ETSI TS 102 744-3-5 V1.1.1 (2015-10)

Satellite Earth Stations and Systems (SES);
Family SL Satellite Radio Interface (Release 1);
Part 3: Control Plane and User Plane Specifications;
Sub-part 5: Adaptation Layer Interface

Reference
DTS/SES-00299-3-5

Keywords
3GPP, GPRS, GSM, GSO, interface, MSS, radio, satellite, TDM, TDMA, UMTS

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://ipr.etsi.org).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Satellite Earth Stations and Systems (SES).

The present document is part 3, sub-part 5 of a multi-part deliverable. Full details of the entire series can be found in ETSI TS 102 744-1-1 [i.4].

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Introduction

This multi-part deliverable (Release 1) defines a satellite radio interface that provides UMTS services to users of mobile terminals via geostationary (GEO) satellites in the frequency range 1 518,000 MHz to 1 559,000 MHz (downlink) and 1 626,500 MHz to 1 660,500 MHz and 1 668,000 MHz to 1 675,000 MHz (uplink).

1 Scope

The present document defines the Adaptation Layer (AL) peer-to-peer interface of the Family SL satellite radio interface between the Radio Network Controller (RNC) and the User Equipment (UE) used in the satellite network.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] ETSI TS 133 102: "Universal Mobile Telecommunications System (UMTS); 3G security; Security architecture (3GPP TS 33.102 Release 4)".

[2] ETSI TS 133 105: "Universal Mobile Telecommunications System (UMTS); Cryptographic algorithm requirements (3GPP TS 33.105 Release 4)".

[3] ETSI TS 125 331: "Universal Mobile Telecommunications System (UMTS); Radio Resource Control (RRC) protocol specification (3GPP TS 25.331 Release 4)".

[4] ETSI TS 102 744-1-4: "Satellite Earth Stations and Systems (SES); Family SL Satellite Radio Interface (Release 1); Part 1: General Specifications; Sub-part 4: Applicable External Specifications, Symbols and Abbreviations".

[5] ETSI TS 102 744-2-2: "Satellite Earth Stations and Systems (SES); Family SL Satellite Radio Interface (Release 1); Part 2: Physical Layer Specifications; Sub-part 2: Radio Transmission and Reception".

[6] ETSI TS 102 744-3-1: "Satellite Earth Stations and Systems (SES); Family SL Satellite Radio Interface (Release 1); Part 3: Control Plane and User Plane Specifications; Sub-part 1: Bearer Control Layer Interface".

[7] ETSI TS 102 744-3-4: "Satellite Earth Stations and Systems (SES); Family SL Satellite Radio Interface (Release 1); Part 3: Control Plane and User Plane Specifications; Sub-part 4: Bearer Connection Layer Operation".

[8] ETSI TS 102 744-3-6: "Satellite Earth Stations and Systems (SES); Family SL Satellite Radio Interface (Release 1); Part 3: Control Plane and User Plane Specifications; Sub-part 6: Adaptation Layer Operation".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] NMEA 0183 Interface Standard, Version 3.01, National Marine Electronics Association, January 2002.

[i.2] IETF RFC 2507 (1999): "IP Header Compression", M. Degermark, B. Nordgren, S. Pink.

[i.3] IETF RFC 3095 (2001): "Robust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed", C. Bormann, C. Burmeister, M. Degermark, H. Fukushima, H. Hannu, L-E. Jonsson, R. Hakenberg, T. Koren, K. Le, Z. Liu, A. Martensson, A. Miyazaki, K. Svanbro, T. Wiebke, T. Yoshimura, H. Zheng.

[i.4] ETSI TS 102 744-1-1: "Satellite Earth Stations and Systems (SES); Family SL Satellite Radio Interface (Release 1); Part 1: General Specifications; Sub-part 1: Services and Architectures".

3 Symbols and abbreviations

3.1 Symbols

For the purposes of the present document, the symbols given in ETSI TS 102 744-1-4 [4], clause 3 apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI TS 102 744-1-4 [4], clause 3 apply.

4 Adaptation Layer Interface

4.1 Radio Interface Layering

The satellite communication protocol is considered as a number of communication layers, as follows:

- Adaptation Layer (AL);
- Bearer Connection Layer (BCn); and
- Bearer Control Layer (BCt);
- Physical Layer (L1).

The satellite radio interface protocol stack is designed to seamlessly integrate with UMTS Non-Access Stratum entities, such as GPRS Mobility Management (GMM) and Mobility Management (MM), residing in the Core Network (CN) and in the upper layers of the User Equipment (UE).

The Adaptation Layer provides support to the UMTS Non-Access Stratum entities GMM and MM, and uses the services provided by the Bearer Connection Layer, as shown in Figure 4.1. The present document defines the Adaptation Layer peer-to-peer interface between the Radio Network Controller (RNC) and the UE, as shown in Figure 4.1.

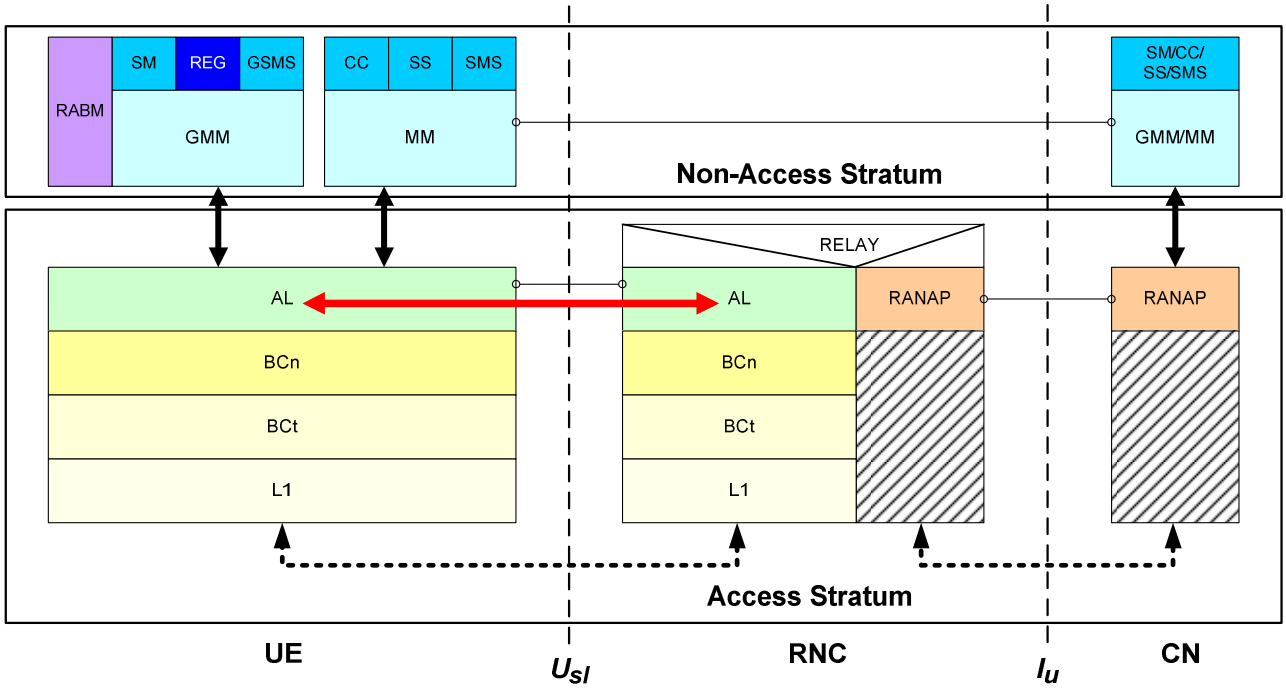


Figure 4.1: Control Plane Protocol Stack Layering with Adaptation Layer peer-to-peer interface indicated

4.2 Conventions used in the present document

4.2.1 Presentation

The following conventions are applied throughout the present document:

- In the ASN.1, variable names are always in lower case letters with hyphenation used to improve readability (e.g. ret-bct-pdu-header). Data Types in the ASN1. Always start with an upper case letter and may contain additional upper case letters to improve readability (e.g. ReturnBCtPDUHeader).
- In the explanatory text these variables are referred to in italics (e.g. *ret-bct-pdu-header)* while Data Types are shown in Helvetica typeface (e.g. BCnPDU).

The layout of the data structures defined in the ASN.1 is also shown in a graphical representation and gives examples of the usage of these structures. In general, the variable names are presented in the same way they are presented in the ASN.1, with the following exceptions:

- insufficient space does not allow the complete variable name to be presented and is therefore abbreviated;
- only one particular value can be assigned to a variable in the particular structure that is presented. In this case the variable is replaced by the appropriate numerical value;
- additional information may be added in brackets for explanatory reasons.

4.2.2 "Reserved" Fields and Values

Fields shown as Reserved BITSTRING (..) in the ASN.1 structures shall be set to zero by the sender and shall be ignored by the receiver.

Values not allocated in distinguished value lists shall not be used by the sender and shall be ignored by the receiver.

NOTE 1: Distinguished Value Lists of type Integer are being used instead of the ENUMERATED data type where the allocated number range is larger than the number of items to be enumerated.

NOTE 2: It should be noted that UEs may only support a lower RI-Version than the one supported by the RNC (see clause 6.1.2.2). In this case, it is likely that Broadcast SDUs/AVPs transmitted by the RNC contain values that are considered as "reserved" by those UEs.

4.2.3 Boolean Variables

BOOLEAN variables shall be encoded as follows:

```
TRUE    ::= 1
FALSE   ::= 0
```

4.2.4 ASN.1 Encoding Rules

The ASN.1 presentation provided in the present document for this interface specification is normative. The encoding rules used for this interface specification are provided in clause 4.3.4 of ETSI TS 102 744-3-1 [6].

5 Adaptation Layer

5.1 Overview

5.1.1 Responsibilities

The Adaptation Layer is responsible for the following:

- **Registration Management:** spot beam selection, system information handling, Non Access Stratum (NAS) system information notification, registration and deregistration (with the RNC), GPS position reporting and GPS position encryption.
- **Mobility Management Support:** providing RRC-like message transport and event notification services to GMM in NAS as well as integrity protection and ciphering control.
- **Radio Bearer Control:** handling signalling related to setup, modification, and release of radio bearers, configuring user plane protocol layers and entities and notifying NAS entities of resource assignments.

5.2 Adaptation Layer use of lower layer Service Access Points

To provide a seamless interface between the satellite network Access Stratum and the UMTS Non-Access Stratum, functional equivalents for a number of UTRAN Radio Resource Control (RRC) messages have been defined. Each RRC message is mapped to an equivalent message for the Satellite Radio Interface. Table 5.1 provides an overview of the relationships between RRC Messages and the equivalent satellite radio interface Common Signalling Messages (Unacknowledged Mode). Table 5.2 provides the same for the satellite radio interface UE Specific Signalling Messages (Acknowledged Mode).

Table 5.1: Mapping of RRC Messages to Satellite Radio Interface Common Signalling

RRC Message	Direction	Satellite Radio Interface Equivalent	See clause
PAGING TYPE 1	To UE	PagingType1	6.1.1
RRC CONNECTION REQUEST	From UE	Register	6.1.2
RRC CONNECTION SETUP	To UE	RegisterAck	6.1.3
RRC CONNECTION REJECT	To UE	RegisterRej	6.1.4
RRC CONNECTION RELEASE	To UE	DeregisterCommon	6.1.5

Table 5.2: Mapping of RRC Messages to Satellite Radio Interface UE Specific Signalling

RRC Message	Direction	Satellite Radio Interface Equivalent	See clause
RRC CONNECTION SETUP COMPLETE	From UE	RegisterComplete	6.2.10
RADIO BEARER SETUP	To UE	Establish	6.2.2
RADIO BEARER SETUP COMPLETE	From UE	EstablishAck	6.2.3
RADIO BEARER SETUP FAILURE	From UE	EstablishReject	6.2.11
RADIO BEARER RECONFIGURATION	To UE	Modify	6.2.6
RADIO BEARER RECONFIGURATION COMPLETE	From UE	ModifyAck	6.2.7
RADIO BEARER RECONFIGURATION FAILURE	From UE	ModifyReject	6.2.13
RADIO BEARER RELEASE	To UE	Release	6.2.4
RADIO BEARER RELEASE COMPLETE	From UE	ReleaseAck	6.2.5
RADIO BEARER RELEASE FAILURE	From UE	ReleaseReject	6.2.12
PAGING TYPE 2	To UE	PagingType2	6.2.14
INITIAL DIRECT TRANSFER	From UE	InitialDirectTransfer	6.2.15
UPLINK DIRECT TRANSFER	From UE	UplinkDirectTransfer	6.2.16
DOWNLINK DIRECT TRANSFER	To UE	DownlinkDirectTransfer	6.2.17
SECURITY MODE COMPLETE	From UE	SecurityModeComplete	6.2.19
SECURITY MODE FAILURE	From UE	SecurityModeFailure	6.2.20
SECURITY MODE COMMAND	To UE	SecurityModeCommand	6.2.18
SIGNALLING CONNECTION RELEASE REQUEST	From UE	SignallingConnectionReleaseReq	6.2.21
SIGNALLING CONNECTION RELEASE	To UE	SignallingConnectionRelease	6.2.22
RRC CONNECTION RELEASE	To UE	Deregister	6.2.27
RRC CONNECTION RELEASE COMPLETE	From UE	DeregisterAck	6.2.28

In addition to the above, a number of UE Specific Signalling Messages do not have a functional equivalent in UTRAN RRC and are required for the satellite radio interface. These are summarized in Table 5.3.

Table 5.3: UE Specific Signalling Messages without RRC equivalent

Satellite Radio Interface UE Specific Signalling Message	Direction	See clause
Handover	To UE	6.2.8
HandoverAck	From UE	6.2.9
UEPositionRequest	To UE	6.2.23
UEPositionResponse	From UE	6.2.24
RegModeUpdate	To UE	6.2.25
SystemInformation	To UE	6.2.26
HandoverRequest	To RNC	6.2.29

6 Adaptation Layer Protocol Data Units

6.0 General

The following clauses define the format of the Protocol Data Units which are used by the Adaptation Layer to signal its peer (AL-PDUs). Clause 6.1 specifies those AL-PDUs which are sent through Common Signalling (AL-ComPDUs), while clause 6.2 specifies those AL-PDUs which are sent on a UE Specific Signalling Connection (AL-SigPDUs).

6.1 Common Signalling Protocol Data Units (ALComPDUs)

6.1.0 General

Common Signalling PDUs are carried within a Common Protocol Data Unit (PDU) in the Bearer Control sub-layer, where the type of the Common Signalling PDU is carried within the ComSigType field within a Bearer Control PDU (see ETSI TS 102 744-3-1 [6]).

```
ALComPDU ::=
    CHOICE {
        empty-common-sig
            NULL,
        paging-type-1
            PagingType1
        register
            Register,
        register-ack
            RegisterAck,
        register-rej
            RegisterRej
        deregister-common
            DeregisterCommon
        }
```

6.1.1 PagingType1

6.1.1.0 General

The PagingType1 Signalling PDU is used by the RNC when the UE is not registered to indicate that the UE should initiate the Registration process. The Adaptation Layer shall inform the Non Access Stratum of the event. The PDU is defined as below, with format shown in Figure 6.1.

```
PagingType1 ::=
    SEQUENCE {
        cn-domain-identity
            CNDomainIdentity,
        paging-cause
            PagingCause
        }
```

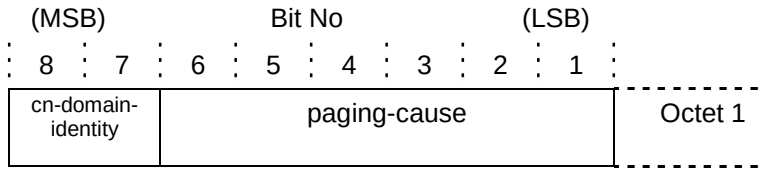


Figure 6.1: PagingType1 Common Signalling PDU

6.1.1.1 CNDomainIdentity

This parameter identifies the Core Network Domain, which originated the paging request. The parameter definition is the same as for the CNDomainIdentity Information Element (IE) specified in [3], clauses 10.3.1.1 and 11.3 as follows:

```
CNDomainIdentity ::=
    INTEGER {
        cs-domain (0),
        ps-domain (1),
        bm-domain (2),
    } (0..3)
```

6.1.1.2 PagingCause

This parameter is also sent from the Core Network and hence follows the definition of the PagingCause IE in [3], clauses 10.3.3.22 and 11.3 as follows:

```
PagingCause ::=
    INTEGER {
        terminatingConversationalCall (0),
        terminatingStreamingCall (1),
        terminatingInteractiveCall (2),
        terminatingBackgroundCall (3),
        terminatingHighPrioritySignalling (4),
        terminatingLowPrioritySignalling (5),
        terminatingCauseUnknown (6)
    } (0..63)
```

6.1.2 Register

6.1.2.0 General

The Register Common Signalling PDU is used by the UE to request the initiation of the registration process. Addressing is performed by the Bearer Control Layer using the Initial UE Identity. The Register Common Signalling PDU is defined as below, with structure as shown in Figure 6.2.

```
Register ::=
    SEQUENCE {
        reg-ref
            RegistrationReference,
        ai-version
            AIVersion,
        cn-domain-identity
            CNDomainIdentity,
        registration-cause
            RegistrationCause,
        reserved
            BITSTRING (SIZE (3)),
        ue-class
            UEClass
    }
```

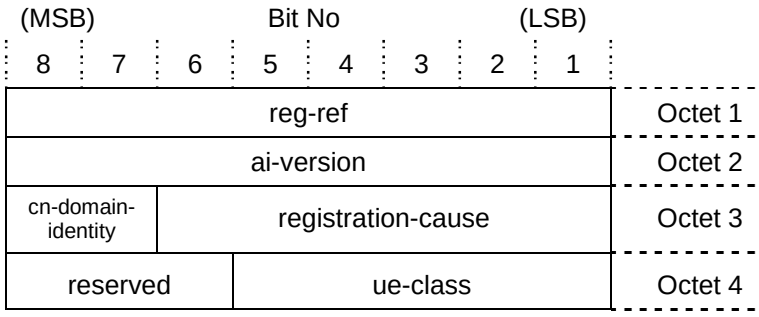


Figure 6.2: Register Common Signalling PDU

The parameter *cn-domain-identity* is defined in clause 6.1.1.1.

6.1.2.1 RegistrationReference

The *reg-ref* field (Registration-Reference) carries a sequence number generated by the UE used to synchronize the forward and return registration information. This information is used at the RNC to determine whether this is a repeat registration request or a new registration request.

```
RegistrationReference ::=
    INTEGER (0..255)
```

When used in the RNC-UE direction, the Registration-Reference field either carries the value specified by the UE, or it may contain a NULL (0) value - indicating that the UE should obey the instruction regardless of the currently stored Registration Reference value.

6.1.2.2 RIVersion

The RI-Version field contains the Radio Interface (RI) version number to which the software within the UE was designed to operate. Note that the RI-Version number is used to refer to all layers within the Access Stratum. The interpretation of the RI-Version number is shown in Table 6.1.

```
RIVersion ::=
    INTEGER {
        syst-initial-release (129),
        syst-extension-1 (130)
        Syst-extension-2 (131)
    } (0..255)
```

Table 6.1: RI-Version Values

RI-Version	Interpretation
0x00 - 0x80	Reserved
0x81	System Initial Release
0x82	System Extension 1
0x83	System Extension 2
0x84 - 0xFF	Reserved

The UE shall support all interface and behavioural requirements corresponding to the declared RI-Version except those requirements that are optional or not applicable to the UE Class.

6.1.2.3 RegistrationCause

This parameter specifies the reason for the registration message. Currently, only normal registration and registration due to emergency call are defined but other values may be allocated in the future.

```
RegistrationCause ::=
    INTEGER {
        normal-registration (0),
        emergency (1)
    } (0..63)
```

6.1.2.4 UEClass

The *ue-class* field identifies the physical capabilities of the UE (i.e. transmitter EIRP, receiver G/T). The UEClass data structure is defined as follows:

```
UEClass ::=
    INTEGER {
        land-A3 (1),
        land-A4 (2),
        land-A5 (3),
        aeronautical-lga (4),
        maritime-lga (5),
        aeronautical-high-gain (6),
        aeronautical-intermediate-gain (7),
        maritime-high-gain (8),
        maritime-intermediate-gain (9),
        land-mobile-high-gain (10),
        land-mobile-intermediate-gain (11),
        land-mobile-lga (12),
        reserved-class-13 (13),
        maritime-intermediate-gain-restricted (14),
        aeronautical-low-gain-restricted (15)
    } (0..31)  -- values > 24 not used
```

The characteristics of each of the UE classes are defined in ETSI TS 102 744-2-2 [5].

6.1.3 RegisterAck

6.1.3.0 General

The RegisterAck Common Signalling PDU is used by the RNC to confirm the completion of the registration process. Addressing is performed by the Bearer Control Layer and is based upon the Initial UE Identity.

