRIME PDU ARQ M: 1
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..00 0111 = PRIME PDU ARQ PKTID: 7
        0... .... = PRIME PDU ARQ M: 0
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..00 0111 = PRIME PDU ARQ ACKID: 7
    GPDU Payload
    PRIME PDU CRC: 1672543141 (63 b0 fb a5) at the end of frame
PRIME 432
    00.. .... = PRIME SAR Type: 0
    ..00 0000 = PRIME SAR Nseg: 0
```

```
    1... .... = PRIME 432 one bit: 1
    .00. .... = PRIME 432 Command: 0
    ...1 .... = PRIME 432 Command/Response: 1
    .... 0000 = PRIME 432 Qualifier: 0
    PRIME 432 Destination L_SAP: 1
    PRIME 432 Source L_SAP: 1
    PRIME 432 data: 6034a109060760857405080101 8a0207808b07608574...

-- 432 Data explanation:
60 34 // AARQ
a1 09 06 07 60 85 74 05 08 01 01 // application-context-name
8a 02 07 80  // acse-requirements
8b 07 60 85 74 05 08 02 01 // mechanism-name
ac 08 80 06 31 32 33 34 35 36 // calling-authentication-value
be 10 04 0e 01 00 00 00 06 5f 1f 04 00 00 30 1d ff ff
    // user-information xDLMS InititateRequest
```

Response: MAC frame carrying an AARE APDU

```
Hex:
0000    00 00 ee 15 00 00 e0 08 31 bd 03 00 90 01 01 61
0010    29 a1 09 06 07 60 85 74 05 08 01 01 a2 03 02 01
0020    00 a3 05 a1 03 02 01 00 be 10 04 0e 08 00 06 5f
0030    1f 04 00 00 10 1d 00 f8 00 07 92 0f a2 d7

PRIME
00.. .... = PRIME PDU Unused: 0
    ..00 .... = PRIME PDU Header Type: GPDU (0)
    Generic MAC Header
        .... 0000  0... .... = PRIME Generic MAC Header Reserved: 0
        .0.. .... = PRIME Generic MAC Downlink: 0
        ..00 0000 = PRIME Generic MAC Header level: 0
        PRIME Generic MAC checksum: 238
    GPDU Header
        000. .... = PRIME PDU General Header reserved: 0
        ...1 .... = PRIME PDU General Header NAD: 1
        .... 01.. = PRIME PDU General Header Prio: 1
        .... ..0. = PRIME PDU General Header control: 0
        .... ...1  0000 0000 = PRIME PDU General Header LCID: 256
        PRIME PDU General Header SID: 0
        1110 0000  0000 10.. = PRIME PDU General Header LNID: 14338
        .... ..0. = PRIME PDU General Header padding: 0
        .... ...0  0011 0001 = PRIME PDU General Header length: 49
    GPDU ARQ
        1... .... = PRIME PDU ARQ M: 1
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..11 1101 = PRIME PDU ARQ PKTID: 61
        0... .... = PRIME PDU ARQ M: 0
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..00 0011 = PRIME PDU ARQ ACKID: 3
    GPDU Payload
    PRIME PDU CRC: 2450498263 (92 0f a2 d7) at the end of frame
PRIME 432
    00.. .... = PRIME SAR Type: 0
    ..00 0000 = PRIME SAR Nseg: 0
    1... .... = PRIME 432 one bit: 1
    .00. .... = PRIME 432 Command: 0
    ...1 .... = PRIME 432 Command/Response: 1
    .... 0000 = PRIME 432 Qualifier: 0
    PRIME 432 Destination L_SAP: 1
    PRIME 432 Source L_SAP: 1
    PRIME 432 data: 6129a109060760857405080101a203020100a305a10302...
```

```
-- 432 Data explanation:

61 29 // AARE APDU
  a1 09 06 07 60 85 74 05 08 01 01 // application-context-name
  a2 03 02 01 00 // result
  a3 05 a1 03 02 01 00 // result-source-diagnostic
  be 10 04 0e 08 00 06 5f 1f 04 00 00 10 1d 00 f8 00 07
  // user-information xDLMS-InititateResponse
```

Read date and time, get-request

```
0000    00 40 29 05 00 00 e0 08 13 83 3e 00 90 01 01 c0
0010    01 c1 00 08 00 00 01 00 00 ff 02 00 7f 80 a6 4d

