Leena Eronen

MEDIAATTORIOHJELMISTO JA SIIHEN LIITTYVÄ PROJEKTITYÖ

Insinöörityö
Kajaanin ammattikorkeakoulu
Tekniikan ja liikenteen ala
Tietotekniikan koulutusohjelma
Syksy 2001

Kajaanin ammattikorkeakoulu

INSINÖÖRITYÖ
TIIVISTELMÄ

Osasto Tekniikka	Koulutusohjelma Tietotekniikka
Tekijä Leena Eronen	
Työn nimi Mediaattoriohjelmisto ja siihen liittyvä projektityö	
Vaihtoehtoiset ammattiopinnot Tietoverkot	Ohjaajat Jukka Heino AMK Arto Karjalainen Ebsolut Oy
Aika 14.09.2001	Sivumäärä 59+8

Tiivistelmä

Insinöörityön tavoitteena oli saada kokonaisvaltainen käsitys Comptel Corporationin mediaattoriohjelmistosta (MDS). Insinöörityö sisältää yleisen GSM-tekniikan ja mediaattoriohjelmisto-osuuden lisäksi projektityön. Projektityön tavoitteena on toimittaa eurooppalaiselle asiakkaalle ohjelmisto, joka käsittelee Ericsson 8.0 keskuksesta saatuja matkapuhelutietoja, ja toimittaa ne teleoperaattorin laskutusjärjestelmään.

Työ toteutettiin Ebsolut Oy:lle, jonka yksi tärkeistä yhteistyökumppaneista on Comptel Corporation. Ebsolut Oy on kajaanilainen kesällä 2000 perustettu yritys. Se on keskittynyt voimakkaasti kasvavan ja kehittyvän tietoliikenteen tietojenkäsittelyn ohjelmistokehitykseen. Comptel on Elisa Communicationsin tytaryhtiö, joka kehittää ja toimittaa mediaattoriohjelmistoja tietoliikenneyrityksille.

Projektityö sisältää ohjelman suunnittelun, toteutuksen, testauksen ja metodin tekemisen. Työn suunnittelu toteutettiin Word –tekstinkäsittelyohjelmalla valmiita mallipohjia käyttäen. Työn toteutus, testaus ja metodit toteutettiin Comptelin unix –kehityskoneella. Ohjelmointikielenä käytettiin S-lang tulkkia.

Projektityötä testattiin yhdessä operaattorin kanssa, ja muutoksia koodiin ja dokumenttiin tehtiin testauksen pohjalta koko projektin aikana. Ohjelmisto on ollut operaattorin tuotantokoneella 10. elokuuta lähtien, ja se on toiminut moitteettomasti. Nopeasti kehittyvän puhelinalan myötä mediaattoriohjelmiston kehitystyö jatkuu ohjelmiston valmistuttua. Kehitystyötä tämän ohjelmiston osalta jatkaa Comptel Corporation, Ebsolut Oy sekä asiakas itse. Pienimuotoinen dokumentti ja koodimuutos on jo tehty elokuun lopussa Ebsolut Oy:ssä.

Luottamuksellinen Kyllä X Ei	
Hakusanat	
Säilytyspaikka	

Kajaanin ammattikorkeakoulu
Kajaani Polytechnic

ABSTRACT
FINAL YEAR PROJECT

Faculty Faculty of Engineering	Degree programme Information Technology
Author Leena Eronen	
Title Mediation Device Solution and Project Work	
Optional professional studies Networks	Instructor(s) Jukka Heino Arto Karjalainen
Date 14.09.2001	Total number of pages 59+8

Abstract

The purpose of this final year project was to provide comprehensive information about the Mediation Device Solution (MDS) of Comptel Corporation. This document contains GSM-engineering, MDS and project work. The purpose of the project work was to implement the MDS-program, which manages the flow of information between the telecommunications network and the European customer service and billing system.

The project was implemented for Ebsolut Oy, which was established in summer 2000 in Kajaani. Comptel is a one consequence of co-operation of Ebsolut Oy. Comptel is a solution provider developing and delivering high quality mediation devices for telecommunications industry.

The project work contained design, implementation, testing and the implementation of the method. The designing work was implemented by the Word word processing program. The implementation, testing and method were implemented in the Unix development machine of Comptel. The programming was implemented by the programming language of S-lang.

The project work was tested by the author and the customer. Modifications of the document and the program were implemented during testing. The MDS-program was installed into the manufacturing machine of the customer on 10 August. The program has worked correctly. Because of ever-increasing flow of call information, developing complete programs continues. Comptel, Ebsolut and the customer will develop this program in future.

Confidential
Yes x
No

Keywords

Deposited at

SISÄLLYSLUETTELO

TERMINOLOGIAA

BSC	Base Station Controller, tukiasemaohjain
BSS	Base Station Sub-System, tukiasemajärjestelmä
BTS	Base Tranceiver Station, tukiasema
CDR	Call Data Record, puhelutietue. ARM:ssa oleva prosessoitava ja laskutettava puhelutietue, joka voi olla monessa eri muodossa prosessin aikana.
GMSC	Gateway MSC, GSM-kauttakulkukeskus
GPRS	General System Packet Radio Service, GSM-pakettidatapalvelu
HLR	Home Location Register, kotirekisteri
INPUT-tiedosto	Asiakkaan puhelutiedosto ennen prosessointia. Se on tavallisesti muodoltaan asiakkaan erityisformaatissa.
MDS	Mediation Device Solution, Comptel’n sovellusohjelmisto, joka hoitaa laskutustietojen keruun- että palveluiden aktivointitoiminnot.
MDS/AMD	Accounting Mediation Device, laskutusmediaattori
MDS/ARM	Accounting Record Modification application, joka huolehtii laskutustietojen muutoksista laskutussysteemiin ja hyväksyy ne.
MSC	Mobile Switching Center, matkapuhelintilaajan päätekeskus.

NSS	Network and Switching Sub-System, keskusjärjestelmä
OSS	Operations Sub-System, käytön hallinta järjestelmä
OUTPUT -tiedosto	Saadaan tuloksena, kun asiakkaan puhelutiedosto prosessoidaan ARM:ssa.
PSTN	Public Switched Telephone Network, kiinteä puhelinverkko
Release Notes	Sisältää tietoa ko. ohjelmistopaketista, kuten paketin ja dokumentin versionumerot, paketin asennusohjeet sekä Release Notes:n versiohallinnan.
Ruleset	MDS/ARM ohjelmakoodi, säännöstö
SMSC	Short Message Service Center, lyhytsanomakeskus.
Spesifikaatio	MDS/ARM säännöstön toiminnallinen kuvaus, dokumentti
YKM	Yhteiskanavan merkinanto
VLR	Visitor Location Register, vierailijarekisteri
Älyverkko	Älyverkko ei ole itsenäinen verkko, vaan älyverkon avulla parannetaan ja monipuolistetaan jo olemassa olevan televerkon palveluja. Sen tarkoituksena on erottaa toisistaan puhelinverkossa puhelun kytkentä ja ohjaus.

1 JOHDANTO

Insinöörityö toteutetaan kajaanilaiselle Ebsolut Oy:lle. Työ sisältää yleisen osan GSM-tekniikasta ja mediaattoriohjelmistosta sekä projektiluontoisen asiakastyön.