The Register-Ack Common Signalling PDU is defined as below, with structure as shown in Figure 6.3.

```
RegisterAck ::=
    SEQUENCE {
        reg-ref
            RegistrationReference,
        bearer-conn-id
            BcnID,
        bearer-conn-type
            BcnType,
        num-param
            NumParam,
        bcn-param-list
            BcnParamList,
```

```
        ctrl-flags
            CtrlFlags,
        reserved
            BIT STRING (SIZE(4)),
        bct-type
            BctType,
        bct-id
            BctID,
        bct-epdu
            BCtEPDU OPTIONAL,
        reserved2
            BIT STRING (SIZE(4)),
        reg-mode
            RegistrationMode
    }
```

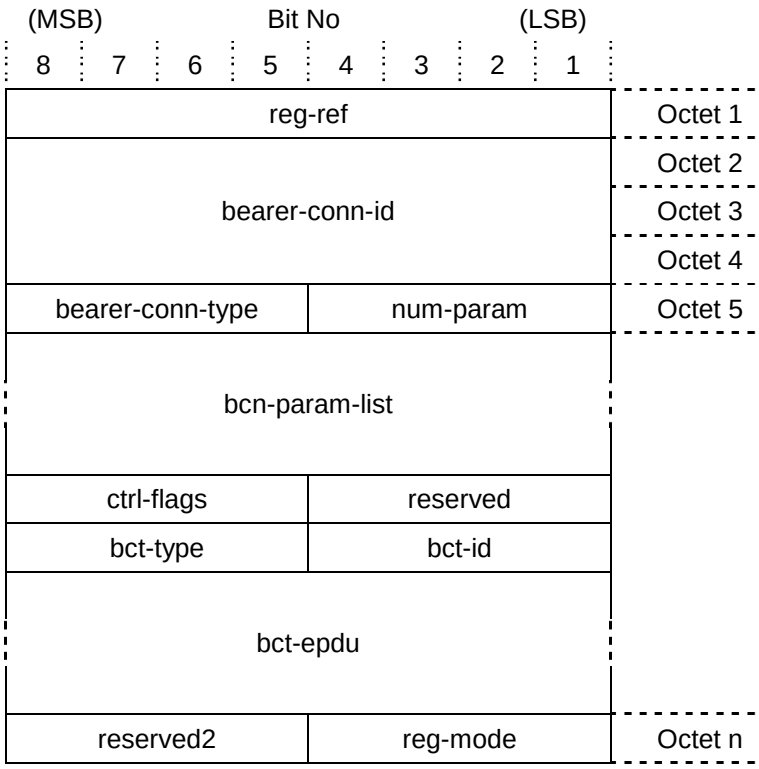


Figure 6.3: Register-Ack Common Signalling PDU

The parameter *reg-ref* is defined in clause 6.1.2.1.

6.1.3.1 BcnID

The parameter *bearer-conn-id* is used to uniquely identify a connection within the context of a satellite.

```
BcnID ::=
    INTEGER (0.. 16777215);
```

Value 0x000000 is reserved. Values in the range 0xFFFFF0 to 0xFFFFFF are reserved for Cell Broadcast connections.

6.1.3.2 BcnType

The *bearer-conn-type* field defines the properties of the Bearer Connection and in the RegisterAck message is always set to 2 (Acknowledged Mode, In Sequence Delivery) since the RegisterAck message causes the UE specific signalling connection to be established. For a full specification of BcnType see clause 6.2.2.1.

6.1.3.3 NumParam

This parameter is used to define the number of bearer connection parameters being transferred as Bearer Connection AVPs in the *bcn-param-list*.

```
NumParam ::=
    INTEGER (0..15)
```

6.1.3.4 BcnParamList

The *bcn-param-list* carries a set of parameters for the connection being initiated as well as allowing for the transport of Adaptation Layer AVPs. The list of parameters is dependent upon the connection being initiated. The parameters are transported in BCnAVPs which are described in detail in clause 6.4 while the Adaptation Layer AVPs are specified in clause 6.3. The AdaptationLayerAVPListLengthAVP is defined in clause 6.4.7.

```
BcnParamList ::=
    SEQUENCE {
        bcn-avp-list
            SEQUENCE SIZE(0..15) OF BCnAVP,
        optional-avp-list
            SEQUENCE {
                al-avp-list-len-avp
                    AdaptationLayerAVPListLengthAVP,
                al-avp-list
                    SEQUENCE OF AdaptationLayerAVP
            } OPTIONAL
    }
```

6.1.3.5 CtrlFlags

This parameter contains a set of control flags which are defined as follows:

```
CtrlFlags ::=
    SEQUENCE {
        bct-epdu-follows
            BOOLEAN,
        reserved
            BIT STRING (SIZE(3))
    }
```

The *bct-epdu-follows* flag indicates whether any Bearer Control Embedded PDUs are present.

6.1.3.6 BctType

The *bct-type* field contains the bearer control type to which the UE-specific signalling connection is to be attached. If the *bct-type* field is different to the current Bearer Control process type, then a handover to a new Bearer Control is taking place. Values are defined as follows:

```
BctType ::=
    INTEGER {
        current-bearer-control (0),
        m4-bearer-control (1),
        ai-bearer-control (2),
        ldr-bearer-control (3)
    } (0..15)
```

The value of 0 is used to indicate that the current bearer control is to be used. Any other value indicates that a new bearer control is being used at the RNC, and all existing parameter information shall be discarded within the UE - defaults are to be used unless over-ridden.

6.1.3.7 BctID

The *bct-id* field identifies the specific bearer control process to which the UE-specific signalling connection is to be attached. If this is different to the current *bct-id* transferred in the Bulletin Board, then a handover to a new bearer control is taking place.

```
BctID ::=
    INTEGER (0..15)
```

6.1.3.8 BCtEPDU

This parameter contains a sequence of Bearer Control Embedded Signalling PDUs which are formulated within the Bearer Control Layer at the RNC and transported with an Adaptation Layer Signalling PDU.

The BCtEPDU contains Bearer Control specific configuration information which is to be passed to the Bearer Control Layer at the UE during the attachment process (note that attachment to a Bearer Control takes place both at establishment and at handover (which may take place either as a result of a load balancing exercise at the RNC or during a connection modify)). The format is transparent to the Adaptation Layer and is defined in ETSI TS 102 744-3-1 [6].

From the Adaptation Layer perspective the following applies:

```
BCtEPDU::=
    OCTET STRING (SIZE (0..255))
```

6.1.3.9 RegistrationMode

The *reg-mode* parameter is used by the RNC to control the behaviour of the UE and restrict the actions that the UE may take. For a detailed description of the behaviour related to *reg-mode* see ETSI TS 102 744-3-6 [8].
The RegistrationMode is defined as follows:

```
RegistrationMode ::=
    INTERGER {
        conditional-registration (0),
        full-registration (3)
    } (0..15)
```

6.1.4 RegisterRej

6.1.4.0 General

The RegisterRej Common Signalling PDU is used by the RNC to indicate the failure of the registration process. Addressing is performed by the Bearer Control Layer and is based upon the Initial UE Identity.

The Register-Rej Common Signalling PDU is defined as below, with structure as shown in Figure 6.4.

```
RegisterRej ::=
    SEQUENCE {
        reg-ref
            RegistrationReference,
        cause
            CHOICE {
                rej-cause
                    RejectionCause,
                prot-err-cause
                    ProtocolErrorCause
            }
    }
```

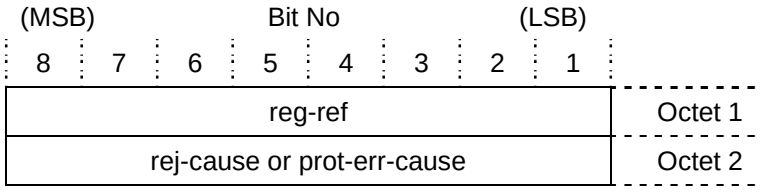


Figure 6.4: Register-Rej Common Signalling PDU

The parameter *reg-ref* is explained in clause 6.1.2.1.

6.1.4.1 RejectionCause and ProtocolErrorCause

The *cause* parameter may either contain a rejection cause or a protocol error cause value. The *rej-cause* parameter is of type RejectionCause which is defined as follows:

```
RejectionCause ::=
    INTEGER {
        rnc-failure (1)
        congestion (2),
        unsupported-ai-version(3),
        unsupported-ue-class(4),
        usim-required(5)
    } (0..255)  -- valid range 0-127
                -- 128-255 reserved for ProtocolErrorCause
```

The *prot-err-cause* parameter is of type ProtocolErrorCause which is defined as follows:

```
ProtocolErrorCause ::=
    INTEGER {
        asn1-violation-or-encoding-error (128),
        message-type-nonexistent (129),
        message-not-compatible-with-receiver-state (130),
        ie-value-not-comprehended (131),
        ie-missing (132)
    } (0..255)  -- ProtocolErrorCause values in range 128-255
```

6.1.5 DeregisterCommon

6.1.5.0 General

The DeregisterCommon Common Signalling PDU is broadcast by the RNC to deregister all Ues on a particular bearer. To deregister individual Ues, the Deregister message is sent as an AL-Sig-PDU (see clause 6.2.27).

The DeregisterCommon Common Signalling PDU is defined as below, with structure as shown in Figure 6.5.

```
DeregisterCommon ::=
    SEQUENCE {
        reg-ref
            RegistrationReference,
        cause
            CHOICE {
                deregistration-cause
                    DeregistrationCause,
                prot-err-cause
                    ProtocolErrorCause
            }
    }
```

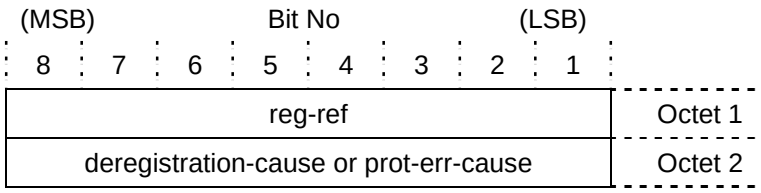


Figure 6.5: DeregisterCommon Common Signalling PDU

The parameter *reg-ref* (see clause 6.1.2.1) shall be set to 0x00.

6.1.5.1 DeregistrationCause and ProtocolErrorCause

The *cause* parameter may either contain a deregistration cause or a protocol error cause value. The *deregistration-cause* parameter in the DeregisterCommon message is of type DeregistrationCause and defined as follows:

```
DeregistrationCause ::=
    INTEGER {
        register-complete-not-received (1),
        service-area-barred (2),
        position-required (3),
        cn-reset (4),
        ue-inactivity (5),
        position-response-not-received (6),
        position-age-exceeds-maximum (7),
        decryption-error (8),
        user-specified-position-not-permitted (9),
        rnc-operator-initiated-deregistration (10),
        number-of-tracked-satellites-below-minimum (11),
        lease-group-not-available (12),
        lease-mode-handover-failed (13),
        radio-failure(14),
        unsupported-ue-class-subclass (15),
        elevation-too-low (16),
        protocol-failure (17),
        invalid-ue-capabilities (18)
    } (0..255)  -- valid range 0-127
                -- 128-255 reserved for ProtocolErrorCause
```

The *prot-err-cause* parameter is specified in clause 6.1.4.1.

6.2 UE-Specific Signalling Protocol Data Units (ALSignallingPDUs)

6.2.0 General

UE-Specific Signalling Protocol Data Units (ALSignallingPDUs) are encapsulated in one or a sequence of BCn-PDUs and are transported over the radio interface in Acknowledged Mode.

6.2.1 ALSignallingPDUStructure

6.2.1.0 General

The general structure of the ALSignallingPDU is as shown below in Figure 6.6 (without integrity protection) and Figure 6.7 (with integrity protection):

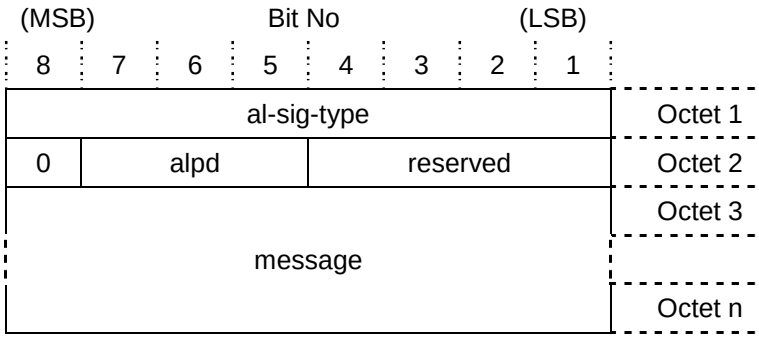


Figure 6.6: ALSignallingPDU Structure (without Integrity Protection)

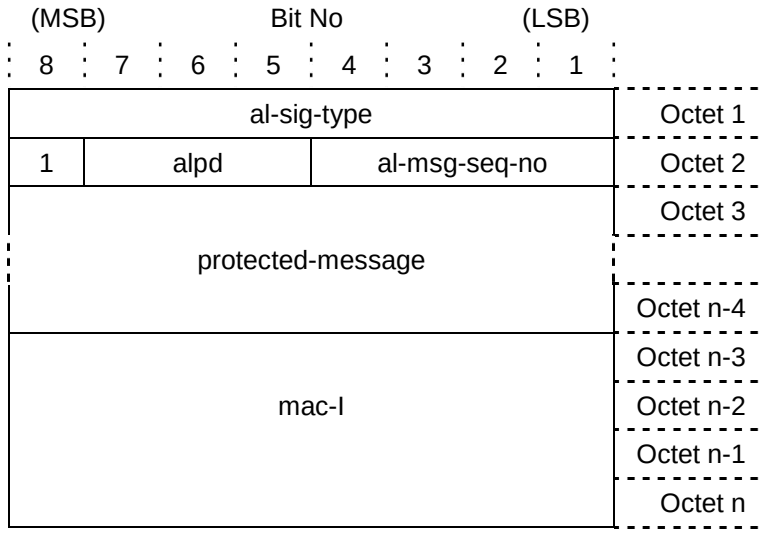


Figure 6.7: ALSignallingPDU Structure (with Integrity Protection)

All UE specific signalling in the satellite network (with the exception of the RegisterComplete message) may be integrity protected as specified in ETSI TS 102 744-3-6 [8] and in [1] and [2]. The ASN.1 definition of this structure is shown below:

```
ALSignallingPDU::=
    SEQUENCE {
        al-sig-type
            ALSignalType,
        al-sig-pdu
            CHOICE {
            -- as appropriate to value of al-signal-type
                empty-ms-sig    NULL,
                establish       INTEGRITY-OPT {Establish},
                establish-ack   INTEGRITY-OPT {EstablishAck},
                release         INTEGRITY-OPT {Release},
                release-ack     INTEGRITY-OPT {ReleaseAck},
                modify          INTEGRITY-OPT {Modify},
                modify-ack      INTEGRITY-OPT {ModifyAck},
                handover        INTEGRITY-OPT {Handover},
                handover-ack    INTEGRITY-OPT {HandoverAck},
                register-complete
                        INTEGRITY-OPT {RegisterComplete},
                establish-reject
                        INTEGRITY-OPT {EstablishReject},
                release-reject
                        INTEGRITY-OPT {ReleaseReject},
                modify-reject   INTEGRITY-OPT {ModifyReject},
                paging-type-2   INTEGRITY-OPT {PagingType2},
                initial-direct-transfer
                        INTEGRITY-OPT {InitialDirectTransfer},
                uplink-direct-transfer
                        INTEGRITY-OPT {UplinkDirectTransfer},
                downlink-direct-transfer
                        INTEGRITY-OPT {DownlinkDirectTransfer},
                security-mode-command
                        INTEGRITY-OPT {SecurityModeCommand},
                security-mode-complete
                        INTEGRITY-OPT {SecurityModeComplete},
                security-mode-failure
                        INTEGRITY-OPT {SecurityModeFailure},
                signalling-connection-release-req
                        INTEGRITY-OPT
                            {SignallingConnectionReleaseReq},
                signalling-connection-release
                        INTEGRITY-OPT
                            {SignallingConnectionRelease},
                ue-position-request
                        INTEGRITY-OPT {UEPositionRequest},
                ue-position-response
                        INTEGRITY-OPT {UEPositionResponse},
                reg-mode-update
```

```
                    INTEGRITY-OPT {RegModeUpdate},
                system-information
                    INTEGRITY-OPT {SystemInformation},
                deregister
                    INTEGRITY-OPT {Deregister},
                deregister-ack
                    INTEGRITY-OPT {DeregisterAck},
                handover-request
                    INTEGRITY-OPT {HandoverRequest}
            }
    }
```

The INTEGRITY-OPT parameterized type definition (ASN.1 1997) is defined as follows:

```
INTEGRITY-OPT {Message} ::=
    CHOICE {
        not-protected
            SEQUENCE {
                ic
                    IntegrityCheckIncluded,
                    -- {encode as FALSE}
                alpd
                    ALProtocolDiscriminator,
                reserved
                    BIT STRING (SIZE(4)),
                message
                    Message
            },
        protected
            SEQUENCE {
                ic
                    IntegrityCheckIncluded,
                    -- {encode as TRUE}
                alpd
                    ALProtocolDiscriminator,
                al-msg-seq-no
                    ALMsgSeqNumber,
                protected-message
                    Message,
                mac-I
                    MACIntegrity
            }
    }
```

6.2.1.1 ALSignalType

This parameter defines the Adaptation Layer Signalling Protocol Data Unit type. For the satellite network, the value for the PDU type is constrained to the range (128..255).

```
ALSignalType ::=
    INTEGER {
        empty-al-sig (128),
        establish (129),
        establish-ack (130),
        release (131),
        release-ack (132),
        modify (133),
        modify-ack (134),
        handover (135),
        handover-ack (136),
        register-complete (137),
        establish-rej (138),
        release-rej (139),
        modify-rej (140),
        paging-type-2 (141),
        initial-direct-transfer (142),
        uplink-direct-transfer (143),
        downlink-direct-transfer (144),
        security-mode-command (145),
        security-mode-complete (146),
        security-mode-failure (147),
        signalling-connection-release-req (148),
        signalling-connection-release (149),
        ue-position-request (150),
        ue-position-response (151),
        reg-mode-update (152),
```

```
        system-information (153),
        deregister (154),
        deregister-ack (155),
        handover-request (156)
    } (128..255)
```

6.2.1.2 IntegrityCheckIncluded

The value of the BOOLEAN integrity-check-included (*ic*) flag determines whether integrity protection information is included within the PDU.

```
IntegrityCheckIncluded ::=
    BOOLEAN
```

6.2.1.3 ALProtocolDiscriminator

The *alpd* parameter facilitates the routing of messages in the Adaptation Layer and is of type ALProtocolDiscriminator which is defined as follows:

```
ALProtocolDiscriminator ::=
    INTEGER {
        regm (1),
        gmmh (2),
        mmh (3),
        rbc-ps (4),
        rbc-cs (5)
    } (0..7)
```

See ETSI TS 102 744-3-6 [8] for a detailed description of the usage of this parameter.

6.2.1.4 ALMsgSeqNumber

The parameter *al-msg-seq-no* is incremented each time an integrity protected ALSignallingPDU is sent. It is used as an input parameter to the integrity protection algorithm and is of type ALMsgSeqNumber as defined below:

```
ALMsgSeqNumber ::=
    INTEGER (0..15)
```

See ETSI TS 102 744-3-6 [8] for a detailed description of the usage of this parameter.

6.2.1.5 Message

The parameter `message` represents individual *al-sig-pdu* messages listed in the ALSignallingPDU structure. The data structures for *al-sig-pdu* messages are defined in clauses 6.2.2 to 6.2.28.

6.2.1.6 MACIntegrity

The parameter *mac-I* carries the Message Authentication Code for Integrity as specified in [1] within an integrity protected ALSignallingPDU is sent and is of type MACIntegrity as defined below:

```
MACIntegrity ::=
    BIT STRING (SIZE (32))
```

6.2.2 Establish

6.2.2.0 General

This message is used by the RNC to establish a new Bearer Connection (Radio Bearer) in the User Plane. The PDU is defined as below, with structure as shown in Figure 6.8.

```
Establish ::=
    SEQUENCE {
        bearer-conn-id
            BcnID,
        bearer-conn-type
            BcnType,
        num-param
            NumParam,
        bcn-param-list
```

```
        BcnParamList,
    ctrl-flags
        CtrlFlags,
    trans-id
        TransactionID,
    bct-type
        BctType,
    bct-id
        BctID,
    bct-epdu
        BCtEPDU OPTIONAL
}
```

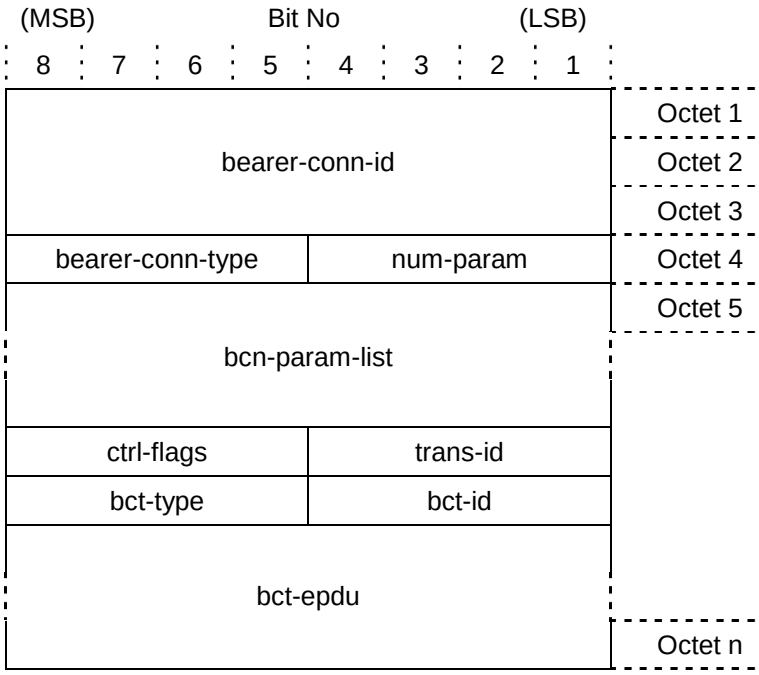


Figure 6.8: Establish Signalling PDU

The parameters *bearer-conn-type* and *trans-id* are defined below. For a definition of all other parameters see clause 6.1.3.1 to clause 6.1.3.9.