PRIME
    00.. .... = PRIME PDU Unused: 0
    ..00 .... = PRIME PDU Header Type: GPDU (0)
    Generic MAC Header
        .... 0000  0... .... = PRIME Generic MAC Header Reserved: 0
        .1.. .... = PRIME Generic MAC Downlink: 1
        ..00 0000 = PRIME Generic MAC Header level: 0
        PRIME Generic MAC checksum: 41
    GPDU Header
        000. .... = PRIME PDU General Header reserved: 0
        ...0 .... = PRIME PDU General Header NAD: 0
        .... 01.. = PRIME PDU General Header Prio: 1
        .... ..0. = PRIME PDU General Header control: 0
        .... ...1  0000 0000 = PRIME PDU General Header LCID: 256
        PRIME PDU General Header SID: 0
        1110 0000  0000 10.. = PRIME PDU General Header LNID: 14338
        .... ..0. = PRIME PDU General Header padding: 0
        .... ...0  0001 0011 = PRIME PDU General Header length: 19
    GPDU ARQ
        1... .... = PRIME PDU ARQ M: 1
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..00 0011 = PRIME PDU ARQ PKTID: 3
        0... .... = PRIME PDU ARQ M: 0
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..11 1110 = PRIME PDU ARQ ACKID: 62
    GPDU Payload
    PRIME PDU CRC: 2139137613 (7f 80 a6 4d) at the end of frame
PRIME 432
    00.. .... = PRIME SAR Type: 0
    ..00 0000 = PRIME SAR Nseg: 0
    1... .... = PRIME 432 one bit: 1
    .00. .... = PRIME 432 Command: 0
    ...1 .... = PRIME 432 Command/Response: 1
    .... 0000 = PRIME 432 Qualifier: 0
    PRIME 432 Destination L_SAP: 1
    PRIME 432 Source L_SAP: 1
    PRIME 432 data: c001c100080000010000ff0200

-- 432 Data explanation:

c0 01 // get-request, get-request-normal
  c1 // invoke-id-and-priority
  00 08 // class-id
  00 00 01 00 00 ff // instance-id
  02 // attribute-id
  00
```

Read date and time, get-response

```
Hex:
0000    00 00 ee 15 00 00 e0 08 18 be 04 00 90 01 01 c4
0010    01 c1 00 09 0c 07 db 03 02 03 0a 34 08 ff 80 00
0020    04 23 1c aa 32

PRIME
    00.. .... = PRIME PDU Unused: 0
    ..00 .... = PRIME PDU Header Type: GPDU (0)
    Generic MAC Header
        .... 0000  0... .... = PRIME Generic MAC Header Reserved: 0
        .0.. .... = PRIME Generic MAC Downlink: 0
        ..00 0000 = PRIME Generic MAC Header level: 0
        PRIME Generic MAC checksum: 238
    GPDU Header
        000. .... = PRIME PDU General Header reserved: 0
        ...1 .... = PRIME PDU General Header NAD: 1
        .... 01.. = PRIME PDU General Header Prio: 1
        .... ..0. = PRIME PDU General Header control: 0
        .... ...1  0000 0000 = PRIME PDU General Header LCID: 256
        PRIME PDU General Header SID: 0
        1110 0000  0000 10.. = PRIME PDU General Header LNID: 14338
        .... ..0. = PRIME PDU General Header padding: 0
        .... ...0  0001 1000 = PRIME PDU General Header length: 24
    GPDU ARQ
        1... .... = PRIME PDU ARQ M: 1
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..11 1110 = PRIME PDU ARQ PKTID: 62
        0... .... = PRIME PDU ARQ M: 0
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..00 0100 = PRIME PDU ARQ ACKID: 4
    GPDU Payload
    PRIME PDU CRC: 589081138 (23 1c aa 32) at the end of frame
PRIME 432
    00.. .... = PRIME SAR Type: 0
    ..00 0000 = PRIME SAR Nseg: 0
    1... .... = PRIME 432 one bit: 1
    .00. .... = PRIME 432 Command: 0
    ...1 .... = PRIME 432 Command/Response: 1
    .... 0000 = PRIME 432 Qualifier: 0
    PRIME 432 Destination L_SAP: 1
    PRIME 432 Source L_SAP: 1
    PRIME 432 data: c401c100090c07db0302030a3408ff800004

-- 432 Data explanation:

c4 01 // get-response, get-response-normal
  c1  // invoke-id-and-priority
  00 09 0c // result, data, octet-string of 12
    07db0302030a3408ff800004 // value
```

Read load profile, get-request with access selector:

```
Hex:

0000    00 40 29 05 00 00 e0 08 46 84 3f 00 90 01 01 c0
0010    01 c1 00 07 01 00 63 01 00 ff 02 01 01 02 04 02
0020    04 12 00 08 09 06 00 00 01 00 00 ff 0f 02 12 00
```

```
0030    00 09 0c 07 db 03 01 ff 10 00 00 ff 80 00 00 09
0040    0c 07 db 03 01 ff 17 00 00 ff 80 00 00 01 00 6b
0050    71 d8 42