GSM-tekniikan osuus työssä sisältää asiaa GSM-verkoista, matkapuhelimesta, puhelun kytkeytymisestä sekä GSM-verkon tarjoamista palveluista. Siinä ei ole tarkoitus perehdyttää lukijaa tarkkoihin yksityiskohtiin, vaan antaa yleiskuvaa ja pohjaa mediaattoriohjelmiston käsittelyä varten. Mediaattoriohjelmisto on laskutustietojen keruu- ja palveluiden aktivointitoimintoihin keskittyvä ohjelmisto.

Ebsolut Oy tulee sanoista Electronic Business Solution, ja on kajaanilainen kesällä 2000 perustettu yritys. Ebsolut Oy on keskittynyt voimakkaasti kasvavan ja kehittyvän tietoliikenteen tietojenkäsittelyn ohjelmistokehitykseen. Esimerkkeinä voisi mainita testausjärjestelmät ja mediaattoriohjelmistot. Sen vahvoja osaamisalueita ovat vaativat projektitoimitukset, menetelmäosaaminen mm. UML, ohjelmistokehitysvälineet: C++, Java, SQL, UNIX sekä mediaattoriohjelmistot. Ebsolut Oy:n yksi tärkeistä yhteistyökumppaneista on Comptel Corporation, jolle Ebsolut Oy tekee lähinnä alihankintatöitä.

Projektityö tehdään Comptelin asiakkaalle, jonka Ebsolut Oy toteuttaa. Projektityön tarkoituksena on toimittaa asiakkaalle ohjelmisto, joka käsittelee Ericsson 8.0 keskuksesta saatuja matkapuhelutietoja, ja toimittaa ne suuren eurooppalaisen teleoperaattorin laskutusjärjestelmään. Työhön kuuluu suunnittelu, toteutus, testaus sekä metodien tekeminen. Tavoitteena on luoda kokonaisuus, johon kuuluu uusi dokumentti (spesifikaatio), ohjelmakoodi (ruleset), testiraportti ja metodin rakentaminen sekä testi- että tuotantokoneelle. Nämä kaikki vaiheet olen tehnyt projektissa itse. Lopuksi ohjelmakoodi on tarkoitus asentaa asiakkaan tuotantokoneelle annetun aikataulun mukaan.

2 GSM-VERKKO

Matkapuhelinverkkojen ja -puhelimien kehitys voidaan jaotella kolmeen sukupolveen. Ensimmäisen sukupolven muodostavat analogiset matkapuhelimet, toisen sukupolven digitaaliset matkapuhelimet ja kolmannen sukupolven muodostavat puhelimet, joissa on edellisiä huomattavasti suurempi datansiirtokapasiteetti.

Tällä hetkellä on käytössä toisen sukupolven matkapuhelinverkot, jotka perustuvat digitaalitekniikkaan. Kyseiset verkot ovat ominaisuuksiltaan tehokkaampia kuin 1. sukupolven järjestelmät ja sen myötä ne mahdollistavat uusien monipuolisten palvelujen käytön, kuten tehokkaamman langattoman datasiirron, tekstiviestit, faksit ja paremman puheensalauksen. Järjestelmät ovat edelleenkin maanosakohtaisia, eikä koko maailman kattavaa verkkoratkaisua ole vieläkään onnistuttu luomaan.

GSM (Global System for Mobile communications) on matkapuhelinstandardi, jota käytetään pääasiassa Euroopassa. Se mahdollistaa puhe- ja datasiirron langattomasti matkapuhelinverkossa. GSM otettiin kaupalliseen käyttöön ensimmäisenä maailmassa Suomessa. Vuosi oli 1992 ja käyttöönotto tapahtui Telen (nykyisin Sonera) ja Radiolinjan toimesta. GSM -tekniikka perustuu laajaspektriseen lähetykseen, taajuushyppelyn ja aikajakomultipleksoinnin (TDMA) käyttöön. Perus GSM-verkko tarjoaa datansiirtomahdollisuudet nopeuksin 4800 ja 9600 kbit/s. Verkko toimii 900 MHz taajuudella. Käyttäjämäärän kasvaessa GSM-verkko on havaittu riittämättömäksi ja suurimmissa keskuksissa on otettu käyttöön GSM-tekniikkaan perustuva GSM-1800-järjestelmä GSM-900:n rinnalle. Puhelinvalmistajat valmistavat kaksitaajuuspuhelimia, jotka toimivat molemmissa verkoissa. Yhdysvalloissa on lisäksi käytössä 1900 MHz:n alueella toimiva GSM-1900.

PDC (Personal Digital Cellular) on Japanin digitaalinen matkapuhelinjärjestelmä, jonka taajuusalueet ovat 800 ja 1500 MHz. PDC käyttää GSM:n tapaan TDMA:ta ja muistuttaa muutoinkin hyvin paljon GSM:ää. Järjestelmän suurin

ero GSM:ään verrattuna on laskutus, joka tapahtuu matkapuhelinten etäisyyden mukaan. Ensimmäinen PDC-verkko otettiin käyttöön maaliskuussa 1993.

CDMA (Code Division Multiple Access) on GSM:n kilpailija matkapuhelinverkkojärjestelmänä. Se otettiin ensimmäisenä käyttöön Hong Kongissa vuonna 1995 ja vuonna 1996 Yhdysvalloissa, jossa käytetään nimeä IS-95 (N-CDMA). CDMA perustuu aikajaon ja taajuushyppelyn lisäksi matemaattisten koodien käyttöön. Näin ollen CDMA:n tarjoama kapasiteetti on suuri ja se tarjoaakin tiedonsiirtonopeuden 64 kbit/s. Tällä hetkellä maailmassa on 40 miljoonaa CDMA:n käyttäjää.

D-AMPS on digitaalinen versio 1. sukupolven AMPS:ta (Advanced Mobile Phone Service, Yhdysvallat). Se käyttää TDMA-pohjaista IS-54-tekniikkaa taajuudella 800 MHz, joka on sama kuin AMPS:in käyttämä taajuus. D-AMPS hyödyntää vanhoja AMPS verkkoja ja ne toimivat rinnakkain.

2.5 sukupolvi perustuu GSM-verkon kehittämiseen seuraavin ratkaisuin: HSCSD (High Speed Circuit Switched Data), nopea piirikytkentäinen datasiirto, GPRS (General Packet Radio System), pakettikytkentäinen radioverkko sekä EDGE (Enhanhed Data rates for Global Evolution), parannettu datansiirto.

GSM tarjoaa piirikytkentäisen tiedonsiirtomenetelmän, joka sopii hyvin puheensiirtoon sen korkeiden "reaaliaikavaatimusten" vuoksi. GSM tarjoaa myös mahdollisuuden lyhytsanomien lähetykseen ja rajoitetusti datansiirtoon. Piirikytkentäisyydestä johtuen GSM-järjestelmässä MS (Mobile Station) varaa yhteyden vaatimat resurssit käyttöönsä koko yhteyden ajaksi.