6.2.2.1 BcnType

The *bearer-conn-type* field defines the properties of the Bearer Connection and is specified as follows:

```
BcnType ::=
    INTEGER {
        tm-bidir-no-err (0),
        tm-bidir-err (1),
        am-bidir-in-seq (2),
        am-bidir-out-seq (3),
        um-bidir (4),
        um-unidir (6)
    }(0..15)
```

Values for BcnType are interpreted as shown in Table 6.2.

Table 6.2: Interpretation of BcnType Values

Value	Bearer Connection Mode	Qualifier
0	Transparent Mode	bi-directional - no delivery of erroneous PDUs
1	Transparent Mode	bi-directional - delivery of erroneous PDUs
2	Acknowledged Mode	bi-directional - in-sequence delivery of PDUs only
3	Acknowledged Mode	bi-directional - out-of-sequence delivery of PDUs permitted
4	Unacknowledged Mode	bi-directional
6	Unacknowledged Mode	to UE only

NOTE: Unacknowledged Mode does not support out-of-sequence delivery or delivery of erroneous PDUs, hence no such qualifier needs to be specified.

A detailed specification of the behaviour of the different Bearer Connection Modes can be found in ETSI TS 102 744-3-4 [7].

6.2.2.2 TransactionID

This parameter is used to uniquely associate a response or acknowledgement with a request, and allows multiple requests to be in transit at any one time.

```
TransactionID ::=
    INTEGER (0..15)
```

6.2.3 EstablishAck

6.2.3.0 General

This message is used by the UE to acknowledge the establishment of new Bearer Connections (Radio Bearer) in the User Plane. The PDU is defined as below, with structure as shown in Figure 6.9.

```
EstablishAck ::=
    SEQUENCE {
        bearer-conn-id
            BcnID,
        ctrl-flags
            CtrlFlags,
        trans-id
            TransactionID,
        adaptation-layer-avp-list
            AdaptationLayerAVPList OPTIONAL
    }
```

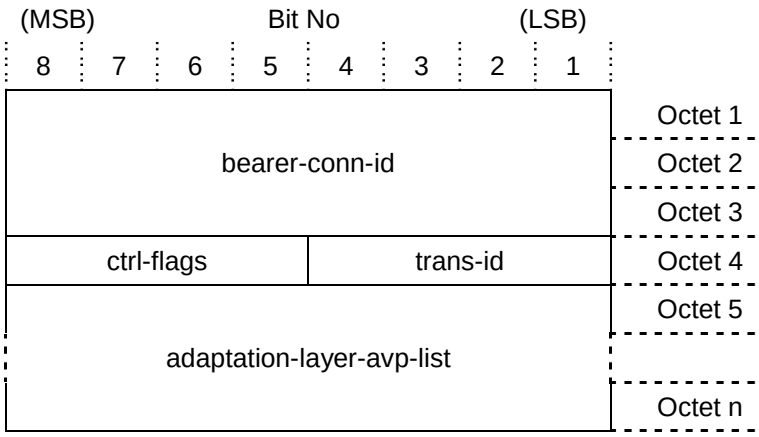


Figure 6.9: EstablishAck Signalling PDU

For a definition of *bearer-conn-id* see clause 6.1.3.1. The parameters *ctrl-flags* and *trans-id* are defined in clauses 6.1.3.5 and 6.2.2.2 respectively.

6.2.3.1 AdaptationLayerAVPList

The parameter *adaptation-layer-avp-list* is of type AdaptationLayerAVPList which is defined as follows:

```
AdaptationLayerAVPList ::=
    SEQUENCE {
        adaptation-layer-avp
            AdaptationLayerAVP
    }
```

The data type AdaptationLayerAVP is defined in clause 6.3.

6.2.4 Release

6.2.4.0 General

This message is used by the RNC to release a Bearer Connection (Radio Bearer) and associated Bearer Connection and Bearer Control Layer resources. The PDU is defined as below, with structure as shown in Figure 6.10.

```
Release ::=
    SEQUENCE {
        bearer-conn-id
            BcnID,
        ctrl-flags
            CtrlFlags,
        trans-id
            TransactionID,
        rel-cause
            ReleaseCause,
        additional-info
            SEQUENCE {
                bct-type
                    BctType,
                bct-id
                    BctID,
                bct-epdu
                    BCtEPDU
            } OPTIONAL
    }
```

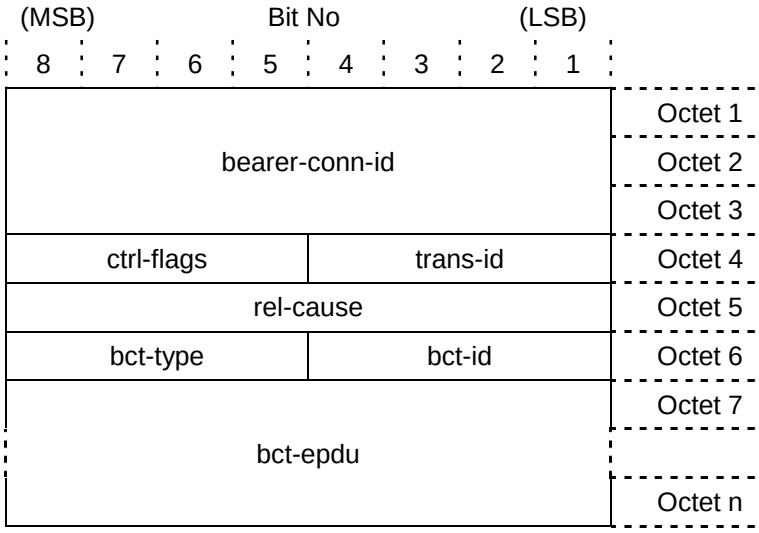


Figure 6.10: Release Signalling PDU

The parameters *ctrl-flags* and *trans-id* are defined in clauses 6.1.3.5 and 6.2.2.2 respectively. For a definition of all other parameters except *rel-cause* see clause 6.1.3.1 to clause 6.1.3.9.

6.2.4.1 ReleaseCause

This parameter is used to specify a reason for sending the Release message.

```
ReleaseCause ::=
    INTEGER {
        normal-release-cn-initiated (0),
        rnc-security-failure (1)
    } (0..255)
```

6.2.5 ReleaseAck

This message is used by the UE to acknowledge the release of a Bearer Connection (Radio Bearer) and associated resources. The PDU is defined as below, with structure as shown in Figure 6.11.

```
ReleaseAck ::=
    SEQUENCE {
        bearer-conn-id
            BcnID,
        ctrl-flags
            CtrlFlags,
        trans-id
            TransactionID
    }
```

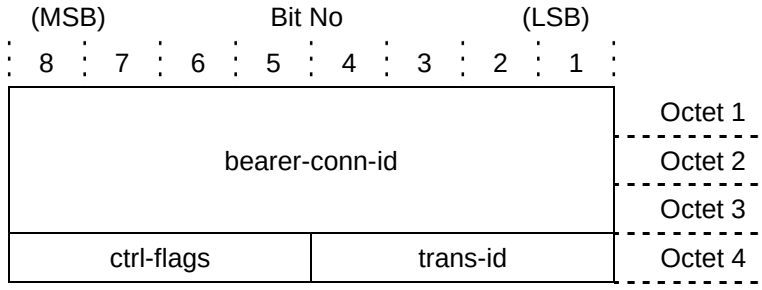


Figure 6.11: ReleaseAck Signalling PDU

The parameter *bearer-conn-id* is defined in clause 6.1.3.1 while the parameters *ctrl-flags* and *trans-id* are defined in clauses 6.1.3.5 and 6.2.2.2 respectively.

6.2.6 Modify

This message is used by the RNC to modify the Quality of Service attributes of a particular Bearer Connection. The PDU is defined as below, with structure as shown in Figure 6.12.

```
Modify ::=
    SEQUENCE {
        bearer-conn-id
            BcnID,
        bearer-conn-type
            BcnType,
        numparam
            NumParam,
        bcn-param-list
            BcnParamList,
        ctrl-flags
            CtrlFlags,
        trans-id
            TransactionID,
        bct-type
            BctType,
        bct-id
            BctID,
        bct-epdu
            BCtEPDU OPTIONAL
    }
```

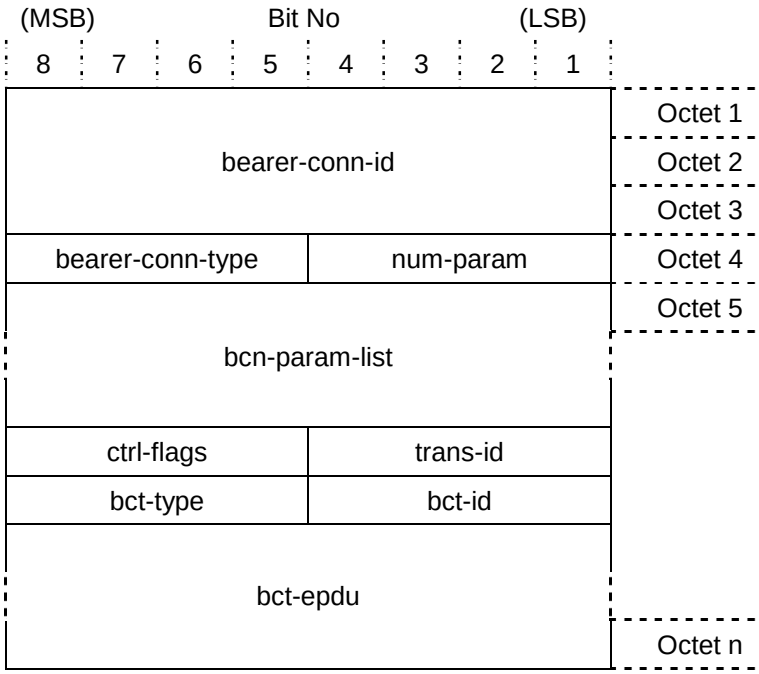


Figure 6.12: Modify Signalling PDU

The parameter *trans-id* is defined in clause 6.2.2.2. For a definition of all other parameters see clause 6.1.3.1 to clause 6.1.3.9.

6.2.7 ModifyAck

This message is used by the UE to acknowledge the modification the Quality of Service attributes of a particular Bearer Connection. The PDU is defined as below, with structure as shown in Figure 6.13.

```
ModifyAck ::=
    SEQUENCE {
        bearer-conn-id
            BcnID,
        ctrl-flags
            CtrlFlags,
        trans-id
            TransactionID,
        adaptation-layer-avp-list
            AdaptationLayerAVPList OPTIONAL
    }
```

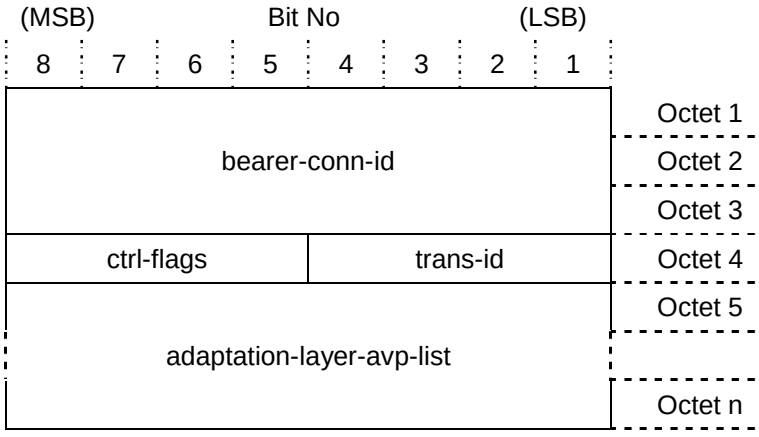


Figure 6.13: Modify-Ack Signalling PDU

The parameter *bearer-conn-id* is defined in clause 6.1.3.1. The parameters *ctrl-flags* and *trans-id* are defined in clauses 6.1.3.5 and 6.2.2.2 respectively and *adaptation-layer-avp-list* is defined in clause 6.2.3.1.

6.2.8 Handover

This message is used by the RNC to hand over a particular Bearer Connection between Bearer Control processes or to alter the characteristics of an existing Bearer Control process or addressing mode to be used by the UE for operation on the current Bearer Control. The PDU is defined as below, with structure as shown in Figure 6.14.

```
Handover ::=
    SEQUENCE {
        ctrl-flags
            CtrlFlags,
        trans-id
            TransactionID,
        bct-type
            BctType,
        bct-id
            BctID,
        bct-epdu
            BCtEPDU
    }
```

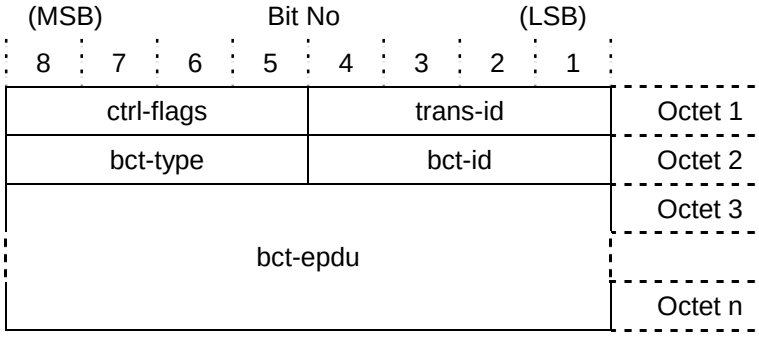


Figure 6.14: Handover Signalling PDU

The parameter *trans-id* is defined in clause 6.2.2.2 For a definition of all parameters see clause 6.1.3.1 to clause 6.1.3.9.

6.2.9 HandoverAck

This message is used by the UE to Acknowledge the Handover of a Bearer Connection between Bearer Controls or alteration in the characteristics of a Bearer Control. The PDU is defined as below, with structure as shown in Figure 6.15.

```
HandoverAck ::=
    SEQUENCE {
        ctrl-flags
            CtrlFlags,
        trans-id
            TransactionID
    }
```

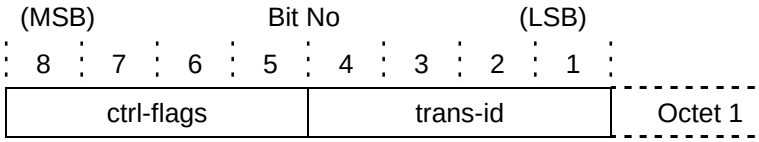


Figure 6.15: Handover-Ack Signalling PDU

For a description of the parameters *ctrl-flags* and *trans-id* see clauses 6.1.3.5 and 6.2.2.2 respectively.

6.2.10 RegisterComplete

6.2.10.0 General

This message is used by the UE to provide the RNC with the UMTS security parameter *start* for each Core Network service domain and UE capability information. The PDU is defined as below, with structure as shown in Figure 6.16.

```
RegisterComplete ::=
    SEQUENCE {
        reg-ref
            RegistrationReference,
        start-list
            SEQUENCE (SIZE (1..maxCNDomain)) OF SEQUENCE {
                    ch
                        ChainIndicator,
                    reserved
                        BIT STRING (SIZE (1)),
                    cn-domain-identity
                        CNDomainIdentity,
                    start-value
                        StartValue
            },
        ue-radio-access-capability
            UERadioAccessCapability OPTIONAL
    }
```

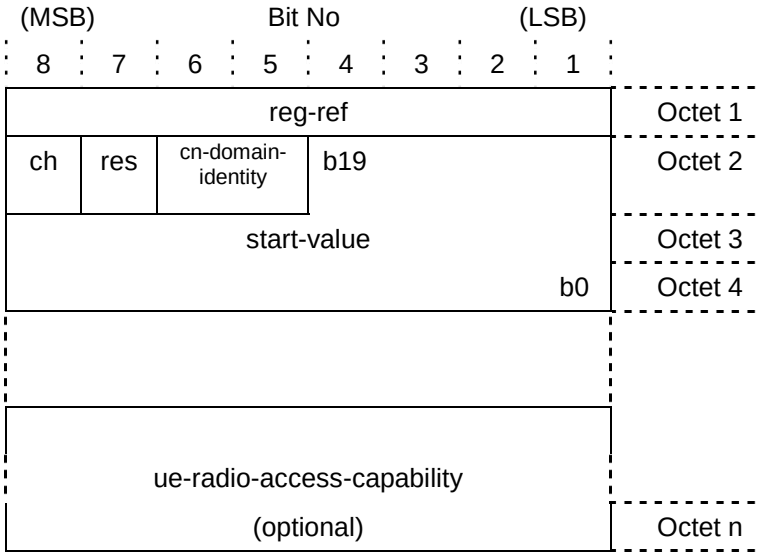


Figure 6.16: RegisterComplete Signalling PDU

The parameter *ch* (chain indicator) specifies whether the element in *start-list* is the last one in the list (if *ch* = 0) or if it is followed by another *start-list* element (if *ch* = 1). There is normally one *start-list* per Core Network (CN) domain.

```
ChainIndicator ::=
    BOOLEAN
    --TRUE if another element follows, FALSE otherwise
```

The parameter *reg-ref* is explained in clause 6.1.2.1 while *cn-domain-identity* is specified in clause 6.1.1.1.

6.2.10.1 StartValue

This parameter returns the *start-value* for the ciphering and integrity protection algorithms from the UE to the RNC and follows the definition of the same IE in [3], clauses 10.3.3.38 and 11.3:

```
StartValue ::=
    BIT STRING (SIZE(20))
```

6.2.10.2 UERadioAccessCapability

6.2.10.2.0 General

The parameter *ue-radio-access-capability* carries information about the UE's Packet Data Convergence Protocol (PDCP), security and other optional capabilities. The type UERadioAccessCapability is defined as follows:

```
UERadioAccessCapability ::=
    CHOICE {
        capability-extension-not-present
            SEQUENCE {
                pdcp-capability
```

```
                PDCPCapability,
            security-capability
                SecurityCapability
        },
    capability-extension-present
        SEQUENCE {
            long-pdcp-capability
                LongPDCPCapability,
            security-capability
                SecurityCapability,
            capability-extension
                CapabilityExtension
        }
}
```

The parameter *pdcp-capability* is either two or four octets in length, while *security-capability* has a fixed length of four octets, hence the total length of UERadioAccessCapability when the CapabilityExtension is not present is either six or eight octets.

The parameter *long-pdcp-capability* is four octets in length, hence the total length of UERadioAccessCapability when the CapabilityExtension is present will be more than 8 octets.

6.2.10.2.1 PDCPCapability and LongPDCPCapability

The parameter *pdcp-capability* carries information about the PDCP algorithms which are implemented in the UE. The parameter is of type PDCPCapability and is defined as follows, with structure as shown in Figures 6.17 and 6.18.

```
PDCPCapability ::=
    -- with Release 4 Extension
    SEQUENCE {
        lossless-SRNS-relocation-support
            BOOLEAN,
        generation-and-cid-order-complies-with-rfc-2507
            BOOLEAN,
        support-for-RFC2507
            INTEGER (0..63),
        -- full definition in Appendix 1
        support-for-RFC3095
            SEQUENCE {
                esp-profile-unsupported
                    BOOLEAN,
                udp-profile-unsupported
                    BOOLEAN,
                rtp-profile-unsupported
                    BOOLEAN,
                uncompressed-profile-unsupported
                    BOOLEAN,
                support-and-context-options
                    INTEGER (0..15),
                -- full definition in Appendix 1
                reverse-compression-depth
                    INTEGER (0..65535) DEFAULT 0
            }
    }
```

If the value of the parameter *reverse-compression-depth* is zero, then it is not included in the structure (see Figure 6.18). The parameter *generation-and-cid-order-complies-with-rfc-2507 (gc)* shall always be set to "TRUE" (i.e. the UE shall comply with RFC 2507 [i.2]).

If the parameter *support-for-rfc2507-on-um-connections-udp-only(um)* is set to "FALSE" then the RNC does not enable header compression for UM connections. If set to "TRUE", then the RNC enables header compression for UM connections, but for UDP packets only.

