ERICSSON

Release 98 CDR ASN.1 Format

PRINTOUT DESCRIPTION

11/190 83-AXB 250 12-V3 Uen AG

Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

Trademark List

All trademarks mentioned herein are the property of their respective owners. These are shown in the document Trademark Information.

Contents

1 Introduction

This document presents the ASN.1 format for release 98 CDRs generated by the EPG.

For more information on CDRs, see CDR Format for the GGSN and PGW and CDR Format for the SGW.

2 ASN.1 Format

```
GgsnR98Ber DEFINITIONS IMPLICIT TAGS ::= BEGIN

CallEventRecord ::= CHOICE
{
  ggsnPDPRecord [1] GGSNPDPRecord
}

GGSNPDPRecord ::= SET
{
  recordType           [0] CallEventRecordType,
  servedIMSI           [3] IMSI,
  ggsnAddress          [4] GSNAddress,
  chargingID           [5] ChargingID,
  sgsnAddress          [6] SEQUENCE OF GSNAddress OPTIONAL,
  accessPointNameNI    [7] AccessPointNameNI OPTIONAL,
  pdpType              [8] PDPType OPTIONAL,
  servedPDPAddress     [9] PDPAddress OPTIONAL,
  dynamicAddressFlag   [11] DynamicAddressFlag OPTIONAL,
  listOfTrafficVolumes [12] SEQUENCE OF ChangeOfCharCondition,
  recordOpeningTime    [13] TimeStamp,
  duration             [14] CallDuration,
  causeForRecClosing   [15] CauseForRecClosing,
  recordSequenceNumber [17] INTEGER OPTIONAL,
  nodeID               [18] NodeID OPTIONAL,
  recordExtensions     [19] ManagementExtensions OPTIONAL,
  localSequenceNumber  [20] LocalSequenceNumber OPTIONAL,
  apnSelectionMode     [21] APNSelectionMode OPTIONAL,
  servedMSISDN         [22] MSISDN OPTIONAL,
  sgsnPLMNIdentifier   [27] PLMN-Id OPTIONAL
}


AccessPointNameNI ::= IA5String (SIZE(1..63))
AddressString ::= OCTET STRING (SIZE (1..20))
APNSelectionMode ::= ENUMERATED
{
  mSorNetworkProvidedSubscriptionVerified (0),
  mSProvidedSubscriptionNotVerified       (1),
  networkProvidedSubscriptionNotVerified  (2)
}
CallDuration ::= INTEGER
CallEventRecordType ::= INTEGER
{
  ggsnPDPRecord (19)
}
CauseForRecClosing ::= INTEGER
```

```
{
  normalRelease            (0),
  abnormalRelease          (4),
  volumeLimit              (16),
  timeLimit                (17),
  sGSNChange               (18),
  maxChangeCond            (19),
  managementInitRelease    (100),
  pLMNChange               (101),
  creditControlChange      (102),
  creditControlInitRelease (104),
  policyControlInitRelease (105)
}
ChangeCondition ::= ENUMERATED
{
  qoSChange               (0),
  tariffTime              (1),
  recordClosure           (2)
}
ChangeOfCharCondition ::= SEQUENCE
{
  qosNegotiated          [2] QoSInformation OPTIONAL,
  dataVolumeGPRSUplink   [3] DataVolumeGPRS,
  dataVolumeGPRSDownlink [4] DataVolumeGPRS,
  changeCondition        [5] ChangeCondition,
  changeTime             [6] TimeStamp
}
ChargingID ::= INTEGER (0..4294967295)
DataVolumeGPRS ::= INTEGER
DynamicAddressFlag ::= BOOLEAN
ETSIAddress ::= AddressString
GSNAddress ::= IPAddress
IMSI ::= TBCD-STRING (SIZE (3..8))
IPAddress ::= CHOICE
{
  iPBinaryAddress          IPBinaryAddress,
  iPTextRepresentedAddress IPTextRepresentedAddress
}
IPBinaryAddress ::= CHOICE
{
  iPBinV4Address [0] OCTET STRING (SIZE(4)),
  iPBinV6Address [1] OCTET STRING (SIZE(16))
}
IPTextRepresentedAddress ::= CHOICE
{
  iPTextV4Address [2] IA5String (SIZE(7..15)),
  iPTextV6Address [3] IA5String (SIZE(15..45))
}
ISDN-AddressString ::= AddressString (SIZE(1..9))