Perus-GSM 900 (P-GSM) toimii taajuusalueella 890-915 MHz uplink-suunnassa ja 935-960 MHz downlink-suunnassa. P-GSM-järjestelmässä on 124 kanavaa eli 25 MHz:n taajuuskaista, joka on yleensä jaettu kilpailevien operaattoreiden kesken. Jokaisen kanavan kaistanleveys on 200 KHz. Taajuusalueen alussa on yhden kanavan suojaetäisyys, joten ensimmäinen varsinainen käyttötaajuus on 890,2 MHz. Lähetys- ja vastaanottotaajuudet on jaet-

tu vielä kahdeksaan aikaväliin eli TDMA-kehyksen (Time Division Multiple Access) jaksoihin. Koska GSM-järjestelmässä käytetään useita radiotaajuuksia on kyseessä TDMA:n ja FDMA:n (Frequency Division Multiple Access) yhdistelmä.

GSM:ssä MS (Mobile Station) lähettää ja vastaanottaa omalla liikennekanavallaan (yksi aikaväli TDMA-kehyksessä) yhden purskeen kahdeksan aikavälin aikana, jolloin saavutetaan 9,6 kbit/s tai 14,4 kbit/s tiedonsiirtonopeus riippuen käytetystä modulointi- ja kanavan koodaustekniikasta. GSM:n radiorajapinnan resurssit, jotka piirikytkentäisiltä yhteyksiltä jäävät käyttämättä, tarjoavat tulevalle pakettikytkentäiselle GPRS-järjestelmälle käyttöresurssit.

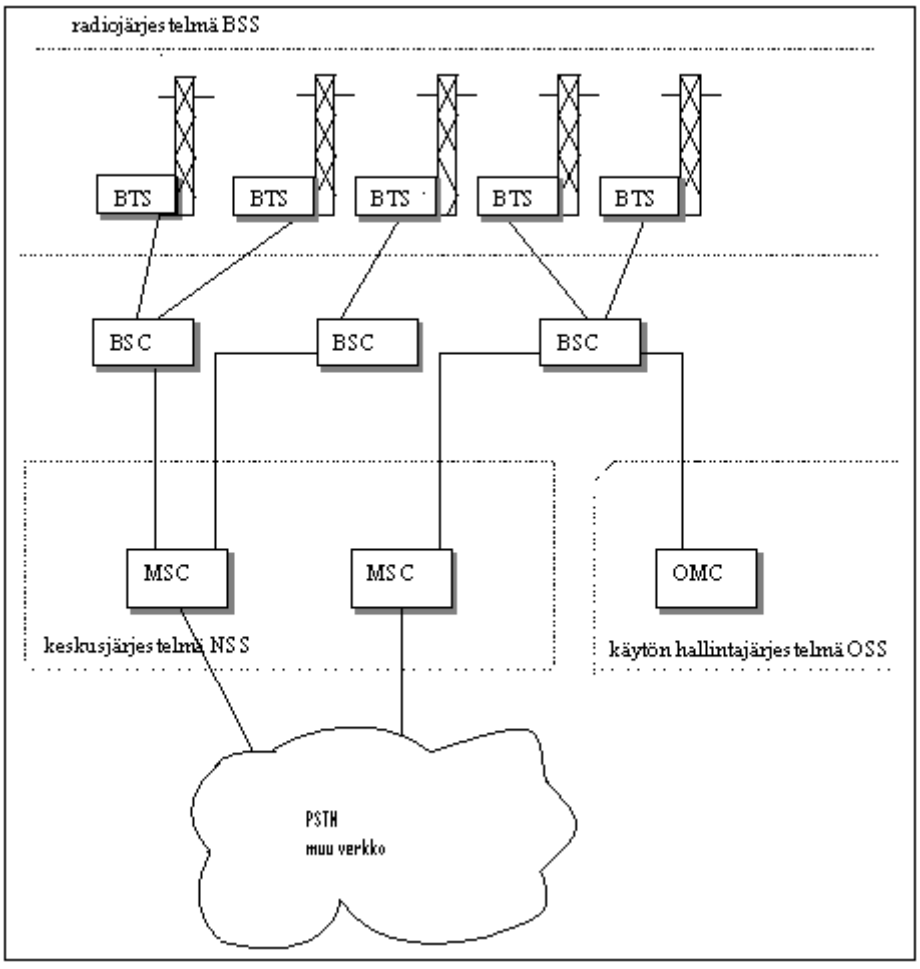
Uudeksi ratkaisuksi (3. sukupolvi) tulee UMTS (Universal Mobile Telecommunications System), joka pystyy hyödyntämään osittain nykyisiä GSM-verkkoja. UMTS tulee ensimmäisen kerran käyttöön mahdollisesti vuonna 2002 ja järjestelmä lienee laajasti käytössä Euroopassa vuonna 2005. Suomi tulee olemaan ensimmäisiä maita, jossa UMTS otetaan käyttöön. Suomessa UMTS-toimiluvat on jo jaettu neljälle operaattorille (Sonera, Radiolinja, Telia, 3G). UMTSin maailmanlaajuisesta suunnittelusta vastaa Kansainvälinen televiestintäliitto ITU (International Telecommunications Union). Järjestö on lähes mahdottoman haasteen edessä, jotta UMTS-järjestelmästä saataisiin maailman laajuinen standardi.

3. sukupolven UMTS-matkapuhelinverkossa on laajakaista- ja multimediaominaisuudet ja sen tiedonsiirtonopeus on jopa 2 Mbit/s. Käytetyt taajuusalueet ovat 1885-2025 MHz ja 2110-2200 MHz. Verkon tulee tukea tarpeesta riippuen eri siirtonopeuksia ja asymmetrista tiedonsiirtoa, johtuen palveluiden erilaisesta kapasiteettitarpeesta [1].

2.1 GSM-verkon osat

GSM-verkko jaetaan keskusjärjestelmään (NSS), tukiasemajärjestelmään (BSS) sekä näitä hallitsevaan ja ohjaavaan käytön hallintajärjestelmään (OSS). Kuvassa 1 on esitetty GSM-järjestelmän periaatteellinen lohkokaaviokuva. Lisäksi GSM-verkoissa on muita elementtejä mm. puhepostin säilytykseen ja laskutustietojen keräykseen, jotka ovat tyypillisesti kunkin operaattorin sisäisiä järjestelmiä.

Matkapuhelinkeskus (MSC) on matkapuhelintilaajan päätekeskus. Sen tärkeimpänä tehtävänä on kytkeä GSM-verkkoon tulevat puhelut oikealle BSS:lle tukiasemaohjaimen (BSC) ja tukiaseman (BTS) kautta sekä BSS:ltä tulevat puhelut muihin verkkoihin. MSC:n toiminta perustuu kiinteän verkon puhelinkeskuksiin. Datayhteyksien osalta MSC:n ja ulkoisten verkkojen yhteensovittamisesta huolehtivat yhteensovitustoiminnot (IWF) [2, s. 33].



Kuva 1. Periaatekuva GSM-verkon arkkitehtuurista [2]

3 MATKAPUHELIN JA MATKAPUHELINPALVELUT

Matkaviestin (MS, Mobile Station) koostuu matkaviestinlaitteesta (ME, Mobile Equipment) ja siihen kytketyistä SIM-kortista (Subcriber Identity Module). Matkapuhelin tarkoittaa ensisijaisesti puheyhteyksiin tarkoitettua matkaviestintä. GSM on digitaalinen, 900 Mhz:n taajuusalueella toimiva solukkotyyppinen matkaviestinjärjestelmä (Global System for Mobile communications). Se jakautuu ajoneuvo- ja käsipuhelimiin, joista jälkimmäinen on ylivoimaisesti GSM-verkoissa suurin osa [2, s. 61]. Spesifikaatiot määrittelevät MS:lle pakollisia ja valinnaisia toimintoja sekä ominaisuuksia. Pakollisia toimintoja ovat mm. maan ja verkon valinta ja IMEI-koodi. Valinnaisia toimintoja ovat mm. numeronäppäimistö ja lyhytsanoman näyttö. Toiminnot ja ominaisuudet on esitelty kokonaisuudessaan liitteessä (A).