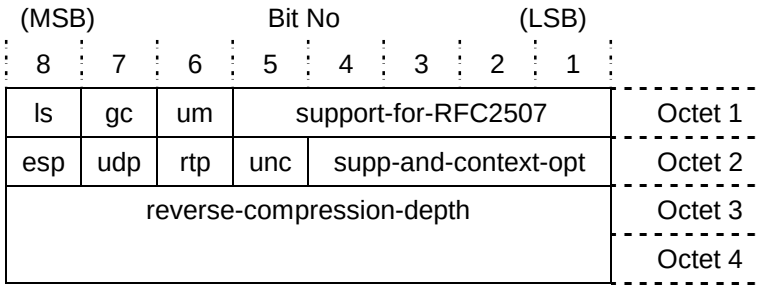


Figure 6.17: PDCPCapability Information Element (*reverse-compression-depth* included)

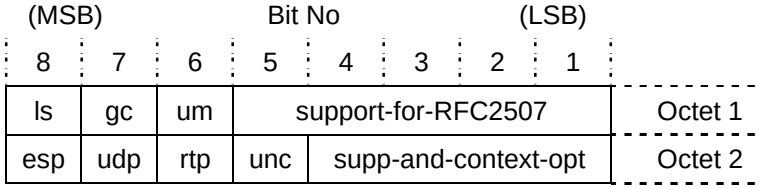


Figure 6.18: PDCPCapability Information Element (*reverse-compression-depth* not included)

The parameter *long-pdcp-capability* is the same as the parameter pdcp-capability except the reverse-compression-depth is always included (even if the value is zero). The parameter is of type LongPDCPCapability and is defined as follows:

```
LongPDCPCapability ::=
    -- with Release 4 Extension
    SEQUENCE {
        lossless-SRNS-relocation-support
            BOOLEAN,
        generation-and-cid-order-complies-with-rfc-2507
            BOOLEAN,
        support-for-rfc2507-on-um-connections-udp-only
            BOOLEAN,
        support-for-RFC2507
            INTEGER {
                rfc2507-not-supported(0),
                max-hc-context-space-by512(1),
                max-hc-context-space-by1024(2),
                max-hc-context-space-by2048(3),
                max-hc-context-space-by4096(4),
                max-hc-context-space-by8192(5),
                max-hc-context-space-by16384(6),
                max-hc-context-space-by32768(7),
                max-hc-context-space-by65536(8),
                max-hc-context-space-by131072(9)
                } (0..31),
        support-for-RFC3095
            SEQUENCE {
                esp-profile-unsupported
                    BOOLEAN,
                udp-profile-unsupported
                    BOOLEAN,
                rtp-profile-unsupported
                    BOOLEAN,
                uncompressed-profile-unsupported
                    BOOLEAN,
                support-and-context-options
                    INTEGER {
                        rfc3095-not-supported(0),
                        max-rohc-context-sessions-s2(1),
                        max-rohc-context-sessions-s4(2),
                        max-rohc-context-sessions-s8(3),
                        max-rohc-context-sessions-s12(4),
                        max-rohc-context-sessions-s16(5),
                        max-rohc-context-sessions-s24(6),
                        max-rohc-context-sessions-s32(7),
                        max-rohc-context-sessions-s48(8),
```

```
                    max-rohc-context-sessions-s64(9),
                    max-rohc-context-sessions-s128(10),
                    max-rohc-context-sessions-s256(11),
                    max-rohc-context-sessions-s512(12),
                    max-rohc-context-sessions-s1024(13),
                    max-rohc-context-sessions-s16384(14)
                    } (0..15),
                full definition in Appendix 1
                reverse-compression-depth
                    INTEGER (0..65535)
            }
    }
```

6.2.10.2.2 SecurityCapability

The parameter *security-capability* carries information about the integrity protection and ciphering algorithms which are implemented in the UE. The parameter is of type SecurityCapability and is defined as follows, with structure as shown in Figure 6.19.

```
SecurityCapability ::=
    SEQUENCE {
        ciphering-algorithm-cap
            BIT STRING (SIZE (16)),
            -- full definition in Appendix 1
        integrity-protection-algorithm-cap
            BIT STRING (SIZE (16)),
            -- full definition in Appendix 1
    }
```

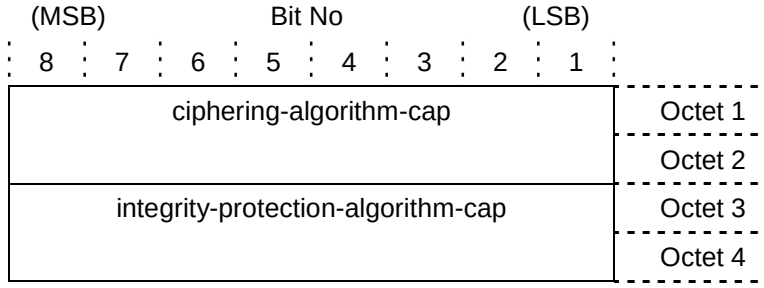


Figure 6.19: SecurityCapability Information Element

6.2.10.2.3 Capability Extension

6.2.10.2.3.0 General

The parameter *capability-extension* carries information about optional UE capabilities and shall be included in the RegisterComplete message as specified in each of the following clauses. The parameter is of typeCapabilityExtensionAVPList.

```
CapabilityExtension ::=
    CapabilityExtensionAVPList
```

6.2.10.2.3.1 CapabilityExtensionAVP Structure

6.2.10.2.3.1.0 General

The structure of the data type CapabilityExtensionAVP is as follows:

```
CapabilityExtensionAVP ::=
    CHOICE {
        cap-extn-short-avp
            CapabilityExtensionShortAVP,
        cap-extn-standard-avp
            CapabilityExtensionStandardAVP
    }
```

The CapabilityExtensionShortAVP structure is used for a value size of up to eight octets, while the CapabilityExtensionStandardAVP structure can be used for a value size of up to 256 octets. The CapabilityExtensionStandardAVP structure is currently not used.

6.2.10.2.3.1.1 CapabilityExtensionShortAVP

The CapabilityExtensionShortAVP is defined as follows, with structure as shown in Figure 6.20.

```
CapabilityExtensionShortAVP ::=
    SEQUENCE {
        length-control
            BOOLEAN, --{encode as FALSE}
        cap-extn-short-avp-type
            CapabilityExtensionShortAVPType,
        cap-etnx-short-avp-length
            INTEGER(1..8),
            -- encode as minimum bits from lowest bound
        param-value
            CHOICE {
            -- as appropriate to value of cap-extn-short-avp-type
                preferred-rnc
                    PreferredRNCParam,
                lease-mode-capability-param
                    LeaseModeCapabilityParam,
                physical-layer-capability-param
                    PhysicalLayerCapabilityParam,
                additional-ue-capabilities-param
                    AdditionalUECapabilitiesParam,
                ue-sub-class-param
                    UeSubClassParam
            }
    }
```

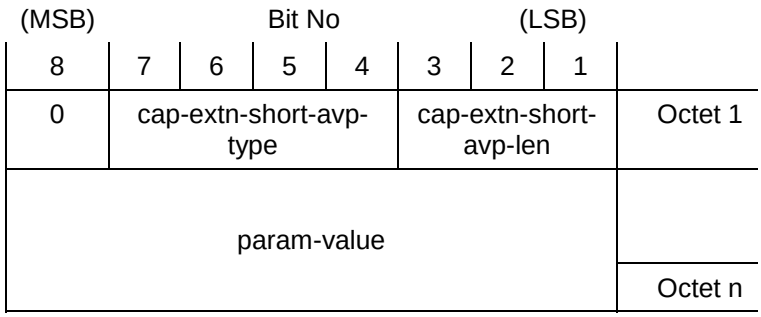


Figure 6.20: ShortAVP Structure

6.2.10.2.3.1.2 CapabilityExtensionShortAVPType

This INTEGER parameter specifies the parameter value of the short AVP:

```
CapabilityExtensionShortAVPType ::=
    INTEGER {
        preferred-rnc(5),
        lease-mode-capability(6),
        physical-layer-capability(11),
        pdcp-info(12),
        additional-capabilities(13),
        ue-subclass (14)
    } (0..15)
```

The following Short AVP Types are defined in Table 6.3.

Table 6.3: Short AVP Types

ShortAVPType	Parameter-Value	Value Length
0x05	PreferredRNC	1
0x06	LeaseModeCapability	1
0x0B	PhysicalLayerCapabilityParam	2
0x0D	AdditionalUECapabilitiesParam	1
0x0E	UESubClassParam	1

6.2.10.2.3.2 LeaseModeCapabilityParam (ShortAVPType 0x06)

The parameter has a length of one octet and carries information about subscriber membership in a BGAN Lease Group. The presence of this AVP in the RegisterComplete message is optional. The parameter is defined below, with structure as shown in Figure 6.21.

```
LeaseModeCapabilityParam ::=
    SEQUENCE {
        lease-group-id
            INTEGER (0..255)
    }
```

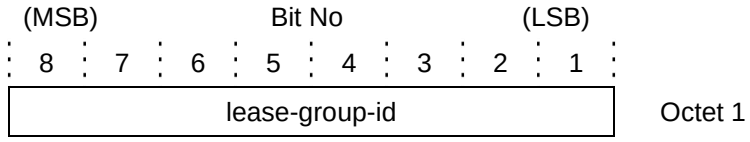


Figure 6.21: LeaseModeCapability Parameter Encoding

A Lease Group is uniquely identified by an 8-bit Lease Group ID (LGID). Valid LGID are in the range 1 to 255. The LGID value "0" is reserved to indicate that the subscriber is not a member of any Lease Group.

6.2.10.2.3.3 PhysicalLayerCapabilityParam (ShortAVPType 0x0B)

This parameter has a length of one octet and is used to inform the RAN of the version of the Bearer Table set that is stored in the UE and other physical layer capabilities of the UE. All UEs that support any of the indicated functionality shall include this AVP in the RegisterComplete message. Ues of all classes 6 to 15 shall always include this AVP in the RegisterComplete message. The parameter is defined below, with structure as shown in Figure 6.22.

```
PhysicalLayerCapabilityParam::=
    SEQUENCE {
        ldr-bearers-supported
            BOOLEAN,
        extended-l-band-supported-by-tx
            BOOLEAN,
        extended-l-band-supported-by-rx
            BOOLEAN,
        hpa-burst-power-control-supported
            BOOLEAN,
        current-bearer-table-set-version
            BearerTableSetVersion,
        hdr-forward-bearers-supported
            BOOLEAN,
        hdr-return-bearers-supported
            BOOLEAN,
        lowest-tx-frequency-as-offset
            INTEGER(0..63)
    }
```

The data type `BearerTableVersion` is defined as follows:

```
BearerTableSetVersion ::=
    INTEGER (0..15)
```

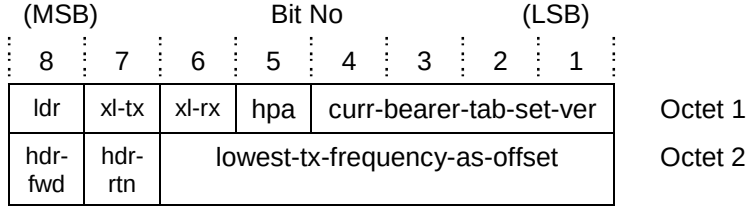


Figure 6.22: PhysicalLayerCapability Parameter Encoding

The *ldr-bearers-supported (ldr)* flag shall be set to "TRUE" if the UE receiver supports the use of F80T1Q1B, F80T2.5X4/16 and F80T5X4/16 bearers and the UE transmitter supports the use of R80T0.5Q, R80T1Q and R80T2.5X4/16 bearers.

The *extended-l-band-supported-by-tx (xl-tx)* and *extended-l-band-supported-by-rx (xl-rx)* flags shall be set to "TRUE" if the UE transmitter and/or receiver supports the extended L-Band frequency ranges as defined in the applicable clause of ETSI TS 102 744-2-2 [5].

The *hpa-burst-power-control-supported* flag shall be set to "TRUE", except when the UE HPA cannot support burst-to-burst power control.

The *lowest-tx-frequency-as-offset* indicates the lowest transmit frequency supported by this UE as an offset from 1 626,5 MHz in 100 kHz steps. This element shall normally be set to "0" (indicating the full transmit frequency range is supported) unless other values are permitted for the Class of this UE in the appropriate clause of ETSI TS 102 744-2-2 [5].

The *current-bearer-table-set-version* shall be set to "0" unless the UE received an update of the Bearer Tables from the RNC, in which case it shall be set to the version number of the most recently received BearerTableUpdate SDU (1..15).

The *hdr-forward-supported (hdr-fwd)* flag shall be set to "TRUE" if the UE receiver supports the use of F80T2.5X16/32/64 and F80T5X16/32/64 bearers.

The *hdr-return-supported (hdr-rtn)* flag is set to TRUE if the UE transmitter supports the use of R80T2.5X16/32/64 and R80T5X16/32/64 bearers.

6.2.10.2.3.4 AdditionalUECapabilitiesParam (ShortAVPType 0x0D)

This parameter has a length of one octet and is used to indicate additional UE capabilities to the RNC. The presence of this AVP in the RegisterComplete message is optional. The parameter is defined below, with structure as shown in Figure 6.23.

```
AdditionalUECapabilitiesParam ::=
    SEQUENCE {
        inter-rnc-handover-supported
            BOOLEAN,
        reserved
            BIT STRING (SIZE(5)),
        maritime-safety-data-supported
            BOOLEAN,
        maritime-safety-voice-supported
            BOOLEAN
    }
```

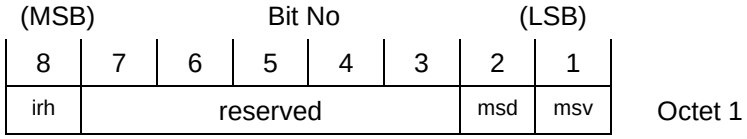


Figure 6.23: AdditionalUECapabilities Parameter Encoding

UEs which comply with the requirements for the Maritime Safety Voice service as specified in ETSI TS 102 744-2-2 [5] shall include the AdditionalUeCapabilitiesParam AVP in the RegisterComplete message and set the *maritime-safety-voice-supported (msv)* flag to "TRUE".

UEs which comply with the requirements for the Maritime Safety Data service as specified in ETSI TS 102 744-2-2 [5] shall include the AdditionalUECapabilitiesParam AVP in the RegisterComplete message and set the *maritime-safety-data-supported (msd)* flag to "TRUE".

Ues which support inter-RNC handover shall include the AdditionalUECapabilitiesParam AVP in the RegisterComplete message and set the *inter-rnc-handover-supported (irh)* flag to "TRUE".

6.2.10.2.3.5 UESubClassParam (ShortAVPType 0x0E)

This parameter has a length of one octet and is used to indicate to the RNC whether the UE belongs to a specific UESubClass (which has variations in the UE characteristics compared to the base class specified by the UEClass). This parameter is present only if the UE belongs to a subclass variant. The parameter is defined below, with structure as shown in Figure 6.24.

```
UeSubClassParam ::=
    SEQUENCE {
        ue-subclass
            INTEGER (0..255)
}
```

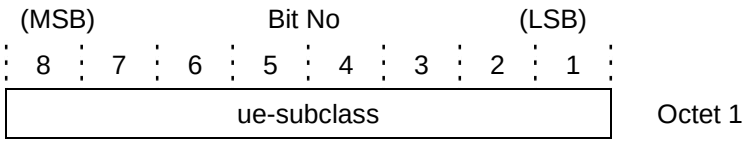


Figure 6.24: UESubClass Parameter Encoding

The *ue-subclass* value "0" is reserved to indicate that the UE is of the base class.

6.2.10.2.3.6 PreferredRNCParam (ShortAVPType 0x05)

This parameter has a length of one octet and is used to indicate to the RNC whether a particular preferred RNC has been defined on the USIM in the UE. This parameter shall be present if the Preferred RNC value is defined on the USIM. The *preferred-rnc-value* is not associated to *rnc-id* used in the Bearer Control Layer and different values for *preferred-rnc-value* and *rnc-id* may reference the same RNC. The parameter is defined below, with structure as shown in Figure 6.25.

```
PreferredRNCParam ::=
    SEQUENCE {
        preferred-rnc-value
            INTEGER (0..255)
}
```

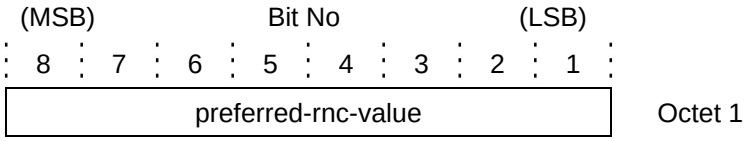


Figure 6.25: PreferredRNC Parameter Encoding

6.2.11 EstablishReject

6.2.11.0 General

This message is used by the UE to indicate a failure in the Establishment of a Bearer Connection. The PDU is defined as below, with structure as shown in Figure 6.26.

```
EstablishReject ::=
    SEQUENCE {
        bearer-conn-id
            BcnID,
        ctrl-flags
            CtrlFlags,
        trans-id
            TransactionID,
        cause
            CHOICE {
                failure-cause
                    FailureCause,
                prot-err-cause
                    ProtocolErrorCause
            }
    }
```

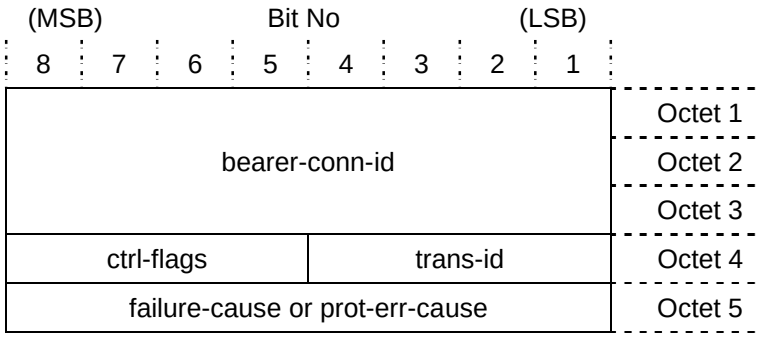


Figure 6.26: EstablishReject Signalling PDU

The parameter *bearer-conn-id* is defined in clause 6.1.3.1 while the parameters *ctrl-flags* and *trans-id* are defined in clauses 6.1.3.5 and 6.2.2.2 respectively.

6.2.11.1 FailureCause and ProtocolErrorCause

The *cause* parameter may either contain a failure cause or a protocol error cause value. The *failure-cause* parameter in the EstablishReject message is of type FailureCause and defined as follows:

```
FailureCause ::=
    INTEGER {
        configuration-not-supported(0),
        configuration-incomplete(1),
        invalid-configuration(2),
        physical-channel-failure(3),
        invalid-bcnid(4)
    } (0..255)  -- valid range 0-127
                -- 128-255 reserved for ProtocolErrorCause
```

The parameter *prot-err-cause* parameter is specified in clause 6.1.4.1.

6.2.12 ReleaseReject

This message is used by the UE to indicate a failure to release of a Bearer Connection. The PDU is defined as below, with structure as shown in Figure 6.27.

```
Release Reject ::=
    SEQUENCE {
        bearer-conn-id
            BcnID,
        ctrl-flags
            CtrlFlags,
        trans-id
            TransactionID,
        cause
            CHOICE {
                failure-cause
                    FailureCause,
                prot-err-cause
                    ProtocolErrorCause
            }
    }
```

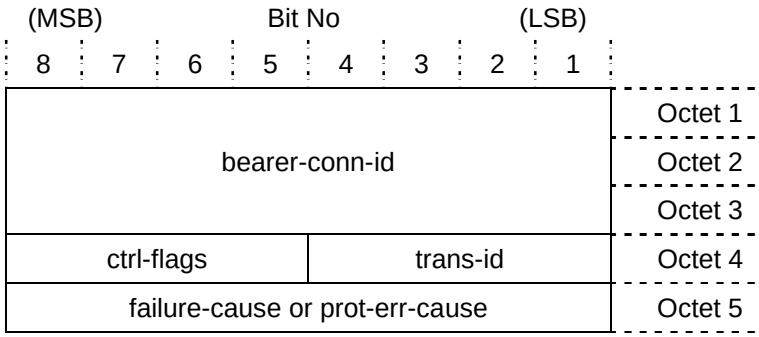


Figure 6.27: ReleaseReject Signalling PDU

The parameter *bearer-conn-id* is defined in clause 6.1.3.1 while the parameters *ctrl-flags* and *trans-id* are defined in clauses 6.1.3.5 and 6.2.2.2 respectively. The parameters *failure-cause* and *prot-err-cause* parameter are specified in clause 6.1.4.1.

6.2.13 ModifyReject

This message is used by the UE to indicate a failure to modify a Bearer Connection. The PDU is defined as below, with structure as shown in Figure 6.28.

```
ModifyReject ::=
    SEQUENCE {
        bearer-conn-id
            BcnID,
        ctrl-flags
            CtrlFlags,
        trans-id
            TransactionID,
        cause
            CHOICE {
                failure-cause
                    FailureCause,
                prot-err-cause
                    ProtocolErrorCause
            }
    }
```

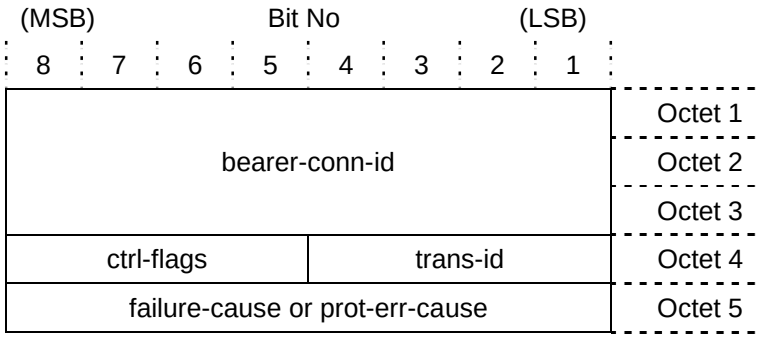


Figure 6.28: ModifyReject Signalling PDU

The parameter *bearer-conn-id* is defined in clause 6.1.3.1 while the parameters *ctrl-flags* and *trans-id* are defined in clauses 6.1.3.5 and 6.2.2.2 respectively. The parameters *failure-cause* and *prot-err-cause* parameter are specified in clause 6.1.4.1.

6.2.14 PagingType2

6.2.14.0 General

The PagingType2 Signalling PDU is used by the RNC when a UE-Specific Signalling Connection exists between the UE and the RNC. The PDU is defined as below, with structure as shown in Figure 6.29.

```
PagingType2 ::=
    SEQUENCE {
        cn-domain-identity
            CNDomainIdentity,
        paging-cause
            PagingCause,
        paging-record-type-id
            PagingRecordTypeID
    }
```

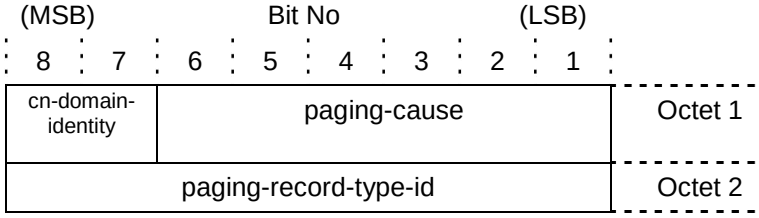


Figure 6.29: PagingType2 Signalling PDU

The parameters *cn-domain-identity* and *paging-cause* are defined in clauses 6.1.1.1 and 6.1.1.2 respectively.