PRIME
    00.. .... = PRIME PDU Unused: 0
    ..00 .... = PRIME PDU Header Type: GPDU (0)
    Generic MAC Header
        .... 0000  0... .... = PRIME Generic MAC Header Reserved: 0
        .1.. .... = PRIME Generic MAC Downlink: 1
        ..00 0000 = PRIME Generic MAC Header level: 0
        PRIME Generic MAC checksum: 41
    GPDU Header
        000. .... = PRIME PDU General Header reserved: 0
        ...0 .... = PRIME PDU General Header NAD: 0
        .... 01.. = PRIME PDU General Header Prio: 1
        .... ..0. = PRIME PDU General Header control: 0
        .... ...1  0000 0000 = PRIME PDU General Header LCID: 256
        PRIME PDU General Header SID: 0
        1110 0000  0000 10.. = PRIME PDU General Header LNID: 14338
        .... ..0. = PRIME PDU General Header padding: 0
        .... ...0  0100 0110 = PRIME PDU General Header length: 70
    GPDU ARQ
        1... .... = PRIME PDU ARQ M: 1
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..00 0100 = PRIME PDU ARQ PKTID: 4
        0... .... = PRIME PDU ARQ M: 0
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..11 1111 = PRIME PDU ARQ ACKID: 63
    GPDU Payload
    PRIME PDU CRC: 1802623042 (6b 71 d8 42)
PRIME 432
    00.. .... = PRIME SAR Type: 0
    ..00 0000 = PRIME SAR Nseg: 0
    1... .... = PRIME 432 one bit: 1
    .00. .... = PRIME 432 Command: 0
    ...1 .... = PRIME 432 Command/Response: 1
    .... 0000 = PRIME 432 Qualifier: 0
    PRIME 432 Destination L_SAP: 1
    PRIME 432 Source L_SAP: 1
    PRIME 432 data: c001c100070100630100ff020101020402041200080906...

-- 432 Data explanation:

c0 01                               // get-request, get-request-normal
c1                                  // invoke-id-and-priority
00 07                               // class-id
01 00 63 01 00 ff                   // instance-id
02                                  // attribute-id
01 01                               // access-selector
  02 04                             // structure of 4
    02 04                           // structure of 4
      12 00 08                      // long-unsigned
      09 06  00 00 01 00 00 ff      // octet-string
      0f 02                         // integer
      12 00 00                      // long unsigned
    09 0c 07 db 03 01 ff 10 00 00 ff 80 00 00 // octet-string
    09 0c 07 db 03 01 ff 17 00 00 ff 80 00 00 // octet-string
    01 00                           // array of 0


Read load profile, get-response (first SAR segment):
```

```
Hex:
0000    00 00 ee 15 00 00 e0 08 4a ff 05 02 90 01 01 c4
0010    02 c1 00 00 00 00 01 00 81 c4 01 08 02 08 09 0c
0020    07 db 03 01 02 10 00 00 ff 80 00 04 11 00 06 00
0030    00 00 00 06 00 00 00 00 06 00 00 00 00 06 00 00
0040    00 00 06 00 00 00 00 06 00 00 00 00 02 08 09 0c
0050    07 db 03 e1 3e b9 b6

PRIME
    00.. .... = PRIME PDU Unused: 0
    ..00 .... = PRIME PDU Header Type: GPDU (0)
    Generic MAC Header
        .... 0000  0... .... = PRIME Generic MAC Header Reserved: 0
        .0.. .... = PRIME Generic MAC Downlink: 0
        ..00 0000 = PRIME Generic MAC Header level: 0
        PRIME Generic MAC checksum: 238
    GPDU Header
        000. .... = PRIME PDU General Header reserved: 0
        ...1 .... = PRIME PDU General Header NAD: 1
        .... 01.. = PRIME PDU General Header Prio: 1
        .... ..0. = PRIME PDU General Header control: 0
        .... ...1  0000 0000 = PRIME PDU General Header LCID: 256
        PRIME PDU General Header SID: 0
        1110 0000  0000 10.. = PRIME PDU General Header LNID: 14338
        .... ..0. = PRIME PDU General Header padding: 0
        .... ...0  0100 1010 = PRIME PDU General Header length: 74
    GPDU ARQ
        1... .... = PRIME PDU ARQ M: 1
        .1.. .... = PRIME PDU ARQ FLUSH: 1
        ..11 1111 = PRIME PDU ARQ PKTID: 63
        0... .... = PRIME PDU ARQ M: 0
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..00 0101 = PRIME PDU ARQ ACKID: 5
    GPDU Payload
    PRIME PDU CRC: 3778984374 (e1 3e b9 b6)
PRIME 432
    00.. .... = PRIME SAR Type: 0
    ..00 0010 = PRIME SAR Nseg: 2
    1... .... = PRIME 432 one bit: 1
    .00. .... = PRIME 432 Command: 0
    ...1 .... = PRIME 432 Command/Response: 1
    .... 0000 = PRIME 432 Qualifier: 0
    PRIME 432 Destination L_SAP: 1
    PRIME 432 Source L_SAP: 1
    PRIME 432 data: c402c100000000010081c401080208090c07db03010210...