LocalSequenceNumber ::= INTEGER (0..4294967295)
ManagementExtensions ::= SET OF ManagementExtension
```

```
ManagementExtension ::= SEQUENCE
{
  identifier       OBJECT IDENTIFIER,
  significance [1] BOOLEAN DEFAULT TRUE,
  information  [2] GprsCdrExtensions
}
MSISDN ::= ISDN-AddressString
NodeID ::= IA5String (SIZE(1..20))
PDPAddress ::= CHOICE
{
  iPAddress   [0] IPAddress,
  eTSIAddress [1] ETSIAddress
}
PDPType ::= OCTET STRING (SIZE(2))
PLMN-Id ::= OCTET STRING (SIZE(3))
QoSDelay ::= ENUMERATED
{
  delayClass1 (1),
  delayClass2 (2),
  delayClass3 (3),
  delayClass4 (4)
}
QoSInformation ::=SEQUENCE
{
  reliability    [0] QoSReliability,
  delay          [1] QoSDelay,
  precedence     [2] QoSPrecedence,
  peakThroughput [3] QoSPeakThroughput,
  meanThroughput [4] QoSMeanThroughput
}
QoSMeanThroughput ::= ENUMERATED
{
  bestEffort          (0),
  mean100octetPh      (1),
  mean200octetPh      (2),
  mean500octetPh      (3),
  mean1000octetPh     (4),
  mean2000octetPh     (5),
  mean5000octetPh     (6),
  mean10000octetPh    (7),
  mean20000octetPh    (8),
  mean50000octetPh    (9),
  mean100000octetPh   (10),
  mean200000octetPh   (11),
  mean500000octetPh   (12),
  mean1000000octetPh  (13),
  mean2000000octetPh  (14),
  mean5000000octetPh  (15),
  mean10000000octetPh (16),
  mean20000000octetPh (17),
  mean50000000octetPh (18)
```

```
}
QoSPeakThroughput ::= ENUMERATED
{
  unspecified       (0),
  upTo1000octetPs   (1),
  upTo2000octetPs   (2),
  upTo4000octetPs   (3),
  upTo8000octetPs   (4),
  upTo16000octetPs  (5),
  upTo32000octetPs  (6),
  upTo64000octetPs  (7),
  upTo128000octetPs (8),
  upTo256000octetPs (9)
}
QoSPrecedence ::= ENUMERATED
{
  unspecified    (0),
  highPriority   (1),
  normalPriority (2),
  lowPriority    (3)
}
QoSReliability ::= ENUMERATED
{
  unspecifiedReliability  (0),
  acknowledgedGTP         (1),
  unackGTPAcknowLLC       (2),
  unackGTPLLCAcknowRLC    (3),
  unackGTPLLCRLC          (4),
  unacknowUnprotectedData (5)
}
TBCD-STRING ::= OCTET STRING
TimeStamp ::= OCTET STRING (SIZE(9))

GprsCdrExtensions ::= SET
{
  chargingContainers [1] SEQUENCE OF ChargingContainer OPTIONAL,
  creditControlInfo  [2] CreditControlInfo OPTIONAL,
  policyControlInfo  [3] PolicyControlInfo OPTIONAL,
  userCategory       [5] INTEGER OPTIONAL
}

BlockItem ::= SEQUENCE
{
  blocks        [0] INTEGER,
  blockRate     [1] RealUnit OPTIONAL,
  blockSize     [2] INTEGER,
  blockSizeType [3] BlockSizeType,
  startTime     [4] TimeStamp OPTIONAL,
  stopTime      [5] TimeStamp OPTIONAL
}
BlockSizeType ::= ENUMERATED
```

```
{
  seconds (0),
  bytes   (1)
}
ChargingContainer ::= SEQUENCE
{
  changeTime           [0] TimeStamp,
  listOfServices       [1] SEQUENCE OF ServiceItem OPTIONAL,
  listOfServiceClasses [2] SEQUENCE OF ServiceClassItem OPTIONAL,
  initialCharge        [3] RealUnit OPTIONAL,
  sessionPolicy        [4] SessionPolicy OPTIONAL,
  blockItem            [5] BlockItem OPTIONAL
}
CreditControlDiagnostics ::= ENUMERATED
{
  creditDenied           (0),
  creditControlFailure   (1),
  creditControlSuspended (2),
  creditControlResumed   (3)
}
CreditControlFailureAction ::= ENUMERATED
{
  release              (0),
  continue             (1),
  continueFreeServices (2)
}
CreditControlFailureReport ::= SEQUENCE
{
  requestType       [0] CreditRequestType,
  requestStatus     [1] CreditRequestStatus,
  resultCode        [2] CreditResultCode OPTIONAL,