6.2.14.1 PagingRecordTypeID

This IE is defined in [3], clauses 10.3.1.10 and 11.3 as follows:

```
PagingRecordTypeID ::=
    INTEGER {
        imsi (0),
        tmsi-or-p-tmsi (1)
    } (0..255)
```

6.2.15 InitialDirectTransfer

6.2.15.0 General

This Signalling PDU is used by the UE to transfer the initial NAS message from the UE via the RNC to the Core Network prior to the establishment of an Iu signalling connection. The NAS message is passed transparently through the RNC. The PDU is defined as below, with structure as shown in Figure 6.30.

```
InitialDirectTransfer::=
    SEQUENCE {
        reserved
            BIT STRING (SIZE (6)),
        cn-domain-identity
            CNDomainIdentity,
        nas-message
            NASMessage
    }
```

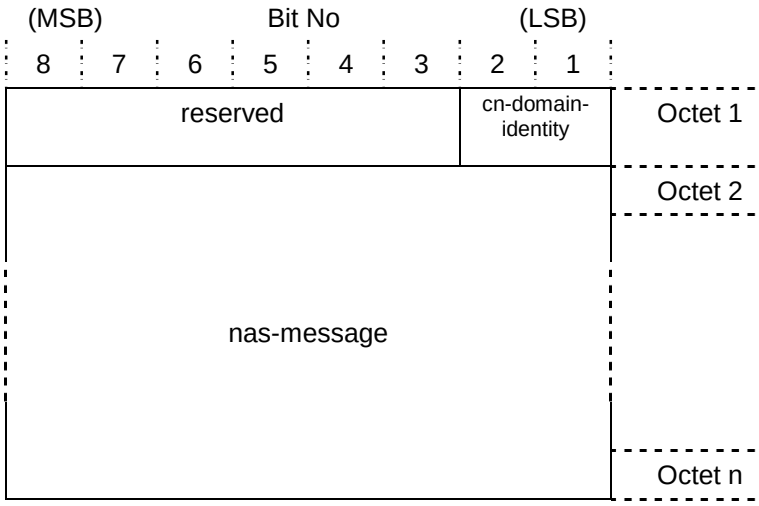


Figure 6.30: InitialDirectTransfer Signalling PDU

The parameter *cn-domain-identity* is defined in clause 6.1.1.1.

6.2.15.1 NASMessage

This parameter carries the message between Non-Access-Stratum peers (e.g. Mobility Management messages) is not interpreted by the Adaptation Layer. Its definition in [3], clauses 10.3.3.14 and 11.3 is modified for the satellite network as follows:

```
NASMessage ::=
    OCTET STRING (SIZE (1..2040))
    -- Satellite network RI cannot support UMTS maximum size of 4095
```

In the event that the AdaptationLayer is requested to send a NASMessage which exceeds the maximum size of NASMessage then the Adaptation Layer shall truncate the message at the maximum length before passing it to the Bearer Connection Layer.

6.2.16 UplinkDirectTransfer

This Signalling PDU is used by the UE to transfer NAS messages via the RNC to the Core Network if an Iu signalling connection already exists. The NAS message is passed transparently through the RNC. The PDU is defined as below, with structure as shown in Figure 6.31.

```
UplinkDirectTransfer::=
    SEQUENCE {
        reserved
            BIT STRING (SIZE (6)),
        cn-domain-identity
            CNDomainIdentity,
        nas-message
            NASMessage
    }
```

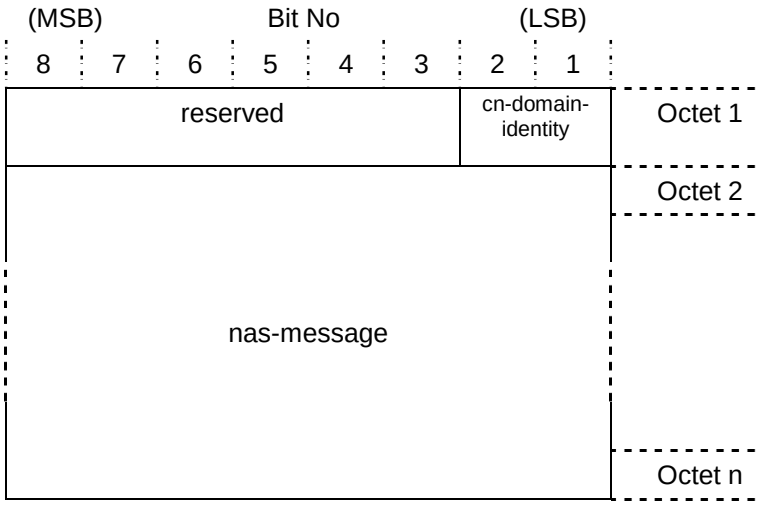


Figure 6.31: UplinkDirectTransfer Signalling PDU

The parameters *cn-domain-identity* and *nas-message* are defined in clauses 6.1.1.1 and 6.2.15.2 respectively.

6.2.17 DownlinkDirectTransfer

This Signalling PDU is used by the RNC to transfer NAS messages from the Core Network to the UE if an Iu signalling connection already exists. The NAS message is passed transparently through the RNC. The PDU is defined as below, with structure as shown in Figure 6.32.

```
DownlinkDirectTransfer ::=
    SEQUENCE {
        reserved
            BIT STRING (SIZE (6)),
        cn-domain-identity
            CNDomainIdentity,
        nas-message
            NASMessage
    }
```

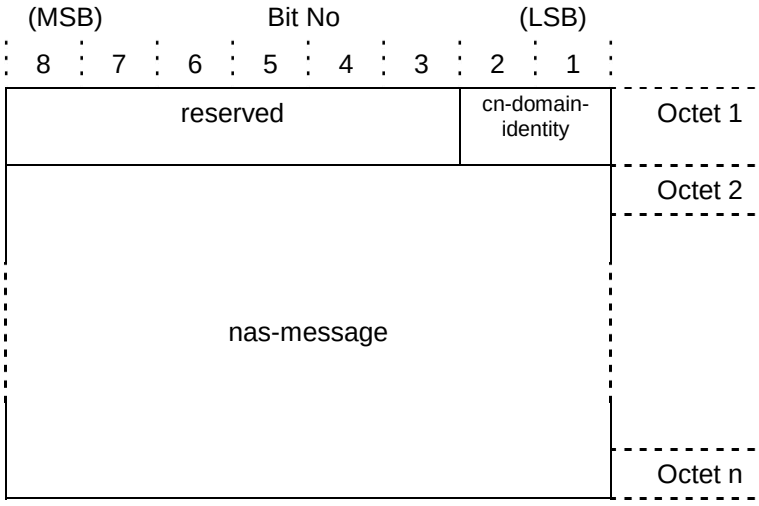


Figure 6.32: DownlinkDirectTransfer Signalling PDU

The parameters *cn-domain-identity* and *nas-message* are defined in clauses 6.1.1.1 and 6.2.15.2 respectively.

6.2.18 SecurityModeCommand

This Signalling PDU is used by the RNC to start or reconfigure integrity protection and/or ciphering in the UE. The PDU is defined as below, with structure as shown in Figure 6.33.

```
SecurityModeCommand ::=
    SEQUENCE {
        reserved
            BIT STRING (SIZE (6)),
        cn-domain-identity
            CNDomainIdentity,
        security-capability
            SecurityCapability,
        ciphering-mode-info-avp
            CipheringModeInfoAVP OPTIONAL,
        integrity-protection-mode-info-avp
            IntegrityProtectionModeInfoAVP OPTIONAL
    }
```

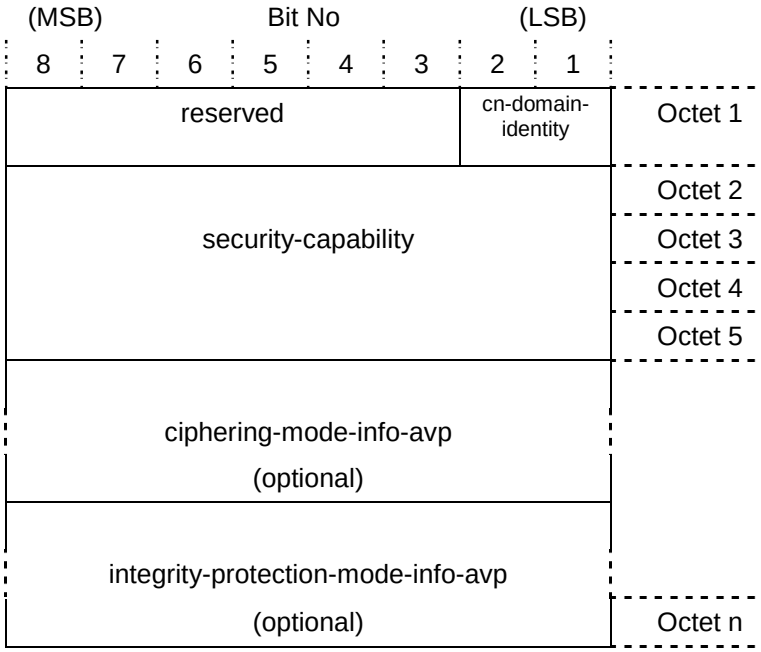


Figure 6.33: SecurityModeCommand Signalling PDU

The parameter *cn-domain-identity* is defined in clause 6.1.1.1 and *security-capability* is defined in clause 6.2.10.2.2. *Ciphering-mode-info-avp* and *integrity-protection-mode-info-avp* are defined in clauses 6.3.11 and 6.3.13 respectively.

6.2.19 SecurityModeComplete

This message is sent from the UE to the RNC to confirm the start or reconfiguration of integrity protection and/or ciphering in the UE. The PDU is defined as below, with structure as shown in Figure 6.34.

```
SecurityModeComplete::=
    SEQUENCE {
        ul-integrity-protection-activation-info-avp
            ULIntegrityProtectionActivationInfoAVP OPTIONAL,
        ul-ciphering-activation-time-info-list
            RBActivationTimeInfoList OPTIONAL
        }
```

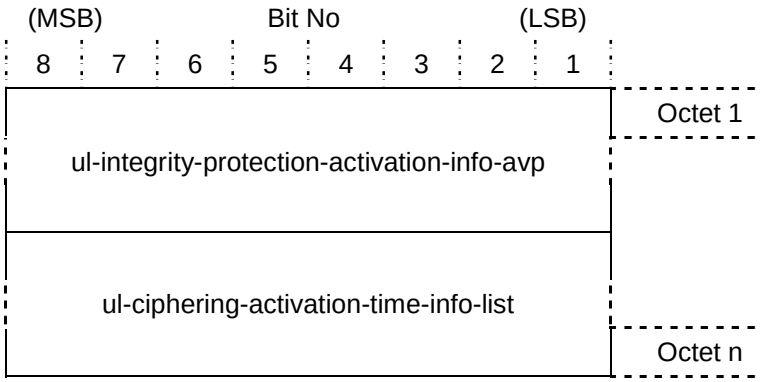


Figure 6.34: SecurityModeComplete Signalling PDU

The parameters *ul-integrity-protection-activation-info-avp* and *ul-ciphering-activation-time-info-list* (which is of type RBActivationTimeInfoList) are defined in clauses 6.3.10 and 6.3.11.2 respectively.

6.2.20 SecurityModeFailure

6.2.20.0 General

This message is sent from the UE to the RNC to indicate a failure to act on a received SecurityModeControl message. The PDU is defined as below, with structure as shown in Figure 6.35.

```
SecurityModeFailure::=
    SEQUENCE {
        cause
            CHOICE {
                security-failure-cause
                    SecurityFailureCause,
                prot-err-cause
                    ProtocolErrorCause
            },
        ul-integrity-protection-activation-info-avp
            ULIntegrityProtectionActivationInfoAVP OPTIONAL
    }
```

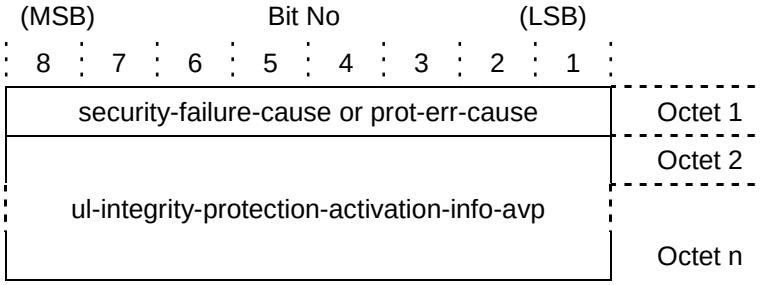


Figure 6.35: SecurityModeFailure Signalling PDU

The parameter *prot-err-cause* is defined in clause 6.1.4.1. The parameter *ul-integrity-protection-activation-info-avp* is defined in clause 6.3.10.

6.2.20.1 SecurityFailureCause

The parameter *security-failure-cause* is of type SecurityFailureCause and is defined as follows:

```
SecurityFailureCause ::=
    INTEGER {
        unsupported-or-mismatched-security-configuration (0),
        integrity-protection-algorithm-failure (1),
        ciphering-algorithm-failure (2)
    } (0..255)  -- valid range 0-127
                -- 128-255 reserved for ProtocolErrorCause
```

6.2.21 SignallingConnectionReleaseReq

6.2.21.0 General

This message is sent from the UE to the RNC to request the release of an existing Iu signalling connection from the RNC towards the specified CN domain. The PDU is defined as below, with structure as shown in Figure 6.36.

```
SignallingConnectionReleaseReq ::=
    SEQUENCE {
        reserved
            BIT STRING (SIZE (6)),
        cn-domain-identity
            CNDomainIdentity,
        cause
            CHOICE {
                conn-rel-cause
                    ConnectionReleaseCause,
                prot-err-cause
                    ProtocolErrorCause
            }
    }
```

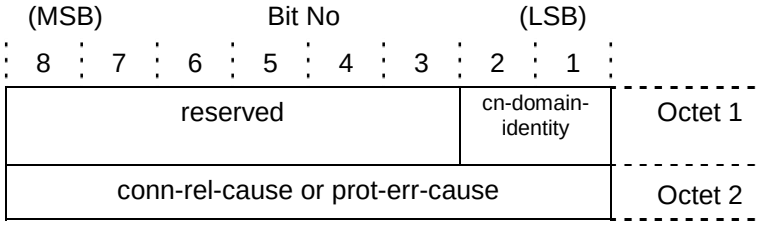


Figure 6.36: SignallingConnectionReleaseRequest Signalling PDU

The parameter *cn-domain-identity* is defined in clause 6.1.1.1 while *prot-err-cause* is defined in clause 6.1.4.1.

6.2.21.1 ConnectionReleaseCause

The parameter *conn-rel-cause* is of type ConnectionReleaseCause and is defined as follows:

```
ConnectionReleaseCause ::=
    INTEGER {
        mm-sublayer-initiated (0)
    } (0..255)  -- valid range 0-127
                -- 128-255 reserved for ProtocolErrorCause
```

6.2.22 SignallingConnectionRelease

This message is sent by the RNC to inform the UE that the ongoing Iu signalling connection from the RNC to the specified CN domain has been released. The PDU is defined as below, with structure as shown in Figure 6.37.

```
SignallingConnectionRelease::=
    SEQUENCE {
        reserved
            BIT STRING (SIZE (6)),
        cn-domain-identity
            CNDomainIdentity
    }
```

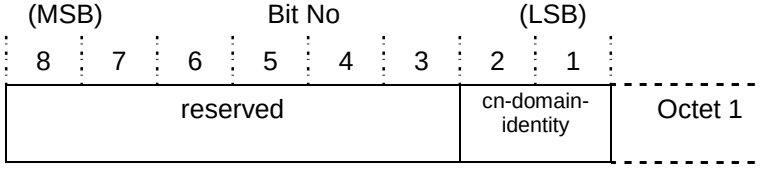


Figure 6.37: SignallingConnectionRelease Signalling PDU

The parameter *cn-domain-identity* is defined in clause 6.1.1.1.

6.2.23 UEPositionRequest

This message is used by the RNC during the registration procedure (only) to command the UE to provide its GPS position. To maintain user confidentiality, the RNC may either provide a public key to be used by the UE to encrypt its response, or an index to a pre-defined public key which is stored in the UE. The PDU is defined as below, with structure as shown in Figures 6.38 and 6.39.

```
UEPositionRequest ::=
    CHOICE {
        public-key-index
            PublicKeyIndex,
        public-key
            PublicKey
    }
```

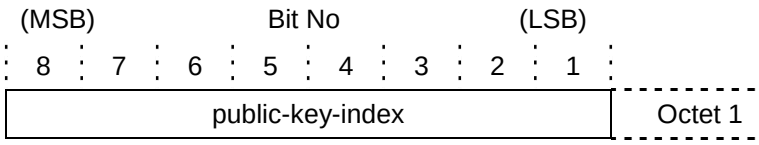


Figure 6.38: UEPositionRequest Signalling PDU (with PublicKeyIndex)

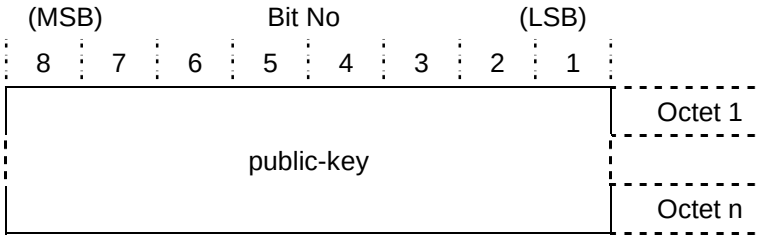


Figure 6.39: UEPositionRequest Signalling PDU (with PublicKey provided)

The parameter *public-key-index* is of type PublicKeyIndex which is defined as follows:

```
PublicKeyIndex ::=
    INTEGER (0..255)
```

The parameter *public-key-index* selects a public key that shall be used for encryption from a list stored in the UE, except if *public-key-index* = 0, then the GPS position shall be sent unencrypted.

The parameter *public-key* is of type PublicKey which is defined as follows:

```
PublicKey ::=
    SEQUENCE {
        n-length
            INTEGER(0..65535) (CONSTRAINED BY {-- length of n in bits
                expect (640<..<1024) in multiples of 8 bits --}),
        n
            OCTET STRING (SIZE (0..maxKeySize)) (CONSTRAINED BY
                {-- expect (80<..<128) appropriate to value
                    of n-length --}),
        e-length
            INTEGER(0..65535) (CONSTRAINED BY {-- length of e in bits
                expect (8<..1024) in multiples of 8 bits --}),
        e
            OCTET STRING (SIZE (0..maxKeySize)) (CONSTRAINED BY,
                {-- expect (1<..<128) appropriate to value
                    of e-length --})
    }
```

If the UEPositionRequest message contains *public-key* or a non-zero value of *public-key-index* then the UE shall encrypt *ue-position* in the UEPositionResponse (solicited or unsolicited) and HandoverRequest messages. The encryption parameters in the UEPositionRequest message shall also be used to encrypt the *ue-position* in any subsequent unsolicited UEPositionResponse message and also in any HandoverRequest message.

6.2.24 UEPositionResponse

6.2.24.0 General

This message is used by the UE to provide the GPS position to the RNC, either in response to the UEPositionRequest message or unsolicited (see clause 6.2.25). The GPS position may be encrypted using the method described in ETSI TS 102 744-3-6 [8]. The PDU is defined as below, with structure as shown in Figure 6.40.

```
UEPositionResponse ::=
    SEQUENCE {
        ue-position
            CHOICE {
                unencrypted
                    GPSPositionString,
                encrypted
                    EncryptedGPSPositionString,
                spot-beam-id
                    SpotBeamID
            }
    }
```

If the UEPositionRequest message contained *public-key* or a non-zero value of *public-key-index* then the UE shall encrypt *ue-position* in the UEPositionResponse message (solicited or unsolicited).