SIM-kortti

Matkapuhelintilaajan SIM-kortti on älykortti, jota voidaan kutsua myös tilaajan tunnistusyksiköksi tai GSM-kortiksi. Siihen on tallennettu pysyvästi tilaajaa koskevia tunnistetietoja sekä tunnistusalgoritmit A3 ja A8. Käyttäjä voi tallentaa SIM-kortille muuttuvina tietoina puhelinnumeroita alfanumeeriseen tietoon yhdistettynä, lyhytsanomia sekä suosituimmuuslistan matkapuhelinverkoista roaming-tilanteessa. Lisäksi tilaaja voi määrittää SIM-kortille esimerkiksi puheluiden estoja.

SIM-kortin avulla tilaaja voi käyttää periaatteessa mitä tahansa GSM-matkapuhelinta. Esimerkiksi eurooppalainen GSM 900 tai GSM 1800–verkon tilaaja voi omalla SIM-kortillaan käyttää USA:n GSM 1900–verkkoa sopivalla matkaviestinlaitteella, kunhan kyseisellä operaattorilla on roaming-sopimus tilaajan kotiverkon operaattorin kanssa.

Hyötynäkökohtina SIM-kortista ovat mm. turvallisuustoiminnot, yhteensopivuus periaatteessa kaikkien GSM-puhelinten ja –verkkojen kanssa, sekä palveluiden joustava lisääminen ja poistaminen [2, s. 73].

3.1 Järjestelmän numerointi

GSM-järjestelmään liittyy monia numeroita. Tilaaja käyttää niistä vain pientä osaa ja käytännössä julkisen tilaajanumeron ja PIN-koodin lisäksi ei yleensä tarvita muita. Tilaajan tunnistukseen ja yhteyksien muodostukseen käytetään kuitenkin numeroita, joita verkko tarvitsee toimiakseen, ja lisäksi tilaajalaitteella sekä radioverkolla on omat numeronsa. Seuraavassa niistä on esitelty yleisimmät.

Tilaajanumerot

Matkaviestintilaajan kansainvälinen ISDN-numero, MSISDN on julkinen luettelonumero. Sitä voidaan käyttää kansainvälisillä yhteyksillä GT-osoitteena (Global Title). MSISDN-numero sisältää maan numeron CC:n (Country Code), kansainvälisen verkkotunnuksen NDC:n (National Destination Code) ja varsinaiset tilaajanumeron SN:n (Subscriber Number). Esimerkiksi Suomen CC on 358. NDC on GSM-verkon tunnus kansallisessa numerointisuunnitelmassa. Esimerkiksi Radiolinjan verkossa se on 50. Kansainvälisessä verkossa MSISDN-numero sisältää kaukoliikenteen tunnuksen P:n, NDC:n ja SN:n. Esimerkiksi Suomella P on 0. NDC:n muodostama numero on Radiolinjalla 050.

Yhdellä tilaajalla voi olla useita MSISDN-numeroita. Siksi GSM-verkossa käytetään tilaajan ensisijaisena hakuavaimena kansainvälistä matkaviestintätilaajan tunnusta IMSI (International Mobile Subscriber Identity), joka on tilaajan yksikäsitteisesti määrittävä numero. IMSI:n avulla mm. kohdistetaan liikennemaksut. IMSI on enintään 15 merkin pituinen numero. IMSI-tunniste koostuu tunnuksista NMSI (National Mobile Subscriber Identity) ja MCC (Mobile Country Code). NMSI on matkaviestintilaajan kansallinen tunnus. MCC-numero on X.121 –suosituksen mukainen kolmenumeroinen matkaviestinnän maatunnus. Suomella se on 244, joka eroaa puhelinverkon maatunnuksesta CC. NMSI-numero koostuu edelleen kahdesta tunnuksesta MNC ja MSIN. Matkaviestinverkon tunnus MNC (Mobile Network Code) on GSM-verkon yksilöivä 2-numeroinen koodi, joka eroaa GSM-verkon tunnuksesta NDC. Matka-

viestintilaajan tunnus (MSIN, Mobile Subscriber Identification Number) on tilaajan yksilöivä koodi verkon sisällä [2].

Päätelaitteen numero

Kansainvälisen matkaviestimen laitetunnuksen (IMEI, Internatioanal Mobile Equipment Identity) avulla matkaviestin voidaan tunnistaa yksikäsitteisesti. Laitetunnuksella ei ole vastaavuutta tilaajan henkilöön, IMSI- tai MSISDN-numeroihin, vaan se on puhtaasti laitteen koodi. Puhelu sallitaan, mikäli verkko toteaa puhelun aloitussignaloinnissa IMEI:n hyväksytyksi. IMEI-koodia käytetään mm. luvattomien ja varastettujen päätelaitteiden tunnistamista varten laiterekisterin avulla. IMEI on 15-osainen numero, joka koostuu tunnuksista TAC, FAC, SNR ja SP. TAC (Type Approval Code) on kuusinumeroinen tyyppihyväksyntäkoodi, FAC (Final Assembly Code) kaksinumeroinen valmistuspaikkatunnus, SNR (Serial Number) kuusinumeroinen sarjanumero ja SP (Spare) yksinumeroinen varanumero (joka on aina 0 MS:n lähettäessä sen). Koska IMEI-koodista ilmenee yksiselitteisesti sekä laitteen merkki että tyyppi, voi operaattori monitoroida koodin perusteella mm. verkossaan käytössä olevien puhelintyyppien käyttöjakaumia [2].

SIM-kortin numero

Kullakin GSM-järjestelmän SIM-kortilla on yksilöllinen ICC-numero (Integrated Circuit Card), josta ilmenee operaattorin tunnus, kortin valmistusaika, valmistajan tunnus, sarjanumero ja tarkistustunnus [2, s. 202].

Verkon numerot

Kullakin GSM-verkon sijaintialueella (LA, Location Area) on oma sijaintialuetunnuksensa LAI (Location Area Identification), joka koostuu tunnuksista MCC, MNC ja LAC. MCC (Mobile Country Code) on matkaviestinnän maatunnus ja MNC (Mobile Network Code) on matkaviestinverkon tunnus. Nämä kaksi numeroa ovat samoja kuin IMSI-koodissa. LAC (Location Area Code) on sijaintialueen koodi, joka muodostuu numerosta välillä 0…65535 eli kahdesta