-- 432 Data explanation (raw-data value continues in following
segments):

c4 02                  // get-response, get-response-with-data-block
  c1                   // invoke-id-and-priority
  00                   // last-block (FALSE)
  00 00 00 01          // block number (1)
  00 81 c4             // result, raw-data
    01 08 02 08 09 0c 07 db 03 01 02 10 00 00 ff 80 // value
    00 04 11 00 06 00 00 00 00 06 00 00 00 00 06 00 // value
    00 00 00 06 00 00 00 00 06 00 00 00 00 06 00 00 // value
    00 00 02 08 09 0c 07 db 03 //value
```

Read load profile, get-response (second SAR segment):

```
Hex:
0000    00 00 ee 15 00 00 e0 08 4a c0 05 40 01 02 11 00
0010    00 ff 80 00 04 11 00 06 00 00 00 00 06 00 00 00
0020    00 06 00 00 00 00 06 00 00 00 00 06 00 00 00 00
0030    06 00 00 00 00 02 08 09 0c 07 db 03 01 02 12 00
0040    00 ff 80 00 04 11 00 06 00 00 00 00 06 00 00 00
0050    00 06 00 a0 4e 93 4d

PRIME
    00.. .... = PRIME PDU Unused: 0
    ..00 .... = PRIME PDU Header Type: GPDU (0)
    Generic MAC Header
        .... 0000  0... .... = PRIME Generic MAC Header Reserved: 0
        .0.. .... = PRIME Generic MAC Downlink: 0
        ..00 0000 = PRIME Generic MAC Header level: 0
        PRIME Generic MAC checksum: 238
    GPDU Header
        000. .... = PRIME PDU General Header reserved: 0
        ...1 .... = PRIME PDU General Header NAD: 1
        .... 01.. = PRIME PDU General Header Prio: 1
        .... ..0. = PRIME PDU General Header control: 0
        .... ...1  0000 0000 = PRIME PDU General Header LCID: 256
        PRIME PDU General Header SID: 0
        1110 0000  0000 10.. = PRIME PDU General Header LNID: 14338
        .... ..0. = PRIME PDU General Header padding: 0
        .... ...0  0100 1010 = PRIME PDU General Header length: 74
    GPDU ARQ
        1... .... = PRIME PDU ARQ M: 1
        .1.. .... = PRIME PDU ARQ FLUSH: 1
        ..00 0000 = PRIME PDU ARQ PKTID: 0
        0... .... = PRIME PDU ARQ M: 0
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..00 0101 = PRIME PDU ARQ ACKID: 5
    GPDU Payload
    PRIME PDU CRC: 2689504077 (a0 4e 93 4d)
PRIME 432
    01.. .... = PRIME SAR Type: 1
    ..00 0000 = PRIME SAR Nseg: 0
    PRIME 432 data: 0102110000ff800004110006000000000600000000600...

-- 432 Data continues from previous segment.
```

Read load profile, get-response (last SAR segment):

```
Hex:
0000    00 00 ee 15 00 00 e0 08 47 81 05 81 00 00 00 06
0010    00 00 00 00 06 00 00 00 00 06 00 00 00 00 02 08
0020    09 0c 07 db 03 01 02 13 00 00 ff 80 00 04 11 00
0030    06 00 00 00 00 06 00 00 00 00 06 00 00 00 00 06
0040    00 00 00 00 06 00 00 00 00 06 00 00 00 00 02 08
0050    7f 85 52 2d


PRIME
    00.. .... = PRIME PDU Unused: 0
    ..00 .... = PRIME PDU Header Type: GPDU (0)
    Generic MAC Header
        .... 0000  0... .... = PRIME Generic MAC Header Reserved: 0
        .0.. .... = PRIME Generic MAC Downlink: 0
```