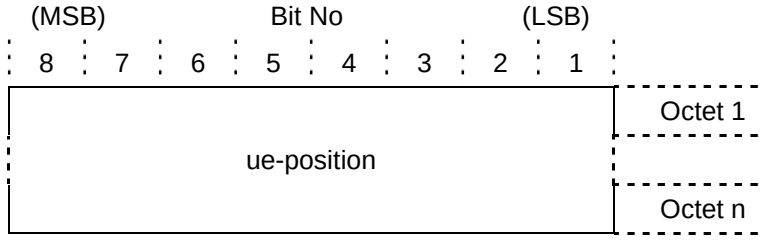


Figure 6.40: UEPositionResponse Signalling PDU

6.2.24.1 Ue-position

The format of the *ue-position* string (prior to encryption) is derived from the National Marine Electronics Association (NMEA-0183) sentence type GGA [i.1]. It is either 35 or 38 characters long with each character encoded as 8-bit ASCII. All position reporting in terms of latitude and longitude and as well as any conversions of position to other coordinate systems shall be in accordance with the WGS-84 reference ellipsoid. The type GPSPositionString is defined as follows:

```
GPSPositionString ::=
    -- Interpret IA5String as 8-bit US ASCII
    -- Use zero-fill in all fields as required (fill from left)
    SEQUENCE {
        gps-fix-date
            SEQUENCE {
                year
                    IA5String (SIZE(4))(FROM("0".."9")),
                month
                    IA5String (SIZE(2))(FROM("0".."9")),
                day
                    IA5String (SIZE(2))(FROM("0".."9"))
            },
        gps-fix-time - UTC
            SEQUENCE {
                hours
                    IA5String (SIZE(2))(FROM("0".."9")),
                minutes
                    IA5String (SIZE(2))(FROM("0".."9")),
                seconds
                    IA5String (SIZE(2))(FROM("0".."9"))
            },
        latitude
            SEQUENCE {
                degrees
                    IA5String (SIZE(2))(FROM("0".."9")),
                minutes-units
                    IA5String (SIZE(2))(FROM("0".."9")),
```

```
                minutes-thousandths
                    IA5String (SIZE(3))(FROM("0".."9")),
                latitude-sense
                    IA5String (SIZE(1))(FROM("N"|"S"))
                    -- "N" indicates degrees North,
                    -- "S" indicates Degrees South
            },
        longitude
            SEQUENCE {
                degrees
                    IA5String (SIZE(3))(FROM("0".."9")),
                minutes-units
                    IA5String (SIZE(2))(FROM("0".."9")),
                minutes-thousandths
                    IA5String (SIZE(3))(FROM("0".."9")),
                longitude-sense
                    IA5String (SIZE(1))(FROM("E"|"W"))
                    -- "E" indicates degrees East,
                    -- "W" indicates Degrees West
            },
        fix-quality
            CHOICE {
                gps-fix
                    IA5String (SIZE(1))(FROM("1")),
                dgps-fix
                    IA5String (SIZE(1))(FROM("2")),
                user-specified-position
                    IA5String (SIZE(1))(FROM("3")),
                irs-fix
                    IA5String (SIZE(1))(FROM("4"))
            },
        next-part
            CHOICE {
                gps-or-dgps-fix
                    SEQUENCE {
                        number-of-satellites-tracked
                            IA5String (SIZE(1))(FROM("0"."9")),
                            -- encode as "9" if 9 or more
                            -- satellites are tracked
                        horizontal-dilution-of-precision
                            SEQUENCE {
                                units
                                    IA5String (SIZE(1))(FROM("0".."9")),
                                    -- encode as "9" if HDOP is greater
                                    -- or equal to 9.9
                                tenths
                                    IA5String (SIZE(1))(FROM("0".."9"))
                                    -- encode as "9" if HDOP is greater
                                    -- or equal to 9.9
                            },
                        loa-time
                            IA5String(SIZE(3))(FROM("0".."9")) OPTIONAL
                            -- encode as "255" if greater than 255 minutes
                    },
                user-specified-position-or-irs-fix
                    SEQUENCE {
                        number-of-satellites-tracked
                            IA5String (SIZE(1))(FROM("0")),
                        horizontal-dilution-of-precision
                            IA5String (SIZE(2))(FROM("9"))
                    }
            }
    }
```

The type EncryptedGPSPositionString is defined as follows:

```
EncryptedGPSPositionString ::=
    OCTET STRING (SIZE(80..maxKeySize))
```

The parameter *spot-beam-id* is used if no other position information is available at the UE. In this case the spot beam ID as broadcast in the current System Information shall be used. The data type SpotBeamID is defined as follows:

```
SpotBeamID ::=
    INTEGER (0..255)
```

6.2.25 RegModeUpdate

This message is used by the RNC to change the registration mode and to control the GPS position display capabilities (if implemented) of a UE at any time after the registration process has been completed. The PDU is defined as below, with structure as shown in Figure 6.41.

```
RegModeUpdate ::=
    SEQUENCE {
        reg-ref
            RegistrationReference,
        reserved
            BIT STRING (SIZE (4)),
        reg-mode
            RegistrationMode,
        gps-report-distance
            GPSReportDistance OPTIONAL
    }
```

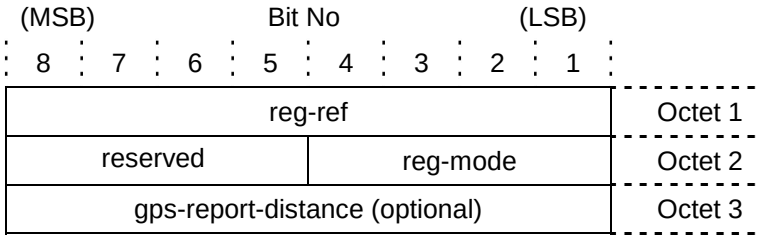


Figure 6.41: RegModeUpdate Signalling PDU

The optional parameter *gps-report-distance* specifies the distance (in km) that a UE can move before it has to send an unsolicited UEPositionResponse message to the RNC. The type GPSReportDistance is defined as follows:

```
GPSReportDistance ::=
    INTEGER (0..255)
```

A zero value in *gps-report-distance* is invalid and shall not be used by the RNC.

The parameters *reg-ref* and *reg-mode* are defined in clauses 6.1.2.1 and 6.1.3.10 respectively.

6.2.26 SystemInformation

This message is used by the RNC to transfer System Information to the UE while in the connected state. The PDU is defined as below, with structure as shown in Figure 6.42.

```
SystemInformation ::=
    bct-epdu
        BCtEPDU
```

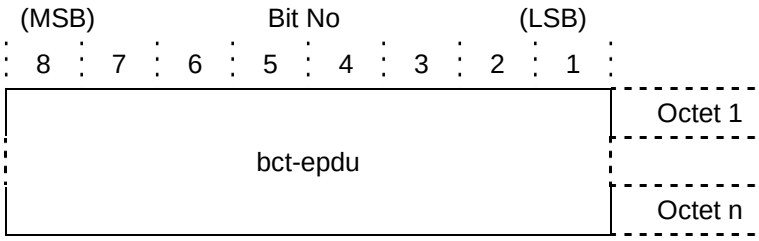


Figure 6.42: SystemInformation Signalling PDU

The format of *bct-epdu* is specified in ETSI TS 102 744-3-1 [6].

6.2.27 Deregister

The Deregister message is used by the RNC to deregister an individual Ues. The Deregister Common Signalling PDU is defined as below, with structure as shown in Figure 6.43.

```
Deregister ::=
    SEQUENCE {
        reg-ref
            RegistrationReference,
        cause
            CHOICE {
                deregistration-cause
                    DeregistrationCause,
                prot-err-cause
                    ProtocolErrorCause
            }
    }
```

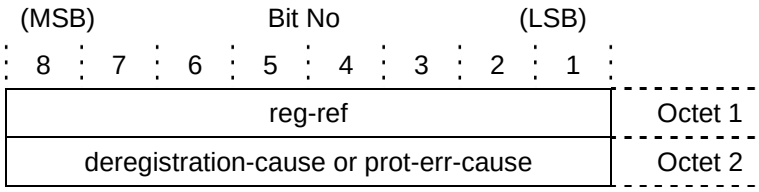


Figure 6.43: Deregister Signalling PDU

The parameters *reg-ref* is defined in clause 6.1.2.1, *deregistration-cause* is defined in clause 6.1.5.1 while *prot-err-cause* is defined in clause 6.1.4.1.

6.2.28 DeregisterAck

The DeregisterAck message is used by the UE to complete the deregistration process. The DeregisterAck Signalling PDU is defined as below, with structure as shown in Figure 6.44.

```
DeregisterAck ::=
    SEQUENCE {
        reg-ref
            RegistrationReference
        }
```

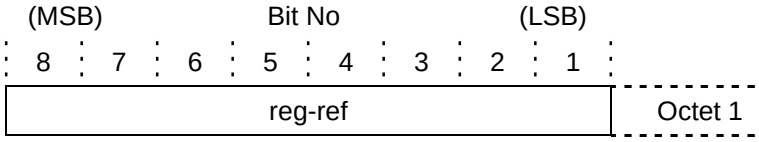


Figure 6.44: DeregisterAck Signalling PDU

The parameter *reg-ref* is defined in clause 6.1.2.1.

6.2.29 HandoverRequest

The HandoverRequest message is used by a moving UE to signal to the RNC that it has changed position and that a Handover into another spot beam may be required. The HandoverRequest Signalling PDU is defined as below, with structure as shown in Figure 6.45.

```
HandoverRequest ::=
    SEQUENCE {
        handover-info
            SEQUENCE {
                reserved
                    BIT STRING (SIZE(2)),
                current-spot-beam-included
                    BOOLEAN,
                ue-position-included
                    BOOLEAN,
                observe-target-preference
                    BOOLEAN,
                target-list-length
                    INTEGER (0..7),
                current-spot-beam
                    SpotBeamID OPTIONAL,
                target-spot-beam-list
                    SEQUENCE SIZE(0..7) OF SpotBeamID,
                    --target-list-length indicates size
```

```
            ue-position
                CHOICE {
                    unencrypted
                        GPSPositionString,
                    encrypted
                        EncryptedGPSPositionString
                } OPTIONAL


        }
}
```

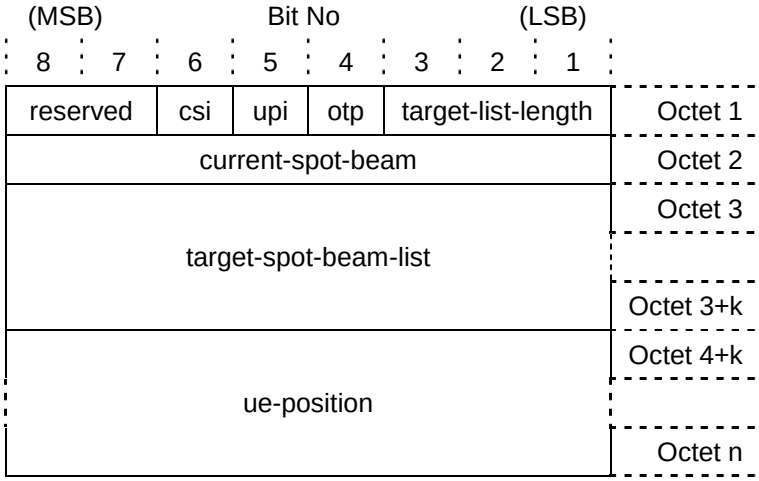


Figure 6.45: HandoverRequest Signalling PDU (*current-spot-beam* and *ue-position included*)

The data types GPSPositionString, EncryptedGPSPositionString, and SpotBeamID are defined in clause 6.2.24.1.

The value of the BOOLEAN *ue-position-included* (*upi*) flag indicates whether the *ue-position* element is included within the PDU. The value of the BOOLEAN *current-spot-beam-included* (*csi*) flag indicates whether the *current-spot-beam* element is included within the PDU. The use of the *observe-target-preference (otp)* flag is described in ETSI TS 102 744-3-6 [8].

If the UEPositionRequest message contained *public-key* or a non-zero value of *public-key-index* then the UE shall encrypt *ue-position* in the HandoverRequest message.

6.3 Adaptation Layer AVPs

6.3.0 General

Adaptation Layer AVPs (AL-AVPs) are used to carry Information Elements (Ies) which are required to provide additional signalling for UMTS specific features (e.g. ciphering, PDCP parameters etc.) in messages related to the establishment, modification and release of user plane connections. In this case, AL-AVPs are contained in the BcnParamList data type (see clause 6.1.3.4). Tables 6.4 and 6.5 list the available AL-AVPs and specify whether they shall be contained in different Adaptation Layer PDUs.

Table 6.4: AL-AVPs supported in Adaptation Layer PDUs from RNC

Parameter Value	Establish	Modify
CipheringModeInfoParam	OP	n/a
PDCPInfoParam	MP if PS	MP if PS
RABInfoParam	MP	MP
NOTE: MP: Mandatory Presence; OP: Optional Presence; n/a: not applicable.		

Table 6.5: AL-AVPs supported in Adaptation Layer PDUs from UE

Attribute-Value-Pair	EstablishAck	ModifyAck
COUNT-CactivationTimeParam	OP	n/a
ULCipheringActivationTimeInfoParam	OP	n/a
CSCallTypeParam	MP if CS	n/a
NOTE: MP: Mandatory Presence; OP: Optional Presence; n/a: not applicable.		

Adaptation Layer AVPs may also be sent in other messages (either individually or in a list of type AdaptationLayerAVPList) in messages related to security mode control and registration.

6.3.1 AdaptationLayerAVP Structure

The structure of the data type AdaptationLayerAVP is as follows:

```
AdaptationLayerAVP ::=
    CHOICE {
        al-short-avp
            ALShortAVP,
        al-standard-avp
            ALStandardAVP
    }
```

The ALShortAVP structure is used for a value size of up to eight octets, while the ALStandardAVP structure can be used for a value size of up to 256 octets.

6.3.2 ALShortAVP

6.3.2.0 General

The ALShortAVP is defined as below, with structure as shown in Figure 6.46.

```
ALShortAVP ::=
    SEQUENCE {
        length-control
            BOOLEAN, --{encode as FALSE}
        al-short-avp-type
            ALShortAVPType,
        al-short-avp-length
            INTEGER(1..8),
            -- encode as minimum bits from lowest bound
        param-value
            CHOICE {
            -- as appropriate to value of al-short-avp-type
                count-c-activation-time-param
                    CountCActivationTimeParam,
                pdcp-sn-info-param
                    PDCPSNInfoParam,
                rab-info-param
                    RABInfoParam,
                ul-ciphering-activation-time-info-param
                    ULCipheringActivationTimeInfoParam,
                cs-call-type-param
                    CSCallTypeParam,
                ul-integrity-protection-activation-info-param
                    ULIntegrityProtectionActivationInfoParam,
                ciphering-mode-info-param
                    CipheringModeInfoParam,
                integrity-protection-mode-info-param
                    IntegrityProtectionModeInfoParam,
                pdcp-info-param
                    PDCPInfoParam
            }
    }
```

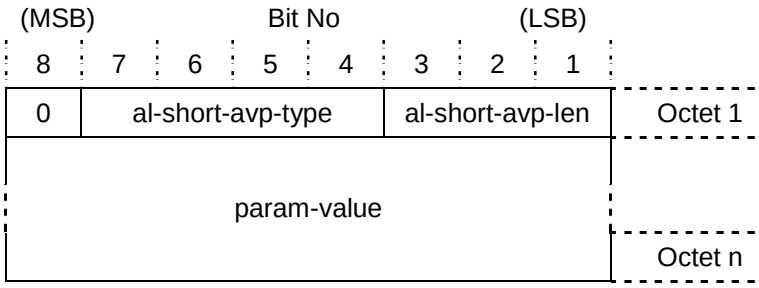


Figure 6.46: ShortAVP Structure

6.3.2.1 ALShortAVPType

This INTEGER parameter specifies the parameter value of the short AVP:

```
ALShortAVPType ::=
    INTEGER {
        count-c-activation-time(1),
        pdcp-sn-info(2),
        rab-info(3),
        uplink-ciphering-activation-time-info(4),
        cs-call-type(5),
        uplink-integrity-protection-activation-info(7),
        ciphering-mode-info(8),
        reserved (9), -- placeholder for access-class-capability(9)
        integrity-protection-mode-info(10),
        pdcp-info(12)
    } (0..15)
```

The Short AVP Types are defined in Table 6.6.

Table 6.6: Short AVP Types

ShortAVPType	Parameter-Value	Value Length
0x01	CountCActivationTimeParam	2
0x02	PDCPSNInfoParam	2
0x03	RABInfoParam	variable
0x04	ULCipheringActivationTimeInfoParam	2
0x05	CSCallTypeParam	1
0x07	ULIntegrityProtectionActivationInfoParam	1
0x08	CipheringModeInfoParam	variable
0x0A	IntegrityProtectionModeInfoParam	1, 2, 4 or 5
0x0C	PDCPInfoParam	variable

6.3.3 ALStandardAVP

6.3.3.0 General

The ALStandardAVP is defined as below, with structure as shown in Figure 6.47.

```
ALStandardAVP ::=
    SEQUENCE {
        length-control
            BOOLEAN, --{encode as TRUE}
        al-standard-avp-type
            ALStandardAVPType,
        al-standard-avp-length
            INTEGER(1..256),
            -- encode as minimum bits from lowest bound
        param-value
            CHOICE {    -- al-standard-avp-type
                ciphering-mode-info-param
                    CipheringModeInfoParam,
                pdcp-info-param
                    PDCPInfoParam
                group-cipher-info-param
                    GroupCipherInfoParam
```

```
        }
    }
```

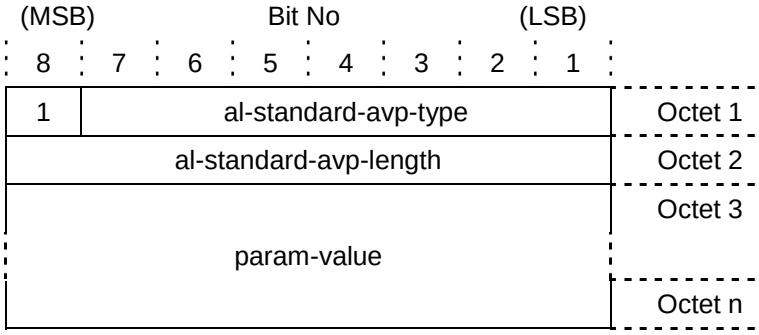


Figure 6.47: Standard-AVP Structure

6.3.3.1 ALStandardAVPType

This INTEGER parameter specifies the parameter value of the standard AVP:

```
ALStandardAVPType ::=
    INTEGER {
        ciphering-mode-info(8),
        pdcp-info(12)
        group-cipher-info(13)
    } (0..127)
```

The Standard AVP Types are defined in Table 6.7.

Table 6.7: Standard AVP Types

StandardAVPType	Parameter-Value	StandardAVPLen
0x08	CipheringModeInfoParam	Variable
0x0C	PDCPInfoParam	variable
0x0D	GroupCipherInfoParam	20 octets

All StandardAVPs in Table 6.7 may also be encoded as ShortAVPs if the parameter value field is less than nine octets long.

The following clauses specify the parameters in each of the AVPs introduced above. It should be noted that the figures illustrate the layout of the *param-value* field only.

6.3.4 CountCActivationTimeParam (ShortAVPType 0x01)

This AVP parameter specifies the COUNT-C activation time for a transparent mode (TM) connection. The parameter is defined as below, with structure as shown in Figure 6.48.

```
CountCActivationTimeParam ::=
    SEQUENCE {
        reserved
            BIT STRING (SIZE (4)),
        frame-number
            FrameNumber
    }
```

The data type FrameNumber is defined as follows:

```
FrameNumber ::=
    INTEGER (0..4095)
```

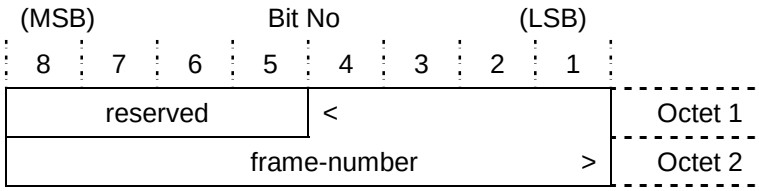


Figure 6.48: CountCActivationTime Parameter Encoding

This parameter has a fixed length of two octets and the AVP is encoded as a short AVP type.

6.3.5 PDCPSNInfoParam (ShortAVPType 0x02)

This AVP parameter encapsulates the PDCP Sequence Number (PDCP SN) Info IE specified in [3], clause 10.3.4.3. The parameter is defined as below, with structure as shown in Figure 6.49.

```
PDCPSNInfoParam ::=
    INTEGER (0..65535)
```

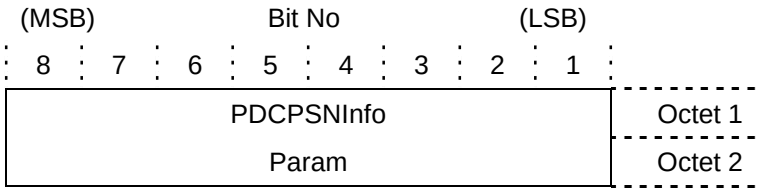


Figure 6.49: PDCPInfo Parameter Encoding

This parameter has a fixed length of two octets and the AVP is encoded as a short AVP type.

6.3.6 RABInfoParam (ShortAVPType 0x03)

6.3.6.0 General

This AVP parameter encapsulates a number of Ies required to signal Radio Access Bearer (RAB) related parameters to the UE. The parameter is defined as below, with structure as shown in Figures 6.50 and 6.51.

```
RABInfoParam ::=
    SEQUENCE {
        rab-Identity            RABIdentity,
        nas-sync-ind-included   BOOLEAN,
        ext-rab-info-present    BOOLEAN,
        cn-domain-identity      CNDomainIdentity,
        nas-sync-or-thp
            CHOICE {
                thp- info
                    SEQUENCE {
                        reserved2
                            BIT STRING (SIZE (2)),
                        thp
                            TrafficHandlingPriority
                    }
                nas-synchronisation-indicator
                            NASSynchronisationIndicator             }
        ext-rab-info    SEQUENCE {
            rab-info-type   RabInfoType,
            rab-info        CHOICE {
                rab-access-priority-info SEQUENCE {
                    reserved3 BIT STRING (SIZE (2)),
                    rab-access-priority RabAccessPriority
                }
            reserved    BIT STRING (SIZE 6)
            }
        } OPTIONAL
    }
```

Figure 6.50: RABInfo Parameter Encoding (without NAS Synchronization Indicator)

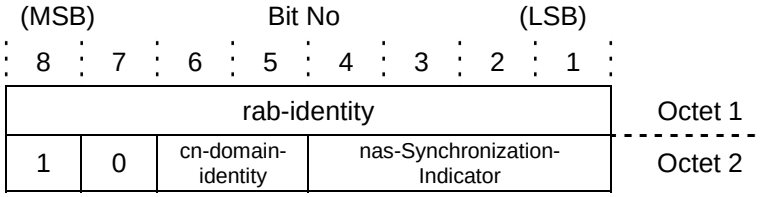


Figure 6.51: RABInfo Parameter Encoding (with NAS Synchronization Indicator)

The *nas-sync-ind-included* flag indicates if the NAS Synchronization Indicator is included, if so, it shall be passed to the layer above. If this flag is set to FALSE, then the parameter *thp* (see clause 6.3.6.1) is included instead.