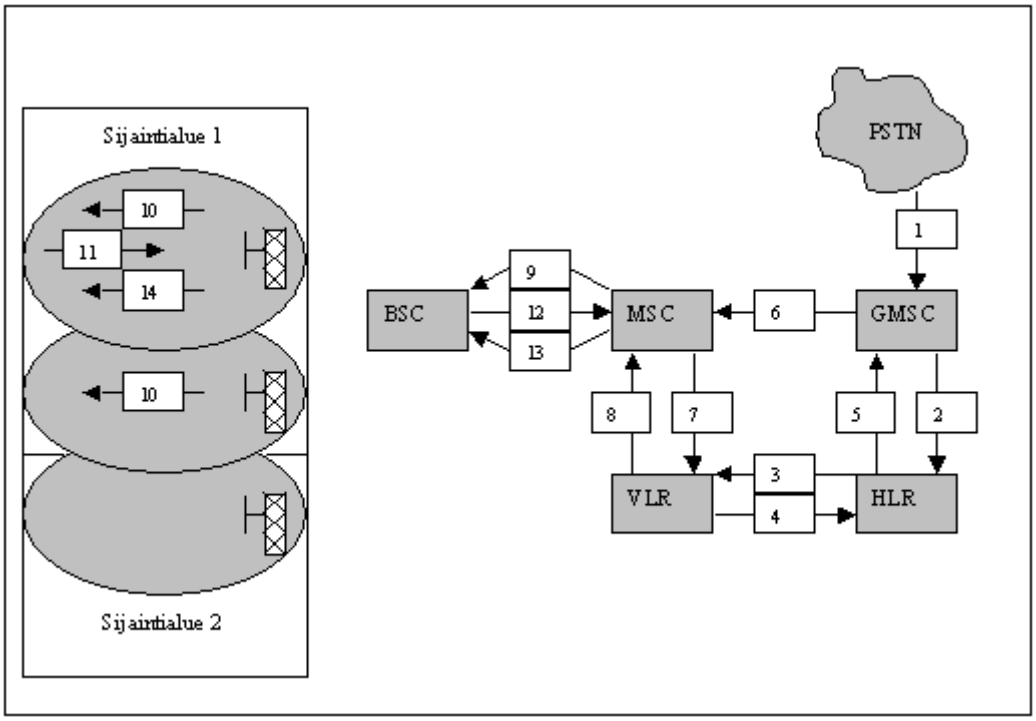
oktetista. Koodi voidaan esittää heksadesimaalisessa muodossa, lukuun ottamatta numeroita 0000 ja FFFF, jotka ovat varattu erikoistilanteisiin LAI:n puuttuessa.

Tukiasematunnus BSIC (Base Station Identity Code) on LAI-tunnusta karkeampi ”värikoodi”, jonka idea on lähinnä erotella alueella olevat solut toisistaan. BSIC muodostuu tunnuksista NCC ja BCC. BSIC voi saada jonkin 64 arvosta. 3-bittinen NCC (Network Colour Code) erottelee samalla alueella kuuluvat eri GSM-verkot toisistaan. Spesifikaatio määrittävät kullekin maalle NCC-koodin, joka on Suomelle 0.

CI (Cell Identity) on solutunnus, jonka numeroarvo on välillä 0…65535. Tämän koodin avulla solu tunnistetaan kunkin sijaintialueen sisällä. Solun kansainvälinen tunnut (CGI, Cell Global Identification) koostuu LAI-koodista (eli MCC-, MNC- ja LAC-kentistä) sekä CI-kentästä [2].

3.2 Puhelun muodostus ja reititys

Kun puhelu on puhelimesta alkava (MOC, Mobile Originated Call), on yhteyden muodostus GSM-verkon osalta suoraviivainen toimenpide. Kun GSM-verkko on tarkastanut tilaajan oikeudet käyttää verkkoa ja käynnistänyt salauksen, ohjataan merkinanto kiinteään verkkoon siitä MSC:stä, johon MS on parhaillaan kytkeytyneenä. Mikäli B-tilaaja on saman keskuksen alla oleva GSM-tilaaja, pysyy merkinanto saman keskuksen sisällä. Kun puhelu on puhelimeen päättyvä (MTS, Mobile Terminated Call), on merkinanto hieman monimutkaisempaa. Mikäli puhelu tulee kiinteän verkon kautta MSC:hen, ohjataan merkinanto kuvan 2 mukaisesti [2, s. 212].



Kuva 2. Merkinannon ohjaus kiinteän verkon kautta MSC:hen [2, s. 213]

Kuvan merkinanto jakautuu seuraavasti:

1) Puhelu saapuu PSTN:stä ensimmäiseen GSM-keskukseen, jota kutsutaan kauttakulkukeskukseksi (GMSC, Gateway MSC).
2) GMSC tekee merkinantokyselyn MSISDN-numeron perusteella B-tilaajan HLR:ään.
3) HLR:ssä on tieto siitä VLR:tä, jossa B-tilaaja on parhaillaan rekisteröityneenä; HLR lähettää merkinantokyselyn VLR:ään pyytäen MSRN-numeroa.
4) VLR valitsee vapaan MSRN-numeron, yhdistää sen B-tilaajan ISDN-numeroon omassa tietokentässään ja lähettää MSRN-numeron HLR:lle.
5) HLR lähettää MSRN:n edelleen GMSC:lle tallentamatta sitä omaan tietokenttäänsä.
6) GMSC reitittää puhelun MSRN-numeroa vastaavaan MSC:hen.

7) MSC kysyy omasta VLR:stä, mitä ISDN-numeroa MSRN-numero vastaa.
8) VLR antaa MSC:lle B-tilaajan ISDN-numeron, tai TMSI:n, mikäli mahdollista.
9) MSC lähettää ISDN- tai TMSI-numeron radiojärjestelmäänsä.
10) Radiojärjestelmä lähettää kutsun (paging) kaikkien niiden tukiasemien kautta, jotka kuuluvat B-tilaajan sijaintialueeseen.
11) MS vastaa oman C1- tai C2-listansa mukaisen parhaan solun kautta kuittauksen.
12) Verkko tarkastaa tilaajan käyttöoikeuden ja suorittaa radiorajapinnan suojaukseen liittyvät laskennat.
13) MSC kytkee liikennekanavan radiojärjestelmään.
14) Puhelu kytkeytyy.

MS:n siirtyessä vierailtavaan verkkoon sijainnin seurantaan (Roaming), on puheluiden muodostuttava edelleen sekä lähtevässä että tulevassa suunnassa. Operaattoreiden keskinäisellä roaming-sopimuksella saadaan puhelut sekä muut erikseen määritettävät palvelut, kuten SMS, toimimaan. Roaming-operaattoreiden kesken lähetetään mm. tilaajan tunnistukseen vaadittavat salaustripletit kansainvälistä YKM-verkkoa pitkin. Roaming-sopimukseen sisältyy teknisten toteutusten lisäksi operaattoreiden välisen laskutuksen periaatteet [2, s 222].

Sijainnin seurannassa tilaajat eivät saa välttämättä samoja palveluita (mm. äänitiedotukset) kuin kotiverkossa. Tähän on hiottu ratkaisua työaiheena nimeltä CAMEL (Customised Applications for Mobile Network), joka mahdollistaa operaattoreiden palveluiden siirron tilaajille kotiverkon ulkopuolelle. CAMEL on toiminto, joka yhdistää GSM- ja älyverkkomaailman [2, s. 223].

3.3 Palvelut

GSM-järjestelmän palvelut jaetaan verkkopalveluihin (Bearer Services) ja telepalveluihin (Tele Services). Lisäksi GSM:ssä määritetään laaja joukko lisäpalveluita (Supplementary Services).