```
        ..00 0000 = PRIME Generic MAC Header level: 0
        PRIME Generic MAC checksum: 238
    GPDU Header
        000. .... = PRIME PDU General Header reserved: 0
        ...1 .... = PRIME PDU General Header NAD: 1
        .... 01.. = PRIME PDU General Header Prio: 1
        .... ..0. = PRIME PDU General Header control: 0
        .... ...1  0000 0000 = PRIME PDU General Header LCID: 256
        PRIME PDU General Header SID: 0
        1110 0000  0000 10.. = PRIME PDU General Header LNID: 14338
        .... ..0. = PRIME PDU General Header padding: 0
        .... ...0  0100 0111 = PRIME PDU General Header length: 71
    GPDU ARQ
        1... .... = PRIME PDU ARQ M: 1
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..00 0001 = PRIME PDU ARQ PKTID: 1
        0... .... = PRIME PDU ARQ M: 0
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..00 0101 = PRIME PDU ARQ ACKID: 5
    GPDU Payload
    PRIME PDU CRC: 2139443757 (7f 85 52 2d)
PRIME 432
    10.. .... = PRIME SAR Type: 2
    ..00 0001 = PRIME SAR Nseg: 1
    PRIME 432 data: 0000000600000000060000000006000000000208090c07...

-- 432 Data continues from previous segment.
```

Read load profile, get-next-block:

```
Hex:
0000    00 40 29 05 00 00 e0 08 0d 85 02 00 90 01 01 c0
0010    02 c1 00 00 00 01 ce 1f 62 ab

PRIME
    00.. .... = PRIME PDU Unused: 0
    ..00 .... = PRIME PDU Header Type: GPDU (0)
    Generic MAC Header
        .... 0000  0... .... = PRIME Generic MAC Header Reserved: 0
        .1.. .... = PRIME Generic MAC Downlink: 1
        ..00 0000 = PRIME Generic MAC Header level: 0
        PRIME Generic MAC checksum: 41
    GPDU Header
        000. .... = PRIME PDU General Header reserved: 0
        ...0 .... = PRIME PDU General Header NAD: 0
        .... 01.. = PRIME PDU General Header Prio: 1
        .... ..0. = PRIME PDU General Header control: 0
        .... ...1  0000 0000 = PRIME PDU General Header LCID: 256
        PRIME PDU General Header SID: 0
        1110 0000  0000 10.. = PRIME PDU General Header LNID: 14338
        .... ..0. = PRIME PDU General Header padding: 0
        .... ...0  0000 1101 = PRIME PDU General Header length: 13
    GPDU ARQ
        1... .... = PRIME PDU ARQ M: 1
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..00 0101 = PRIME PDU ARQ PKTID: 5
        0... .... = PRIME PDU ARQ M: 0
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..00 0010 = PRIME PDU ARQ ACKID: 2
    GPDU Payload
    PRIME PDU CRC: 3458163371 (ce 1f 62 ab)
```

```
PRIME 432
    00.. .... = PRIME SAR Type: 0
    ..00 0000 = PRIME SAR Nseg: 0
    1... .... = PRIME 432 one bit: 1
    .00. .... = PRIME 432 Command: 0
    ...1 .... = PRIME 432 Command/Response: 1
    .... 0000 = PRIME 432 Qualifier: 0
    PRIME 432 Destination L_SAP: 1
    PRIME 432 Source L_SAP: 1
    PRIME 432 data: c002c100000001

-- 432 Data explanation:

c0 02                  // get-request, get-request-for-next-data-block
  c1                   // invoke-id-and-priority
  00000001             // block-number
```

Read load profile, get-response (block 2, first SAR segment):

```
Hex:
Hex:
0000    00 00 ee 15 00 00 e0 08 48 c2 06 02 90 01 01 c4
0010    02 c1 01 00 00 00 02 00 81 be 09 0c 07 db 03 01
0020    02 14 00 00 ff 80 00 04 11 00 06 00 00 00 00 06
0030    00 00 00 00 06 00 00 00 00 06 00 00 00 00 06 00
0040    00 00 00 06 00 00 00 00 02 08 09 0c 07 db 03 01
0050    02 84 28 b4 9a

PRIME
    00.. .... = PRIME PDU Unused: 0
    ..00 .... = PRIME PDU Header Type: GPDU (0)
    Generic MAC Header
        .... 0000  0... .... = PRIME Generic MAC Header Reserved: 0
        .0.. .... = PRIME Generic MAC Downlink: 0
        ..00 0000 = PRIME Generic MAC Header level: 0
        PRIME Generic MAC checksum: 238
    GPDU Header
        000. .... = PRIME PDU General Header reserved: 0
        ...1 .... = PRIME PDU General Header NAD: 1
        .... 01.. = PRIME PDU General Header Prio: 1
        .... ..0. = PRIME PDU General Header control: 0
        .... ...1  0000 0000 = PRIME PDU General Header LCID: 256
        PRIME PDU General Header SID: 0
        1110 0000  0000 10.. = PRIME PDU General Header LNID: 14338
        .... ..0. = PRIME PDU General Header padding: 0
        .... ...0  0100 1000 = PRIME PDU General Header length: 72
    GPDU ARQ
        1... .... = PRIME PDU ARQ M: 1
        .1.. .... = PRIME PDU ARQ FLUSH: 1
        ..00 0010 = PRIME PDU ARQ PKTID: 2
        0... .... = PRIME PDU ARQ M: 0
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..00 0110 = PRIME PDU ARQ ACKID: 6
    GPDU Payload
    PRIME PDU CRC: 2217260186 (84 28 b4 9a)
PRIME 432
    00.. .... = PRIME SAR Type: 0
    ..00 0010 = PRIME SAR Nseg: 2
    1... .... = PRIME 432 one bit: 1
```