The parameter *rab-identity* is of type RABIdentity which is defined as follows:

```
RABIdentity ::=
    BIT STRING (SIZE(8))
```

The parameter *nas-synchronization-indicator* is of type NASSynchronisationIndicator which is defined as follows:

```
NASSynchronisationIndicator ::=
    INTEGER (0..15)
```

The *ext-rab-info-included* flag indicates if the extension information is present. If TRUE then the content of the ext-rab-info field is present, the first two bits of which indicate the *rab-info-type*. Only one construct is currently defined for *rab-info*, which contains an RABAccessPriority information element used to control access to random access channels and to control transmit assembly process for LDR bearers. This construct is defined below with structure as shown in Figure 6.52.

```
RabInfoType ::= INTEGER {
    reserved (0),
    rab-access-priority-info-type (1)
} (0..3)
```

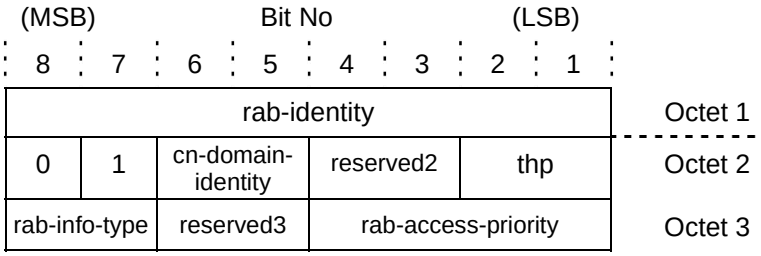


Figure 6.52: RABInfo Parameter Encoding (RabInfoExtension present specifying *rab-access-priority*)

6.3.6.1 TrafficHandlingPriority

The parameter *thp* is of type TrafficHandlingPriority which is defined as follows:

```
TrafficHandlingPriority ::=
    INTEGER {
        traffic-handling-priority-15-or-background (0),
        traffic-handling-priority-1 (1),
```

```
        traffic-handling-priority-2 (2),
        traffic-handling-priority-3 (3)
    } (0..3)
```

The parameter value reflects the Traffic Handling Priority received with the RANAP (Radio Access Network Application Part) RAB AssignmentRequest message if the Core Network requested a RAB setup for an Interactive Class connection. If a Background Class connection was requested by the Core Network then this shall be signalled to the UE as *traffic-handling-priority-15-or-background.*

6.3.6.2 RabAccessPriority

The parameter *rab-access-priority* is of type RabAccessPriority which is defined as follows:

```
RabAccessPriority ::=
    INTEGER (0..15)
```

The parameter is specified by the RNC for this RAB and is used by the UE to determine the radio resources that are available for use and for assembly of transmit bursts. The value of 0 represents the lowest RAB Access Priority. Value 14 is reserved for UE specific signalling, while 15 is reserved for Common Signalling.

6.3.7 ULCipheringActivationTimeInfoParam (ShortAVPType 0x04)

The Activation Time Info value in this AVP is expressed as the BCn Send Sequence Number and specifies when the change in the uplink ciphering occurs. The parameter is defined as below, with structure as shown in Figure 6.53.

```
ULCipheringActivationTimeInfoParam ::=
    SEQUENCE {
        reserved
            BIT STRING (SIZE(6)),
        bcn-send-seq-number
            BcnSendSeqNumber
    }
```

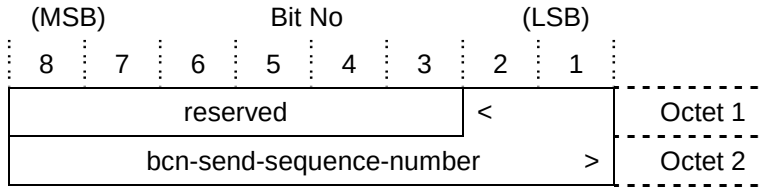


Figure 6.53: ULCipheringActivationTimeInfo Parameter Encoding

The parameter *bcn-send-sequence-number* is of type BcnSendSeqNumber which is defined as follows:

```
BcnSendSeqNumber ::=
    INTEGER (0..1023)
```

This parameter has a fixed length of two octets and the AVP is encoded as a short AVP type.

6.3.8 ULIntegrityProtectionActivationInfoParam (ShortAVPType 0x07)

This parameter is expressed as the current Adaptation Layer (AL) message sequence number when a new integrity protection configuration is activated for the signalling connection. The parameter is defined as below, with structure as shown in Figure 6.54.

```
ULIntegrityProtectionActivationInfoParam ::=
    SEQUENCE {
        reserved
            BIT STRING (SIZE(4)),
        al-msg-seq-no
            ALMsgSeqNumber
    }
```

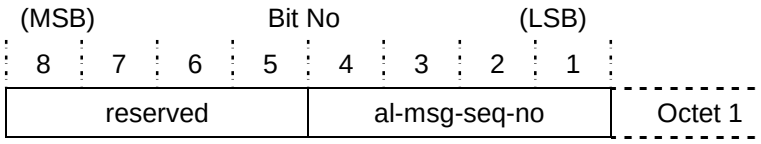


Figure 6.54: ULIntegrityProtectionActionInfo Parameter Encoding

This parameter has a fixed length of one octet and the AVP is encoded as a short AVP type. When it is OPTIONAL and not present, the value of *al-msg-seq-no* can be assumed to be "1".

6.3.9 CipheringModeInfoParam (Short/StandardAVPType 0x08)

6.3.9.0 General

This parameter encapsulates the Ciphering Mode Info IE specified in [3], clause 10.3.3.5. The parameter is defined as below, with structure as shown in Figures 6.55 and 6.56.

```
CipheringModeInfoParam ::=
    SEQUENCE {
        activation-time-for-tm-included
            BOOLEAN,
        reserved
            BIT STRING (SIZE (3)),
        ciphering-mode-command
            CipheringModeCommand,
        tm-activation-time
            SEQUENCE {
                reserved2
                    BIT STRING (SIZE (4)),
                activation-time-for-tm
                    FrameNumber
            } OPTIONAL,
        dl-ciph-activation-time-info-list
            RBActivationTimeInfoList OPTIONAL
    }
```

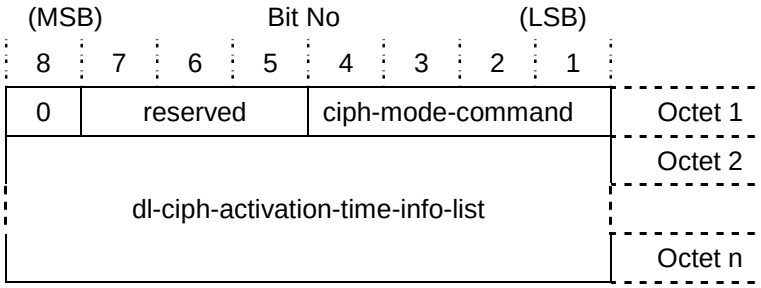


Figure 6.55: CipheringModeInfo Parameter Encoding (*activation-time-for-tm* not included)

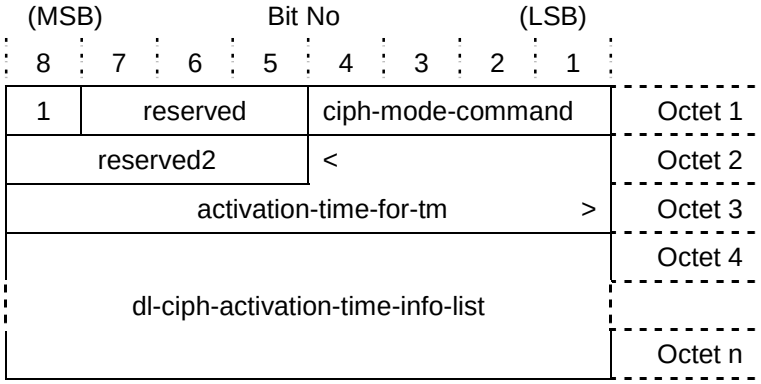


Figure 6.56: CipheringModeInfo Parameter Encoding (*activation-time-for-tm* included)

The BOOLEAN flag *activation-time-for-tm-included* indicates whether the parameter *activation-time-for-tm* is present. This parameter specifies the number of the frame from which the ciphering of TM connections shall start. It is only present if one or more TM connections exist at the time the AVP is sent. The data type FrameNumber is defined in clause 6.3.4.

The CipheringModeInfoParam parameter has a variable length and the AVP is encoded either as a short or standard AVP type, depending on the AVP value length.

6.3.9.1 CipheringModeCommand

The parameter *ciphering-mode-command* specifies the ciphering algorithm to be used and its data type is defined as follows:

```
CipheringModeCommand ::=
    SEQUENCE {
        start-restart
            CipheringAlgorithm
    }
```

with

```
CipheringAlgorithm ::=
    INTEGER {
        uea0(0),
        uea1(1)
    } (0..15)
```

The parameter *dl-ciph-activation-time-info-list* is of type RBActivationTimeInfoList which is defined in the following clause.

6.3.9.2 RBActivationTimeInfoList

The data type RBActivationTimeInfoList contains a list of Translated Bearer Connection IDs and Bearer Connection Sequence Numbers, specifying for each Acknowledged Mode (AM) or Unacknowledged Mode (UM) connection the sequence number from when the specified ciphering shall be applied to the connection.

```
RBActivationTimeInfoList ::=
    SEQUENCE (SIZE (1..maxRB)) OF
        RBActivationTimeInfo
```

The data type RBActivationTimeInfo is defined as follows, with structure as shown in Figures 6.57 and 6.58.

```
RBActivationTimeInfo ::=
    CHOICE {
        pre-0x83-release
            SEQUENCE {
                tbcn-id
                    TranslatedBearerConnectionID,   -- 12 bit
                    reserved
                    BIT STRING (SIZE (2)),
                bcn-send-sequence-number
                    BCnSendSeqNumber            -- 10 bit
            },
        0x83-release
            SEQUENCE {
                bcn-id
                    BearerConnectionID,         -- 24 bit
                    reserved2
                    BIT STRING (SIZE (6)),
                bcn-send-sequence-number
                    BCnSendSeqNumber            -- 10 bit
            }
    }
```

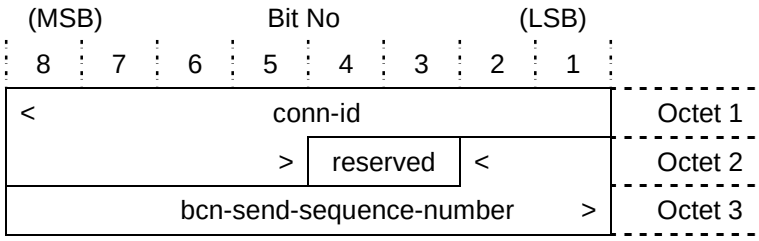


Figure 6.57: RBActivationTimeInfo Encoding (pre-0x83-release)

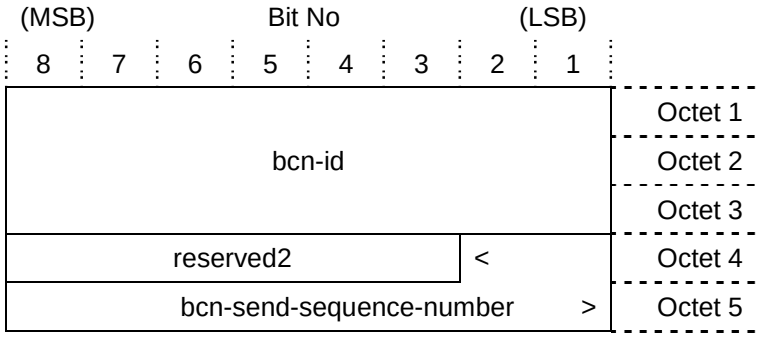


Figure 6.58: RBActivationTimeInfo Encoding (pre-0x83-release)

BcnSendSequenceNumber is defined in clause 6.3.7, BearerConnectionID is defined in clause 6.1.3.1 and TranslatedBearerConnectionID is defined as follows:

```
TranslatedBearerConnectionID ::=
    INTEGER (0..4095)
```

UEs compliant with RI-Version 0x81 or 0x82 (see clause 6.1.2.2) shall use the choice *pre-0x83-release* (using translated Bearer Connection ID to reference the Bearer Connection concerned) while UEs running RI-Version 0x83 and above shall use choice *0x83-release* (using the Bearer Connection ID instead).

6.3.10 IntegrityProtectionModeInfoParam (ShortAVPType 0x0A)

This parameter encapsulates the Integrity Protection Mode Info IE specified in [3], clause 10.3.3.19.

```
IntegrityProtectionModeInfoParam ::=
    SEQUENCE {
        integrity-protection-mode-command
            IntegrityProtectionModeCommand,
        optional-algorithm
            SEQUENCE {
                reserved
                    BIT STRING (SIZE (4)),
                integrity-protection-algorithm
                    IntegrityProtectionAlgorithm
            } OPTIONAL
    }
```

The structure of IntegrityProtectionModeCommand is defined as follows:

```
IntegrityProtectionModeCommand ::=
    CHOICE {
        start
            SEQUENCE {
                integrity-protection-init-number
                    IntegrityProtectionInitNumber
            },
        modify
            SEQUENCE {
                reserved
                    BIT STRING (SIZE (4)),
                dl-integrity-protection-activation-info
                    ALMsgSeqNumber
```

```
        }
    }
```

The data types IntegrityProtectionInitNumber and IntegrityProtectionAlgorithm are defined below. For the definition of the data type ALMsgSeqNumber see clause 6.2.1.5.

```
IntegrityProtectionInitNumber ::=
    BIT STRING (SIZE(32)) - "FRESH" 3GPP TS33.102
IntegrityProtectionAlgorithm ::=
    INTEGER {
        uia1(1)
    } (0..15)
```

The parameter values are packed as shown in Figures 6.59 and 6.60, depending on the information elements contained in the structure.

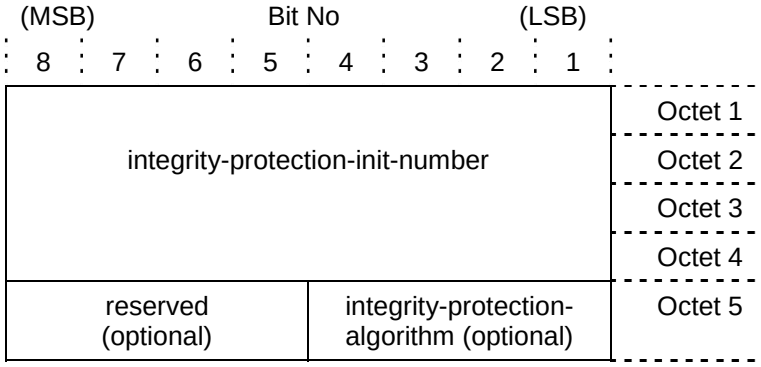


Figure 6.59: IntegrityProtectionModeInfo Parameter Encoding (with *start* and optional *integrity-protection-algorithm* parameters)

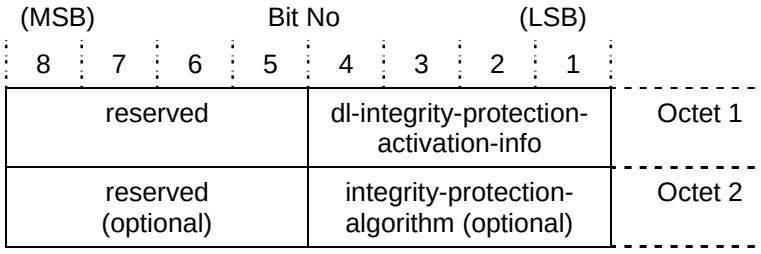


Figure 6.60: IntegrityProtectionModeInfo Parameter Encoding (with *modify* and optional *integrity-protection-algorithm* parameters)

This parameter has a variable length of one, two, four or five octets and the AVP is encoded as a short AVP type.

6.3.11 PDCPInfoParam (Short/StandardAVPType0x0C)

6.3.11.0 General

This parameter encapsulates the PDCP Info IE specified in [3], clause 10.3.4.2. The parameter is defined as follows, with structure as shown in Figure 6.61.

```
PDCPInfoParam ::=
    SEQUENCE {
        lossless-srns-reloc-supported
            BOOLEAN,
        max-pdcp-sn-window-size
            MaxPDCPSNWindowSize,
        pdcp-pdu-header
            PDCPPDUHeader,
        header-compression-info-list-length
            INTEGER (0..31),
        header-compression-info-list
            HeaderCompressionInfoList OPTIONAL
    }
```

The BOOLEAN flag *lossless-srns-reloc-supported* (bit 8 of octet 1 labelled "lsrs" in Figure 6.54) indicates whether the parameters *max-pdcp-sn-window-size* (bit 7 of octet 1 labelled "ms" in Figure 6.54) shall be evaluated. The data type MaxPDCPSNWindowSize is defined as follows:

```
MaxPDCPSNWindowSize ::=
    ENUMERATED {
        sn255(0),
        sn65535(1)
    }
```

The parameter *pdcp-pdu-header* (bit 6 of octet 1 labelled "pph" in Figure 6.54) is of type PDCPPDUHeader which is defined as follows:

```
PDCPPDUHeader ::=
    ENUMERATED {
        present(0),
        absent(1)
    }
```

The parameter *header-compression-info-list-length* defines the length of the *header-compression-info-list*, expressed as the number of information elements of type HeaderCompressionInfo, which follows the first octet of the structure as shown in Figure 6.54. The parameter type HeaderCompressionInfoList is defined as a sequence of elements of type HeaderCompressionInfo which contain algorithm specific info.

```
HeaderCompressionInfoList ::=
    SEQUENCE (SIZE (1..maxPDCPAlgoType)) OF
        HeaderCompressionInfo
HeaderCompressionInfo::=
    SEQUENCE {
        algorithm-specific-info
            AlgorithmSpecificInfo
    }
AlgorithmSpecificInfo ::=
    CHOICE {
        rfc2507-info
            RFC2507Info,
        rfc3095-info
            RFC3095Info
    }
```

The parameters *rfc2507-info* and *rfc3095-info* are defined in the following clauses.

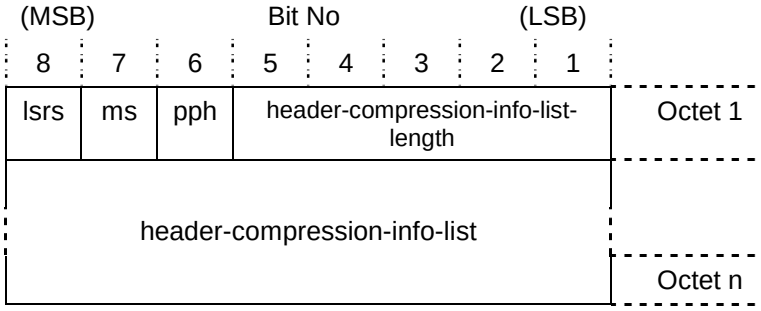


Figure 6.61: PDCPInfoParam Encoding

This parameter has a variable length and the AVP is encoded either as a short or standard AVP type, depending on the AVP value length.

6.3.11.1 RFC2507Info

The parameter *rfc2507-info* carries the header compression info for the RFC 2507 [i.2] compression algorithm. The data type RFC2507Info is defined as follows:

```
RFC2507Info ::=
    SEQUENCE {
        algorith-type
            AlgorithmType,
            -- encode as 0: rfc-2507
        expect-reordering
            ExpectReordering,
        f-max-period-included
```

```
            BOOLEAN,
        f-max-time-included
            BOOLEAN,
        max-header-included
            BOOLEAN,
        tcp-space-included
            BOOLEAN,
        non-tcp-space-included
            BOOLEAN,
        reserved
            BIT STRING (SIZE (2)),
        f-max-period
            INTEGER (0..65535) (CONSTRAINED BY
                {-- expect (1..65535) --}) DEFAULT 256,
        f-max-time
            INTEGER (0..255) (CONSTRAINED BY
                {-- expect (1..255) --}) DEFAULT 5,
        max-header
            INTEGER (0..65535) (CONSTRAINED BY
                {-- expect (60..65535) --}) DEFAULT 168,
        tcp-space
            INTEGER (0..255) (CONSTRAINED BY
                {-- expect (3..255) --}) DEFAULT 15,
        non-tcp-space
            INTEGER (0..65535) (CONSTRAINED BY
                {-- expect (3..65535) --}) DEFAULT 15
    }
```

The parameter *algorithm-type* is of type AlgorithmType which is defined as follows:

```
AlgorithmType ::=
    INTEGER {
        rfc-2507 (0),
        rfc-3095 (1)
} (0..255)
```

The parameter *expect-reordering* is of type ExpectReordering which is defined as follows:

```
ExpectReordering ::=
    ENUMERATED {
        reordering-not-expected(0),
        reordering-expected(1)
    }
```

The BOOLEAN parameters *f-max-period-included*, *f-max-time-included*, *tcp-space-included*, *max-header-included* and *non-tcp-space-included* indicate whether the respective parameters are included in the structure or whether the specified default value applies. These bit fields are abbreviated in Figure 6.62 as fp, ft, ts, mh and nts respectively.