Verkkopalvelut ulottuvat siirtoverkon päästä päähän, ja vastaavat OSI-määritysten tasoja 1…3 eli fyysistä, siirtoyhteys- ja verkkotasoa. Verkkopalvelut mahdollistavat tiedonsiirron käyttäjä- ja verkkorajapinnoilta toisille. Telepalvelut ulottuvat siirtoverkon lisäksi päätelaitteeseen TE (Terminal Equipment) saakka, ja pitävät siten sisällään kaikki OSI-kerrokset 1…7. Telepalvelut mahdollistavat käyttäjien välisen tiedonsiirron määritettyjen yhteyskäytäntöjen mukaisesti [2, s. 107].

Tele- ja verkkopalveluja kutsutaan peruspalveluiksi, jotka ovat joko oletusarvoisesti tai erikseen määritettäessä tilaajasuhteen alusta saakka tilaajan käytettävissä. Lisäpalvelut ovat näiden parannuksia tai näitä tukevia palveluita. Lisäpalvelut täydentävät tai muuntavat peruspalveluita. Lisäpalveluiden määrä kasvaa sitä mukaa kun laitevalmistajat saavat spesifikaatioiden määrityksiä tuotantoon ja kun operaattorit ottavat palveluita käyttöön. Lisäpalveluiden yläpuolelle määritetään vielä lisäarvopalveluita. Ne käyttävät GSM-järjestelmän tele-, verkko- tai lisäpalveluja palvelualustanaan varsinaisen informaation tai ohjaussanomien siirtoon. Esimerkkejä lisäarvopalveluista voivat olla pankkipalvelut tilisaldokyselyineen ja laskun maksuineen sekä uutisotsikkopalvelut [2].

Operaattorien tehtävänä on huolehtia yhteyksien asianmukaisesta laskutuksesta. GSM-spesifikaatiot eivät määritä lopullista laskutuksen toteutusta, mutta verkosta on mahdollista kerätä monista elementeistä yhteyteen liittyvää tietoa erilliseen laskutusjärjestelmään. Yhteystapahtumista käytetään nimitystä ”tiketti”. Laskutustikettejä kerätään esimerkiksi GSM-kauttakulkukeskuksien avulla. Tiedot siirretään laskutusjärjestelmään kootusti, jossa niitä voidaan edelleen käsitellä esimerkiksi yhteydenottoaikaan, tilaajan liittymän määrityk-

siin ja käyttäjän sijaintiin perustuen. GSM-järjestelmän laskutus poikkeaa kiinteän verkon periaatteista yhdessä asiassa merkittävästi: mikäli GSM-käyttäjä on rekisteröityneenä muuhun kuin kotiverkkoon, maksaa hän tulevasta puhelusta kotiverkon ja vierailtavan verkon välisen yhteyden. Matkapuheluiden laskutus on aikaan perustuva [2, s. 113].

4 MEDIAATTORIOHJELMISTO

4.1 Comptel Corporation

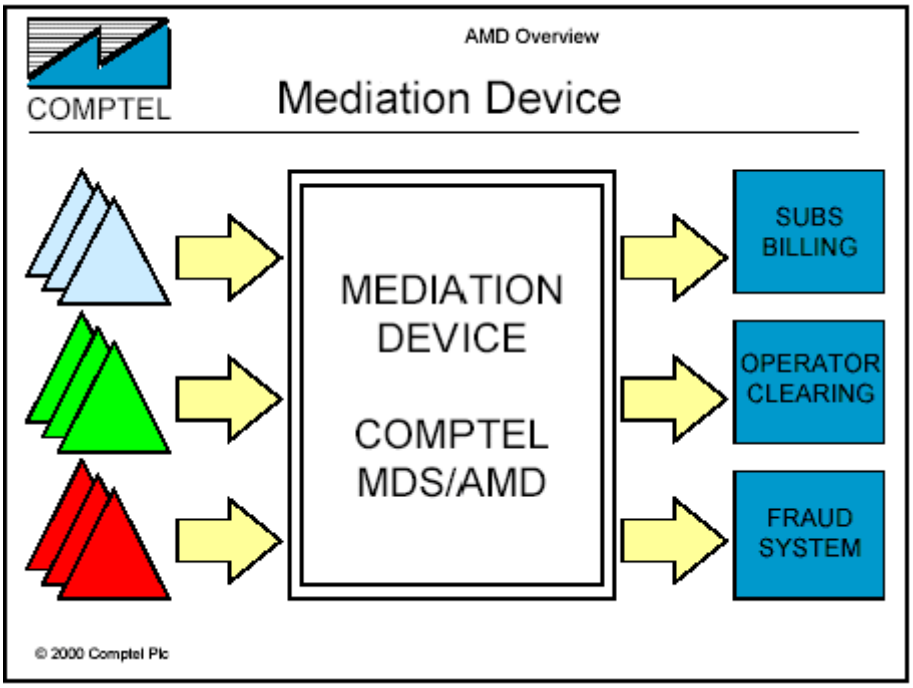
Comptel on Elisa Communicationsin tytaryhtiö, joka kehittää ja toimittaa mediaattoriohjelmistoja (Mediation Device) tietoliikenneyrityksille. Comptelin MDS (Mediation Device Solutions) on yksi johtavista monitoimittajaympäristöjen mediaattoriohjelmistoista. Comptelin MDS-ratkaisuun kuuluvat sekä laskutustietojen keruu- että palveluiden aktivointitoiminnot, joita käytetään laajalti kiinteissä, matkapuhelin-, satelliitti- ja dataverkoissa kaikkialla maailmassa.

Comptel tarjoaa kansainvälisille asiakkailleen telekommunikaatioalan ja tietojärjestelmien tuntemusta sekä konsultointi-, suunnittelu-, järjestelmäkehitys- ja asiakaspalvelua monitoimittajaympäristössä. Comptelilla on kansainvälinen yhteistyöverkosto, johon kuuluu lukuisia johtavia järjestelmäintegraattoreita sekä asiakashallinta- ja laskutusjärjestelmävalmistajia.

Comptelin perustama tytäryhtiö Comptel PASSAGe Oy tarjoaa valmiita liiketoimintaratkaisuja matkapuhelin- ja palveluoperaattoreille. PASSAGe-konsepti tarjoaa ratkaisun sisällöntuottajien ja palveluiden hallintaan. Se huolehtii myös maksutapahtumien välityksestä sekä käyttäjien profiilin hallinnasta, jolloin uusien langattomien Internet-palvelujen käyttöönotto nopeutuu.