```
    .00. .... = PRIME 432 Command: 0
    ...1 .... = PRIME 432 Command/Response: 1
    .... 0000 = PRIME 432 Qualifier: 0
    PRIME 432 Destination L_SAP: 1
    PRIME 432 Source L_SAP: 1
    PRIME 432 data: c402c101000000020081be090c07db030102140000ff80...

-- 432 Data explanation (raw-data value continues in following
segments):

c4 02                  // get-response, get-response-with-data-block
c1                     // invoke-id-and-priority
01                     // last-block (TRUE)
00 00 00 02            // block number (2)
00 81 be               // result, raw-data
09 0c 07 db 03 01 02 14 00 00 ff 80 00 04 11 00 // value
06 00 00 00 00 06 00 00 00 00 06 00 00 00 00 06 // value
00 00 00 00 06 00 00 00 00 06 00 00 00 00 02 08 // value
09 0c 07 db 03 01 02 // value
```

Read load profile, get-response (block 2, second SAR segment):

```
Hex:
0000    00 00 ee 15 00 00 e0 08 48 c3 06 40 15 00 00 ff
0010    80 00 04 11 00 06 00 00 00 00 06 00 00 00 00 06
0020    00 00 00 00 06 00 00 00 00 06 00 00 00 00 06 00
0030    00 00 00 02 08 09 0c 07 db 03 01 02 16 00 00 ff
0040    80 00 04 11 00 06 00 00 00 00 06 00 00 00 00 06
0050    00 68 7f ac 47

PRIME
    00.. .... = PRIME PDU Unused: 0
    ..00 .... = PRIME PDU Header Type: GPDU (0)
    Generic MAC Header
        .... 0000  0... .... = PRIME Generic MAC Header Reserved: 0
        .0.. .... = PRIME Generic MAC Downlink: 0
        ..00 0000 = PRIME Generic MAC Header level: 0
        PRIME Generic MAC checksum: 238
    GPDU Header
        000. .... = PRIME PDU General Header reserved: 0
        ...1 .... = PRIME PDU General Header NAD: 1
        .... 01.. = PRIME PDU General Header Prio: 1
        .... ..0. = PRIME PDU General Header control: 0
        .... ...1  0000 0000 = PRIME PDU General Header LCID: 256
        PRIME PDU General Header SID: 0
        1110 0000  0000 10.. = PRIME PDU General Header LNID: 14338
        .... ..0. = PRIME PDU General Header padding: 0
        .... ...0  0100 1000 = PRIME PDU General Header length: 72
    GPDU ARQ
        1... .... = PRIME PDU ARQ M: 1
        .1.. .... = PRIME PDU ARQ FLUSH: 1
        ..00 0011 = PRIME PDU ARQ PKTID: 3
        0... .... = PRIME PDU ARQ M: 0
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..00 0110 = PRIME PDU ARQ ACKID: 6
    GPDU Payload
    PRIME PDU CRC: 1753197639 (68 7f ac 47)
PRIME 432
    01.. .... = PRIME SAR Type: 1
    ..00 0000 = PRIME SAR Nseg: 0
    PRIME 432 data: 150000ff800004110006000000000600000000060000000...
```

```
-- 432 Data continues from previous segment.
```

Read load profile, get-response (block 2, last SAR segment):

```
Hex:
0000    00 00 ee 15 00 00 e0 08 45 84 06 81 00 00 00 06
0010    00 00 00 00 06 00 00 00 00 06 00 00 00 00 02 08
0020    09 0c 07 db 03 01 02 17 00 00 ff 80 00 04 11 00
0030    06 00 00 00 00 06 00 00 00 00 06 00 00 00 00 06
0040    00 00 00 00 06 00 00 00 00 06 00 00 00 00 e6 1c
0050    51 26