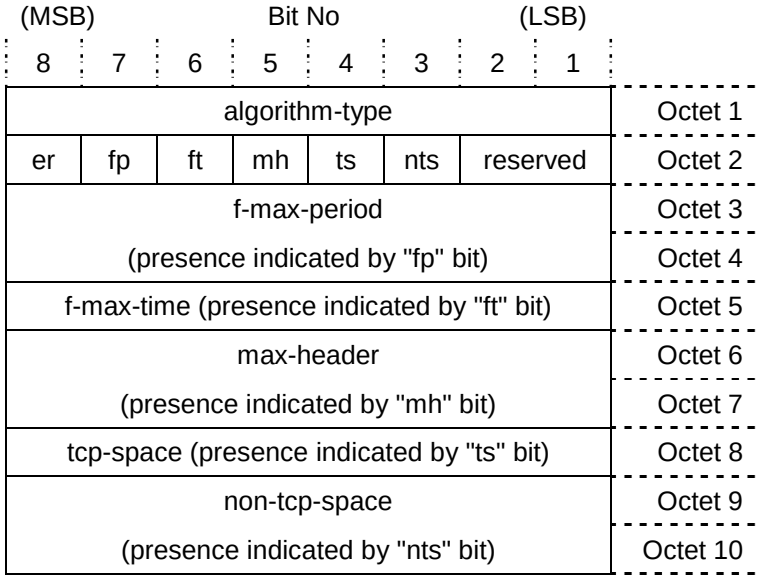


Figure 6.62: RFC2507Info parameter encoding

6.3.11.2 RFC3095Info

6.3.11.2.0 General

The parameter *rfc3095-info* carries the header compression info for the RFC 3095 [i.3] compression algorithm. The data type RFC3095Info is defined as follows, with structure as shown in Figure 6.63.

```
RFC3095Info ::=
    SEQUENCE {
        algorithm-type
            AlgorithmType,
        -- encode as 1: rfc-3095
        uplink-data-present
            BOOLEAN,
        downlink-data-present
            BOOLEAN,
        reserved
            BIT STRING (SIZE (1)),
        rohc-profile-list-length
            INTEGER (0..31),
            -- values>=17 not used
        rohc-profile-list
            ROHCProfileList,
        uplink-rohc-data
                UplinkROHCData OPTIONAL,
        downlink-rohc-data
                DownlinkROHCData OPTIONAL
    }
```

The parameter *algorithm-type* is defined in clause 6.3.11.1. The BOOLEAN parameter *uplink-data-present* indicates whether the parameter *uplink-rohc-data* of type UplinkROHCData is included in the structure. This bit field is abbreviated in Figure 6.63 as udp. The BOOLEAN parameter *downlink-data-present* indicates whether the parameter *downlink-rohc-data* of type DownlinkROHCData is included in the structure. This bit field is abbreviated in Figure 6.63 as ddp. The data types UplinkROHCData and DownlinkROHCData are specified in clauses 6.3.11.2.1 and 6.3.11.2.2.

The parameter *rohc-profile-list-length* indicates the numbers of items in the *rohc-profile-list.* If *rohc-profile-list-length* is of value zero this indicates that only profile number zero (0x0000) is supported and *rohc-profile-list* will be absent. A minimum of one and a maximum of 16 items of type ROHCProfile as defined below are possible in the list:

```
ROHCProfileList ::=
    SEQUENCE {
        profile-list-elements
            SEQUENCE (SIZE (1..maxROHCProfile)) OF
                ROHCProfile,
        padding             -- to align with octet boundary
            SEQUENCE (SIZE (0..1)) OF
                BIT STRING (SIZE (4))
    }
```

with:

```
ROHCProfile ::=
    INTEGER (1..16)
```

The *rohc-profile-list* will therefore always occupy an integer number of octets.

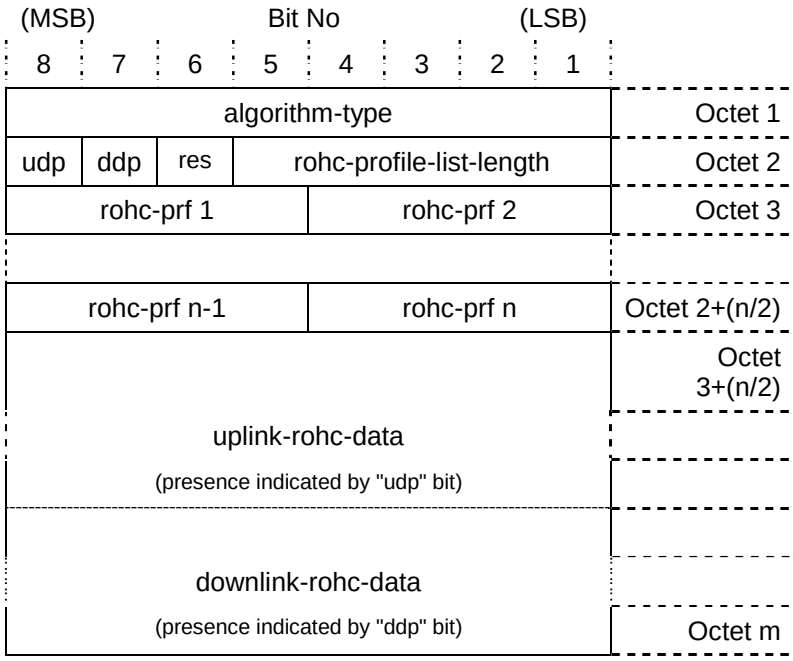


Figure 6.63: RFC3095Info Parameter Encoding

6.3.11.2.1 UplinkROHCData

The parameter *uplink-rohc-data* carries the necessary information elements for configuring the RFC 3095 Robust Header Compression (ROHC) algorithm [i.3] for compression of uplink traffic at the UE. The data type `UplinkROHCData` is defined as follows, with structure as shown in Figure 6.64.

```
UplinkROHCData ::=
    SEQUENCE {
        uplink-cid-inclusion-info
            CIDInclusionInfo,
        uplink-max-cid-included
            BOOLEAN,
        rohc-packet-size-list-length
            INTEGER (0..63),
            -- values>=17 not used
        uplink-max-cid
            INTEGER (1..16383) DEFAULT 15,
            -- encode as 16 bit integer
        rohc-packet-size-list
            ROHCPacketSizeList OPTIONAL
    }
```

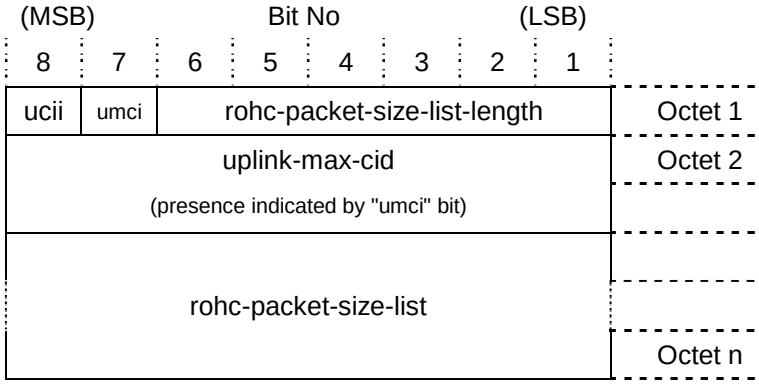


Figure 6.64: UplinkROHCData Parameter Encoding

The parameter *uplink-cid-inclusion-info* is of type CIDInclusionInfo which is defined as follows:

```
CIDInclusionInfo ::=
    ENUMERATED {
        pdcp-header(0),
        rfc3095-packet-format(1)
    }
```

This bit field is abbreviated in Figure 6.64 as ucii.

The BOOLEAN parameters *uplink-max-cid-included* indicates whether the parameter *uplink-max-cid* is included in the structure or whether the specified default value applies. These fields are abbreviated in Figure 6.64 as umci.

The parameter *rohc-packet-size-list-length* indicates the number of items in the *rohc-packet-size-list*. If the *rohc-packet-size-list-length* is of value zero this indicates that the ROHC packet size option is not selected and that *rohc-packet-size-list* is not included in the structure.

The *rohc-packet-size-list* is of type ROHCPacketSizeList which is defined as follows:

```
ROHCPacketSizeList ::=
    SEQUENCE (SIZE (1..maxROHCPacketSizes)) OF
        ROHCPacketSize
```

A maximum of 16 items of type ROHCPacketSize as defined below are possible in the list:

```
ROHCPacketSize ::=
    INTEGER (2..1500)
-- encode as 16 bit integer
```

6.3.11.2.2 DownlinkROHCData

The parameter *downlink-rohc-data* carries the necessary information elements for configuring the RFC 3095 ROHC algorithm [i.3] for decompression of downlink traffic at the UE. The data type DownlinkROHCData is defined as follows, with structure as shown in Figure 6.65.

```
DownlinkROHCData ::=
    SEQUENCE {
        downlink-cid-inclusion-info
            CIDInclusionInfo,
        downlink-max-cid-included
            BOOLEAN,
        reverse-decompression-depth-included
            BOOLEAN,
        reserved
            BIT STRING (SIZE (5)),
        downlink-max-cid
            INTEGER (1..16383) DEFAULT 15,
            -- encode as 16 bit integer
        reverse-decompression-depth
            INTEGER (0..65535) DEFAULT 0
}
```

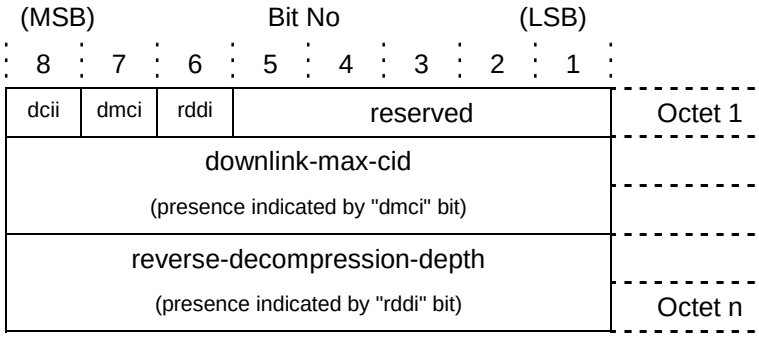


Figure 6.65: DownlinkROHCData Parameter Encoding

The parameter *downlink-cid-inclusion-info* is of type CIDInclusionInfo which is defined in clause 6.3.11.2.1. This bit field is abbreviated in Figure 6.65 as dcii.

The BOOLEAN parameter *downlink-max-cid-included* indicates whether the parameter *downlink-max-cid* is included in the structure or whether the specified default value applies. This bit field is abbreviated in Figure 6.65 as mcii.

The BOOLEAN parameter *reverse-decompression-depth-included* indicates whether the parameter *reverse-decompression-depth* is included in the structure or whether the specified default value applies. This bit field is abbreviated in Figure 6.65 as rddi.

6.3.12 CSCallTypeParam (ParamType 0x05)

This parameter has a length of one octet and is used to signal the Circuit Switched Call Type for the Circuit Switched User Plane Handler (CSH) from the UE to the RNC. The parameter is defined as follows, with structure as shown in Figure 6.66.

```
CSCallTypeParam ::=
    SEQUENCE
        reserved
            BIT STRING (SIZE (4)),
        cs-call-type
            INTEGER {
                type-4kbits-speech (0),
                type-3pt1khz-audio (1),
                type-udi-isdn (2),
                type-rdi-isdn (3)
            } (0..15)
    }
```

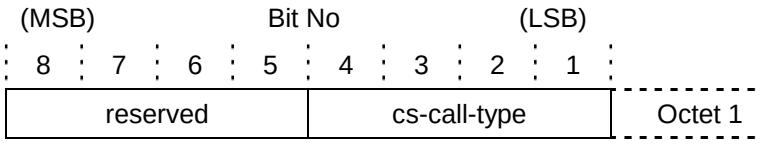


Figure 6.66: CSCallType Parameter Encoding

6.3.13 GroupCipherInfoParam (StandardAVPType 0x0D)

This parameter encapsulates the Group Ciphering Key and Count-C value for generation of the group bearer connection specific decipher keystream block information in the UE.

```
GroupCipherInfoParam ::=    BIT STRING (SIZE (128))
```

The parameter value is structured as shown in Figure 6.67.

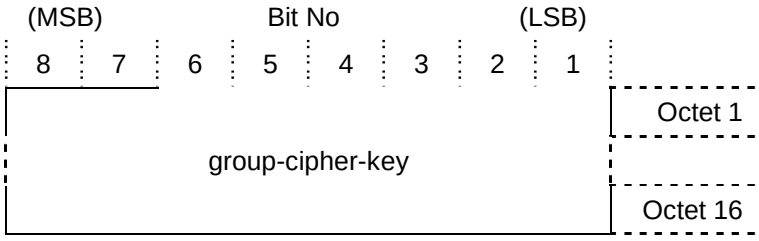


Figure 6.67: GroupCipherInfo Parameter Encoding

Note that the Count-C value for generation of ciphering keys (CKs) is distributed within the embedded BCtPDU. The value of group-cipher-key is used within the generation of CK for the downlink portion of Multimedia Broadcast Multicast Services (MBMS) connections, and the application of this and the transmitted Count-C value is the same as unicast Unacknowledged-mode (ie numbered) connections.

6.4 Connection Layer AVP

6.4.0 General

Connection Sub-Layer AVPs are used to transfer parameters relating to the configuration of the Connection Sub-Layer peers.

6.4.1 BCn-AVP Structure

6.4.1.0 General

The structure of a Connection Sub-Layer AVP is as follows:

```
BCnAVP ::=
    SEQUENCE {
        bcn-avp-type
            BCnAVPType,
        param-value
            CHOICE {
            -- as appropriate to value of
            -- bcn-avp-type
                response-time-param
                    ResponseTimeParam,
                max-idle-time-param
                    MaxIdleTimeParam,
                max-connection-time-param
                    MaxConnectionTimeParam,
                tx-window-size-param
                    TxWindowSizeParam,
                tx-buffer-size-param
                    TxBufferSizeParam,
                csh-configuration-param
                    CSHConfigurationParam,
                adaptation-layer-avp-list-length-param
                    AdaptationLayerAVPListLengthParam
            }
    }
```

Each parameter value has a maximum length of eight bytes. *Bcn-avp-type* determines the type of parameter. The data type BCnAVPType is defined as follows:

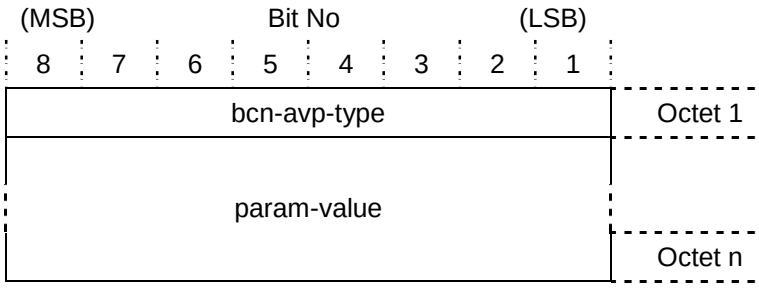
```
BCnAVPType ::=
    INTEGER {
        response-time(8),
        max-idle-time(9),
        max-connection-time(17),
        tx-window-size(33),
        tx-buffer-size(41),
        csh-configuration(48),
        al-avp-list-length(248)
    } (0..255)
```

The values are allocates such that the parameter length can be obtained from the lower three bits of *bcn-avp-type*. Hence the definition of BCnAVPType above is equivalent to the following:

NOTE: This structure is shown in the text for explanatory purposes and is not included in annex A.

```
BCnAVPType ::=
    SEQUENCE {
        prm-len-type
            INTEGER (0..31),    -- PrmLenType
        prm-len
            INTEGER (1..8)      -- PrmLen
    }
```

Note that the parameter type is defined independently for each *prm-len* value. This results in a possible 32 parameter types for each parameter-length value. The resulting AVP structure is as shown below in Figure 6.68.



equivalent to:

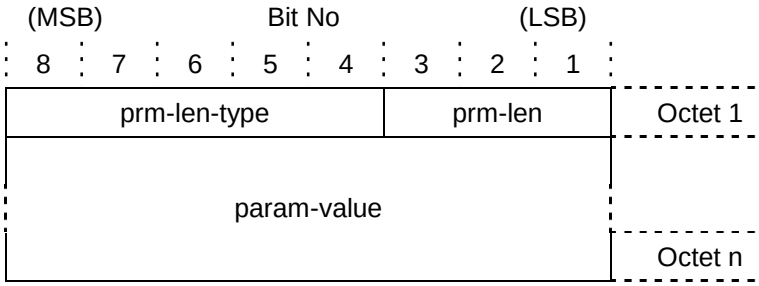


Figure 6.68: BCnAVP Structure

Table 6.8 lists the BCnAVP types and specifies in which Adaptation Layer PDU they may be contained.

Table 6.8: BCn-AVPs supported in Adaptation Layer PDUs
(MP: mandatory presence, OP: optional presence, n/a: not applicable)

BcnParam Type	Length	Param-Value	Register-Ack	Establish	Modify
0x08	1	ResponseTimeParam	OP	OP	OP
0x09	2	MaxIdleTimeParam	OP	OP	OP
0x11	2	MaxConnectionTimeParam	OP	OP	OP
0x21	2	TxWindowSizeParam	n/a	OP	n/a
0x29	2	TxBufferSizeParam	n/a	OP	n/a
0x30	1	CSHConfigurationParam	n/a	MP if CS	n/a
0xF8	1	ALAVPListLengthParam	n/a	OP	OP

The *param-type* value of 0xF8 (*prm-len-type* = 31) shall be used to signal the expansion of the BCn-AVP list with a list of Adaptation Layer AVPs (see clause 6.3) and shall always be the last item in the BCn-AVP-List. This AVP is described in clause 6.4.7.

6.4.1.1 PrmLen

This three bit field specifies the length of the parameter value in the AVP. This field forms the least significant three bits of the *param-type* field, and contains the one less than the length of the *param-value* field. Note that this implies that a zero-length parameter length is not supported.

6.4.1.2 PrmLenType

This five bit field indicates the type of the parameter within the context of the *prm-len* field value. This field forms the most significant five bits of the *param-type* field.

6.4.2 ResponseTimeParam (ParamType 0x08)

This parameter has a length of one octet and is used to define the response timer value for the connection which is being established or modified. This time is used by the ARQ timer to generate a retransmission of a Receive Ready message in the event that no response was received from the peer.

```
ResponseTimeParam ::=
    INTEGER (0..255)
```

The value represents the period of the response timer in 40 ms units. The default setting for the Response-Time is 25, corresponding to 1,0 seconds.

6.4.3 MaxIdleTimeParam (ParamType 0x09)

This parameter has a length of two octets (*prm-len* = 1) and is used to define the period of inactivity for the connection which is being established or modified, before the connection shall be automatically released. A period of inactivity is defined as the period over which no data is successfully transferred (and acknowledged if the connection is operating in Acknowledged Mode) in either direction.

```
MaxIdleTimeParam ::=
    INTEGER (0..65535)
```

The unit of *max-idle-time* is in seconds. If this AVP is not transferred to the UE then the connection may be held for a default of 60 seconds.

6.4.4 MaxConnectionTimeParam (ParamType 0x11)

This parameter has a length of two octets (*prm-len* = 1) and is used to define the maximum period that the connection which is being established or modified, shall be allowed to operate before the connection is automatically released.

```
MaxConnectionTimeParam ::=
    INTEGER (0..65535)
```

The unit of *max-connection-time* is 10 seconds. If this AVP is not transferred to the UE then the connection may operate indefinitely.

6.4.5 TxWindowSizeParam (ParamType 0x21)

This parameter has a length of two octets (*prm-len* = 1) and is used define the transmit window size to utilize at the UE for the connection that is being established.

```
TxWindowSizeParam ::=
    INTEGER (0..511);
    -- encoded within a 16 bit field
```

The units are segments. The default value for *tx-window-size* is the maximum value of 511, corresponding to a transmit window size of 511 segments (corresponding to the sequence number range divided by two minus one).

6.4.6 TxBufferSizeParam (ParamType 0x29)

This parameter has a length of two octets and is used to define the transmit buffer size to utilize at the UE for the connection that is being established, before flow control is asserted on external interfaces.

```
TxBufferSize ::=
    INTEGER (0..65535);
```

The value is specified in a unit size of 256 bytes. The default value for *tx-buffer-size* is 256, corresponding to a transmit buffer size of 65 536 bytes.

6.4.7 AdaptationLayerAVPListLengthParam (ParamType 0xF8)

This parameter has a length of one octet and is used to signal the number of Adaptation Layer AVPs which follow this AVP.

```
AdaptationLayerAVPListLengthParam ::=
    INTEGER (0..255);
```

The contents of the Adaptation Layer AVP List are specified in clause 6.3.

6.4.8 CSHConfigurationParam (ParamType 0x30)

This parameter has a length of one octet and is used to signal the configuration parameters for the Circuit Switched User Plane Handler (CSH). The parameter is defined as follows, with structure as shown in Figure 6.69.

```
CSHConfigurationParam ::=
    SEQUENCE {
        fwd-dtx
            BOOLEAN,
        fwd-cs-frames-per-pdu
            INTEGER (1..8),
        ret-dtx
            BOOLEAN,
        ret-cs-frames-per-pdu
            INTEGER (1..8)
    }
```

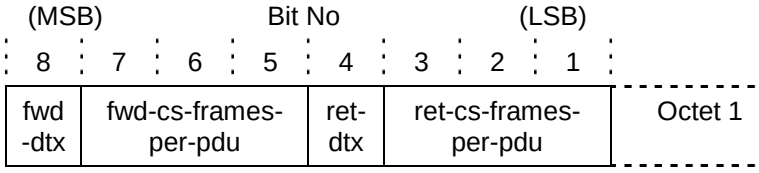


Figure 6.69: CSHConfiguration Parameter Encoding

The parameters *fwd-dtx* and *ret-dtx* define whether the discontinuous transmission (DTX) mode shall be used in the forward and return direction respectively. The parameters *fwd-cs-frames-per-pdu* and *fwd-cs-frames-per-pdu* specify the number of circuit switched payload frames (e.g. voice codec frames) per Transparent Mode Bearer Connection PDU in the forward and return direction respectively.

Annex A (normative): ASN.1

This annex collates the data structures in ASN.1 notation from the present document in alphabetical order, in a format that may be used in a program code compiler.

The code is reproduced in a text file that is contained in archive ts_1027440305v010101p0.zip which accompanies the present document.

History

Document history		
V1.1.1	October 2015	Publication