4.2 Mediaattoriohjelmisto

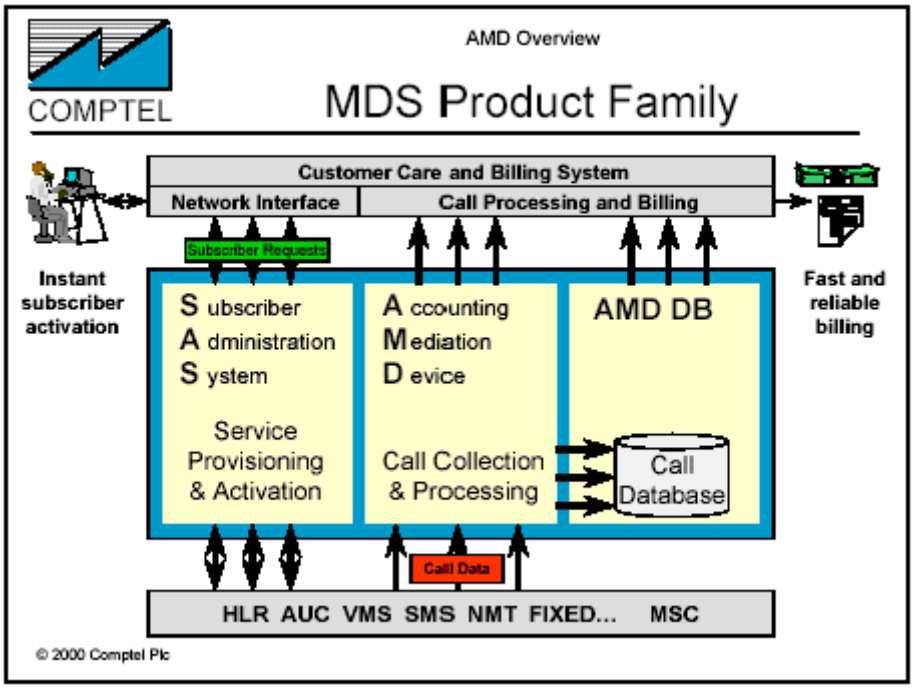
Mediaattoriohjelmistoa tarvitaan keskuksilta tulevien puhelutietojen koodausta, muokkausta ja kelvolliseksi todentamista varten sekä niiden suuntaamista asiakkaan hallinta ja laskutussysteemiin. Yhdestä keskuksesta tulevasta puhelutiedosta voidaan joutua muodostamaan kolme erilaista output-tietuetta: yksi laskutussysteemiin, yksi operaattoreiden väliseen selvitysjärjestelmään ja yksi ilkivallalta suojelutarkoitukseen. Datan järjestäminen uudelleen ja sen käsitteleminen voi olla hankalaa ilman järjestelmällistä organisaatiota, joka hoitaa tänä päivänä syntyvät suuret puhelumäärät. Kuva 3 esittää periaatekuvan mediaattoriohjelmiston sijainnista puhelinjärjestelmässä [3].



Kuva 3. Mediation Device Solution (MDS) [3]

Mediaattoriohjelmisto on kehitetty sekä kiinteitä että matkapuhelinten televiestintä verkko-operaattoreita varten. Se hoitaa yhä kasvavaa tietovirtaa tietoliikenneverkon ja asiakkaan palvelu- ja laskutussysteemin välillä. MDS teknologia seuraa Telecommunications Management Network:n (TMN) rakenteellista mallia ja se on International Telecommunication Unionin (ITU) standardisoima, johon Comptel Corporationin ratkaisu juuri perustuu. [4]

Comptelin mediaattoriohjelmiston tuoteperhe yhdistää asiakkaan hallintasysteemin verkkoelementtiin (NE) (Kuva 4). Tuoteperheeseen kuuluu MDS/AMD (Accounting Mediation Device), MDS/AMD DB (AMD Database) ja MDS/SAS (Subscriber Administration System). AMD kerää, muokkaa, yhdistää ja esihinnoittelee keskuksilta tulevat puhelutiedot. AMD DB varastoi puhelutiedot tietokantaan. API sallii asiakassysteemien tietokantakyselyt. Tietokanta sisältää raportointi- ja ilkivallalta suojelun ominaisuudet. SASin tehtävät ovat tilaajan ja palveluinfon toimittaminen, aktivoiminen, hävittäminen ja tiedusteleminen verkossa. [3]



Kuva 4. MDS tuoteperheen kuvaus [3]

MDS-sovellus käyttää alustanaan Hewlett Packardin, Digitalin ja IBM:n valmistamia Unix-tietokoneita. Unix arkkitehtuuri sopii hyvin MDS sovellukseen FIFO-, moniajo- ja pipe-ominaisuuksiensa vuoksi. Etuna ovat myös unix-koneiden hyvä saatavuus, ja sopivuus sekä pienille että suurille tietoliikenne-operaattoreille. MDS-sovellus käyttää Oracle-tietokantaa. Tyypillinen MDS-asennus koostuu yhdestä MDS-sovelluksesta ja asiakkaalle tehdystä moduulista. [3]

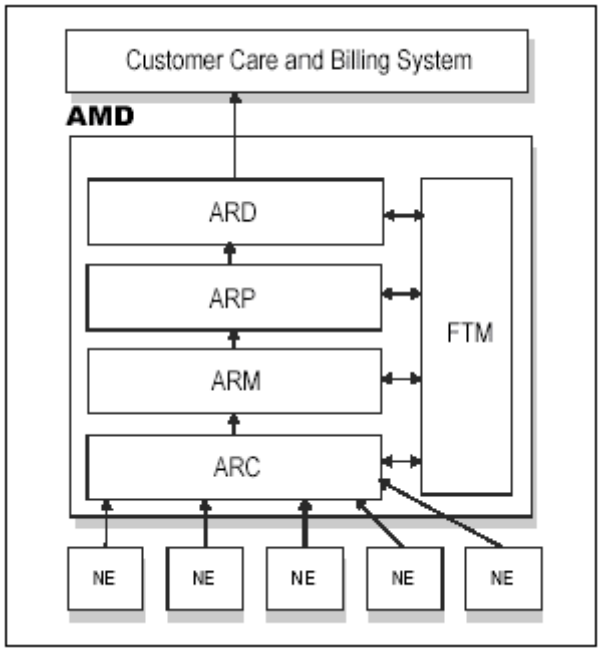
S-lang on tehokas pinopohjainen tulkki, jonka syntaksi on C:n kaltainen. Se on alusta pitäen suunniteltu helposti sulautettavaksi ohjelmiin laajennukseksi. S-langia soveltuu ohjelmien kehittämiseen ja vianetsintään. Koska S-lang muistuttaa C:tä, S-lang-skriptien koodaus C:ksi on helppoa, jos tarvetta tulee [5]. MDS/ARM ympäristössä käytetään S-lang ohjelmointikieltä.

Expect on korkeatason ohjelmointikieli, joka alkujaan on suunniteltu korvaamaan ihmisen vastauksia yksinkertaisiin kysymyksiin vuorovaikutusprosessissa (ftp, telnet). Sen avulla voidaan automatisoida yksinkertaisia tehtäviä. Expect:iä käytetään vuorovaikutteisesti tai ohjelmointikielenä [6]. MDS/SAS ym-

päristössä käytetään Expect–ohjelmointikieltä, kun suoritetaan tehtäviä verkkoelementissä.

4.3 AMD-arkkitehtuuri

MDS/AMD jaetaan pienempiin osiin eli moduuleiksi, joilla jokaisella on oma tehtävänsä puhelutietueiden käsittelyssä (kuva 5). Moduulit ovat FTM (File Transfer Manager), ARC (Accounting Record Collection), ARM (Accounting Record Modification), ARP (Accounting Record Pre-rating) ja ARD (Accounting Record Delivery). AMD:tä valvova moduuli on FTM, joka kontrolloi toisten moduulien toimintoja. ARC huolehtii tiedostojen keräilystä keskuksista unix-koneelle. ARM muuttaa, yhdistää ja valitsee puhelutietoja yhtenäiseen formaattiin, jota seuraavat moduulit (ARP, ARD) myös ymmärtävät. ARP esihinnoittelee puhelut ja hankkii ensimmäisen arvion puhelun hinnasta, joka perustuu luokitteluun ja hinnastoon. ARD huolehtii prosessoidun datan toimituksesta asiakkaan laskutussysteemiin [4].