PRIME
    00.. .... = PRIME PDU Unused: 0
    ..00 .... = PRIME PDU Header Type: GPDU (0)
    Generic MAC Header
        .... 0000  0... .... = PRIME Generic MAC Header Reserved: 0
        .0.. .... = PRIME Generic MAC Downlink: 0
        ..00 0000 = PRIME Generic MAC Header level: 0
        PRIME Generic MAC checksum: 238
    GPDU Header
        000. .... = PRIME PDU General Header reserved: 0
        ...1 .... = PRIME PDU General Header NAD: 1
        .... 01.. = PRIME PDU General Header Prio: 1
        .... ..0. = PRIME PDU General Header control: 0
        .... ...1  0000 0000 = PRIME PDU General Header LCID: 256
        PRIME PDU General Header SID: 0
        1110 0000  0000 10.. = PRIME PDU General Header LNID: 14338
        .... ..0. = PRIME PDU General Header padding: 0
        .... ...0  0100 0101 = PRIME PDU General Header length: 69
    GPDU ARQ
        1... .... = PRIME PDU ARQ M: 1
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..00 0100 = PRIME PDU ARQ PKTID: 4
        0... .... = PRIME PDU ARQ M: 0
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..00 0110 = PRIME PDU ARQ ACKID: 6
    GPDU Payload
    PRIME PDU CRC: 3860615462 (e6 1c51 26)
PRIME 432
    10.. .... = PRIME SAR Type: 2
    ..00 0001 = PRIME SAR Nseg: 1
    PRIME 432 data: 0000000600000000060000000006000000000208090c07...

-- 432 Data continues from previous segment.
```

Release request, MAC frame carrying RLRQ APDU:

```
Hex:
0000    00 40 29 05 00 00 e0 08 08 86 05 00 90 01 01 62
0010    00 2e ef e9 a7

PRIME
    00.. .... = PRIME PDU Unused: 0
    ..00 .... = PRIME PDU Header Type: GPDU (0)
    Generic MAC Header
        .... 0000  0... .... = PRIME Generic MAC Header Reserved: 0
        .1.. .... = PRIME Generic MAC Downlink: 1
        ..00 0000 = PRIME Generic MAC Header level: 0
```

```
        PRIME Generic MAC checksum: 41
    GPDU Header
        000. .... = PRIME PDU General Header reserved: 0
        ...0 .... = PRIME PDU General Header NAD: 0
        .... 01.. = PRIME PDU General Header Prio: 1
        .... ..0. = PRIME PDU General Header control: 0
        .... ...1  0000 0000 = PRIME PDU General Header LCID: 256
        PRIME PDU General Header SID: 0
        1110 0000  0000 10.. = PRIME PDU General Header LNID: 14338
        .... ..0. = PRIME PDU General Header padding: 0
        .... ...0  0000 1000 = PRIME PDU General Header length: 8
    GPDU ARQ
        1... .... = PRIME PDU ARQ M: 1
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..00 0110 = PRIME PDU ARQ PKTID: 6
        0... .... = PRIME PDU ARQ M: 0
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..00 0101 = PRIME PDU ARQ ACKID: 5
    GPDU Payload
    PRIME PDU CRC: 787474855 (2e ef e9 a7)
PRIME 432
    00.. .... = PRIME SAR Type: 0
    ..00 0000 = PRIME SAR Nseg: 0
    1... .... = PRIME 432 one bit: 1
    .00. .... = PRIME 432 Command: 0
    ...1 .... = PRIME 432 Command/Response: 1
    .... 0000 = PRIME 432 Qualifier: 0
    PRIME 432 Destination L_SAP: 1
    PRIME 432 Source L_SAP: 1
    PRIME 432 data: 6200

-- 432 Data explanation:

62 00                  // release-request
```

Release response, MAC frame carrying RLRE APDU:

```
Hex:
0000   00 00 ee 15 00 00 e0 08 08 85 07 00 90 01 01 63
0010   00 a0 9d 21 92

PRIME
    00.. .... = PRIME PDU Unused: 0
    ..00 .... = PRIME PDU Header Type: GPDU (0)
    Generic MAC Header
        .... 0000  0... .... = PRIME Generic MAC Header Reserved: 0
        .0.. .... = PRIME Generic MAC Downlink: 0
        ..00 0000 = PRIME Generic MAC Header level: 0
        PRIME Generic MAC checksum: 238
    GPDU Header
        000. .... = PRIME PDU General Header reserved: 0
        ...1 .... = PRIME PDU General Header NAD: 1
        .... 01.. = PRIME PDU General Header Prio: 1
        .... ..0. = PRIME PDU General Header control: 0
        .... ...1  0000 0000 = PRIME PDU General Header LCID: 256
        PRIME PDU General Header SID: 0
        1110 0000  0000 10.. = PRIME PDU General Header LNID: 14338
        .... ..0. = PRIME PDU General Header padding: 0
        .... ...0  0000 1000 = PRIME PDU General Header length: 8
    GPDU ARQ
        1... .... = PRIME PDU ARQ M: 1
        .0.. .... = PRIME PDU ARQ FLUSH: 0
```