  recordNumber      [4] CreditRecordNumber OPTIONAL,
  startTime         [5] TimeStamp OPTIONAL,
  stopTime          [6] TimeStamp OPTIONAL,
  lastCreditGranted [7] RealUnit OPTIONAL,
  creditUsed        [8] RealUnit OPTIONAL
}
CreditControlInfo ::= SEQUENCE
{
  creditControlEnabled       [0] BOOLEAN,
  ccsAddress                 [1] IPAddress OPTIONAL,
  creditUnit                 [2] CreditUnit OPTIONAL,
  creditControlDiagnostics   [3] CreditControlDiagnostics OPTIONAL,
  creditDeniedAction         [4] CreditDeniedAction OPTIONAL,
  creditControlFailureAction [5] CreditControlFailureAction OPTIONAL,
  creditControlFailureReport [6] CreditControlFailureReport OPTIONAL,
  creditControlSessionId     [7] OCTET STRING (SIZE(1..255)) OPTIONAL
}
CreditDeniedAction ::= ENUMERATED
{
  release                     (0),
```

```
  continue                    (1),
  continueFreeServiceClasses  (2),
  continueLimitServiceClasses (3)
}
CreditRecordNumber ::= INTEGER(0..4294967295)
CreditRequestType ::= ENUMERATED
{
  start   (0),
  interim (1),
  stop    (2)
}
CreditRequestStatus ::= ENUMERATED
{
  unsent   (0),
  noAnswer (1),
  failure  (2)
}
CreditResultCode ::= INTEGER
CreditUnit ::= SEQUENCE
{
  unitType     [0] CreditUnitType,
  currencyCode [1] CurrencyCode OPTIONAL
}
CreditUnitType ::= ENUMERATED
{
  time   (0),
  volume (1),
  event  (2),
  money  (3)
}
CurrencyCode ::= INTEGER
PolicyControlDiagnostics ::= ENUMERATED
{
  policyControlFailure   (1),
  policyControlSuspended (2),
  policyControlResumed   (3)
}
PolicyControlFailureAction ::= ENUMERATED
{
  release               (0),
  continueDefaultPolicy (1)
}
PolicyControlFailureReport ::= SEQUENCE
{
  requestType   [0] PolicyRequestType,
  requestStatus [1] PolicyRequestStatus,
  resultCode    [2] PolicyResultCode OPTIONAL,
  sessionId     [3] OCTET STRING (SIZE(1..255)) OPTIONAL,
  startTime     [4] TimeStamp OPTIONAL,
  stopTime      [5] TimeStamp OPTIONAL
}
```

```
PolicyControlInfo ::= SEQUENCE
{
  policyControlEnabled       [0] BOOLEAN,
  pcsAddress                 [1] IPAddress OPTIONAL,
  policyControlDiagnostics   [2] PolicyControlDiagnostics OPTIONAL,
  policyControlFailureAction [3] PolicyControlFailureAction OPTIONAL,
  policyControlFailureReport [4] PolicyControlFailureReport OPTIONAL,
  currencyCode               [5] CurrencyCode
}
PolicyRequestType ::= ENUMERATED
{
  start   (0),
  interim (1),
  stop    (2)
}
PolicyRequestStatus ::= ENUMERATED
{
  unsent      (0),
  noAnswer    (1),
  failure     (2),
  pCRFRestart (3)
}
PolicyResultCode ::= INTEGER
RealUnit ::= SEQUENCE
{
  valueDigits [0] INTEGER,
  exponent    [1] INTEGER OPTIONAL
}
ServiceClass ::= INTEGER(0..4294967295)
ServiceClassItem ::= SEQUENCE
{
  serviceClass       [0] ServiceClass,
  volumeUplink       [1] INTEGER OPTIONAL,
  volumeDownlink     [2] INTEGER OPTIONAL,
  volumeRateUplink   [3] RealUnit OPTIONAL,
  volumeRateDownlink [4] RealUnit OPTIONAL,
  sessionPolicy      [5] SessionPolicy OPTIONAL,
  blockItem          [6] BlockItem OPTIONAL
}
ServiceIdentifier ::= INTEGER(0..4294967295)
ServiceItem ::= SEQUENCE
{
  serviceIdentifier [0] ServiceIdentifier,
  volumeUplink      [1] INTEGER,
  volumeDownlink    [2] INTEGER
}
SessionPolicy ::= SEQUENCE
{
  volumeLimitAction [0] UsageLimitAction OPTIONAL,
  blockLimitAction  [1] UsageLimitAction OPTIONAL
}
```

```
UsageLimitAction ::= ENUMERATED
{
  reset (1),
  add   (2)
}

END
```