```
        ..00 0101 = PRIME PDU ARQ PKTID: 5
        0... .... = PRIME PDU ARQ M: 0
        .0.. .... = PRIME PDU ARQ FLUSH: 0
        ..00 0111 = PRIME PDU ARQ ACKID: 7
    GPDU Payload
    PRIME PDU CRC: 2694652306 (a0 9d 21 92)
PRIME 432
    00.. .... = PRIME SAR Type: 0
    ..00 0000 = PRIME SAR Nseg: 0
    1... .... = PRIME 432 one bit: 1
    .00. .... = PRIME 432 Command: 0
    ...1 .... = PRIME 432 Command/Response: 1
    .... 0000 = PRIME 432 Qualifier: 0
    PRIME 432 Destination L_SAP: 1
    PRIME 432 Source L_SAP: 1
    PRIME 432 data: 6300

-- 432 Data explanation:

63 00 // release-response
```

Bibliography

STD0005 – *Internet Protocol*
Author: J. Postel
Date: September 1981
Also: RFC0791, RFC0792, RFC0919, RFC0922, RFC0950, RFC1112
Available from: http://www.ietf.org/rfc/rfc0791.txt

STD0006 – *User Datagram Protocol*
Author: J. Postel
Date: 28 August 1980
Also: RFC0768
Available from: http://www.ietf.org/rfc/rfc0768.txt

STD0007 – *Transmission Control Protocol*
Author: J. Postel
Date: September 1981
Also: RFC0793
Available from: http://www.ietf.org/rfc/rfc0793.txt

IPv4 TOS Byte and IPv6 Traffic Class Octet
Available from: http://www.iana.org/assignments/ipv4-tos-byte/ipv4-tos-byte.xml

This page deliberately left blank

British Standards Institution (BSI)

BSI is the national body responsible for preparing British Standards and other standards-related publications, information and services.

BSI is incorporated by Royal Charter. British Standards and other standardization products are published by BSI Standards Limited.

About us

We bring together business, industry, government, consumers, innovators and others to shape their combined experience and expertise into standards -based solutions.

The knowledge embodied in our standards has been carefully assembled in a dependable format and refined through our open consultation process. Organizations of all sizes and across all sectors choose standards to help them achieve their goals.

Information on standards

We can provide you with the knowledge that your organization needs to succeed. Find out more about British Standards by visiting our website at bsigroup.com/standards or contacting our Customer Services team or Knowledge Centre.

Buying standards

You can buy and download PDF versions of BSI publications, including British and adopted European and international standards, through our website at bsigroup.com/shop, where hard copies can also be purchased.

If you need international and foreign standards from other Standards Development Organizations, hard copies can be ordered from our Customer Services team.

Subscriptions

Our range of subscription services are designed to make using standards easier for you. For further information on our subscription products go to bsigroup.com/subscriptions.

With **British Standards Online (BSOL)** you'll have instant access to over 55,000 British and adopted European and international standards from your desktop. It's available 24/7 and is refreshed daily so you'll always be up to date.

You can keep in touch with standards developments and receive substantial discounts on the purchase price of standards, both in single copy and subscription format, by becoming a **BSI Subscribing Member**.

PLUS is an updating service exclusive to BSI Subscribing Members. You will automatically receive the latest hard copy of your standards when they're revised or replaced.

To find out more about becoming a BSI Subscribing Member and the benefits of membership, please visit bsigroup.com/shop.

With a **Multi-User Network Licence (MUNL)** you are able to host standards publications on your intranet. Licences can cover as few or as many users as you wish. With updates supplied as soon as they're available, you can be sure your documentation is current. For further information, email bsmusales@bsigroup.com.

BSI Group Headquarters

389 Chiswick High Road London W4 4AL UK

Revisions

Our British Standards and other publications are updated by amendment or revision.

We continually improve the quality of our products and services to benefit your business. If you find an inaccuracy or ambiguity within a British Standard or other BSI publication please inform the Knowledge Centre.

Copyright

Useful Contacts:

Customer Services
Tel: +44 845 086 9001
Email (orders): orders@bsigroup.com
Email (enquiries): cservices@bsigroup.com

Subscriptions
Tel: +44 845 086 9001
Email: subscriptions@bsigroup.com

Knowledge Centre
Tel: +44 20 8996 7004
Email: knowledgecentre@bsigroup.com

Copyright & Licensing
Tel: +44 20 8996 7070
Email: copyright@bsigroup.com

bsi.

...making excellence a habit.™