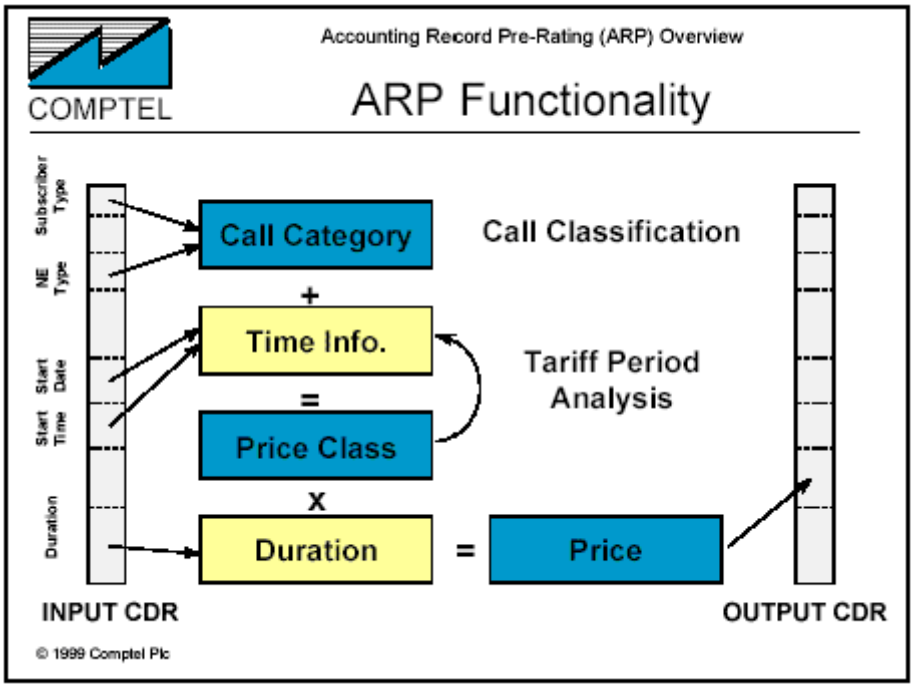


Kuva 5. MDS/AMD arkkitehtuuri [4].

4.3.1 Keräily-, esihinnoittelu- ja siirtomoduuli

Keräilymoduuli ARC kerää keskuksista dataa laskutusta varten. ARC on luotettava tiedoston kerääjä, joka saa selville kadonneet sekä kaksi kertaa saapuneet tiedostot. Jokaiselle keskustyypille on oma moduulinsa. Keskustyypit ovat AXE, DX, DMS, EWSD, Aethos, Aldiscon ja Comverse. Keskustyyppien siirtoprotokollat vaihtelevat keskuksittain. Mm. AXE käyttää siirtoprotokollaa MTP (Message Transfer Protocol), Nortelin DSM-keskus käyttää FTAM-protokollaa (File Transfer, Access, and Management standard) ja Nokian DX-keskus tukee OSI:n FTAM-, OSS/Connection Server – ja FTP-protokollaa. [7]

ARP-moduuli hinnoittelee puhelutietueet tiettyjen sääntöjen mukaan, ja säilyttää hintatiedot asiakkaalle palautusta varten. Puhelun hinta lasketaan puhelun luokittelun, hinnastoerittelyn ja hintastrategian mukaan. Hinnoittelusääntöjä ylläpidetään MDS-käyttöliittymän kautta. ARP luokittelee puhelut yhdistelemällä kenttiä tulevan datan tietueessa (input CDR). Hintaluokka määritellään puhelun aloituspäivän ja –ajan analysoinnilla (kuva 6). Jos puhelun kesto ylittää määritellyn ajanjakson, edellinen askel toistetaan. Määritelty hinta kirjoitetaan lähtevään datan tietueisiin (output CDR). Puhelut jaetaan erilaisiin ryhmiin: GSM-puhelu toiseen saman verkon GSM-puhelimeen, GSM-puhelu toisen verkon GSM-puhelimeen ja GSM kiinteään linjaan. Jokainen katekori jaetaan ajanjaksoihin hintojen mukaan. Ajanjakso on jaoteltu työpäiviin, lauantai- ja sunnuntaipäiviin sekä kellonaikoihin. Jokaisen puhelun hinta muodostuu puhelun ryhmän ja hinta-ajanjakson yhdistelmästä.



Kuva 6. Yleiskuvaus puhelun hinnanmuodostumisesta[8]

ARP tukee erilaisia hintasääntöjä, jotka perustuvat kolmeen periaatteeseen: hinta per puhelu, hinta per minuutti ja hinta per pulssi. Hinnoittelussa voidaan käyttää myös näiden kolmen yhdistelmää. ARP ohjelma suoritetaan yhdessä ARM-metodin komentojonossa. ARP:iin tuleva tiedosto on ARM:sta lähtevä tiedosto. [8]

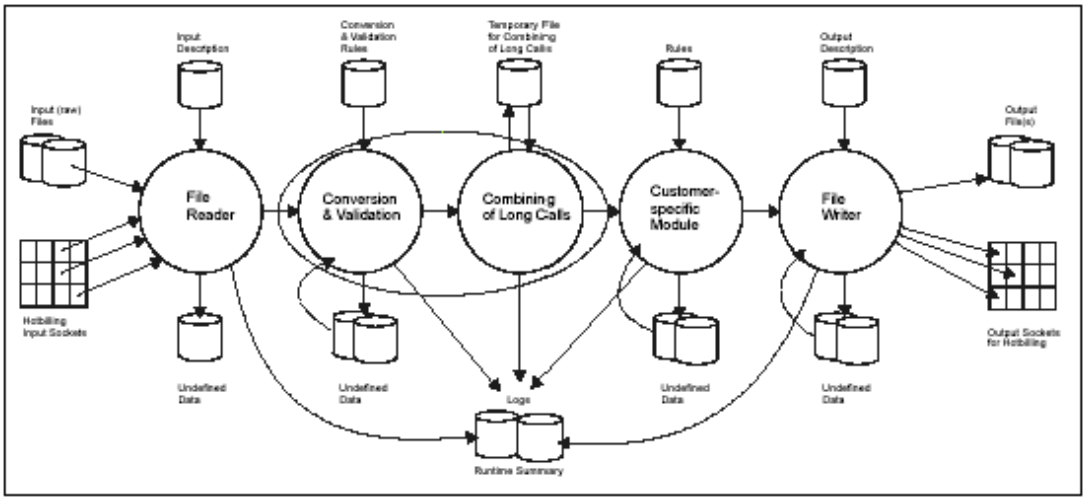
ARD huolehtii datan siirtämisestä asiakkaan laskutussysteemiin. Kun prosessointi ja esihinnoittelu tiedolle on suoritettu, ARD siirtää prosessoidun datan eteenpäin. Laskutussysteemi on joko samalla tai eri tietokoneella, kuin puheludatan prosessointi. ARD ohjelma suoritetaan ARM metodin komentojonossa.

4.3.2 Puhelutietueiden muokkaus –moduuli

ARM on itsenäinen MDS:n osa. ARM:n tarkoitus on tuottaa standardi liityntäpinta puhelutietojen prosessointiin. ARM sallii tiedon käsittelyn pieninä palasina erilaisissa verkkoratkaisuissa. Kun ARM vastaanottaa dataa erilaisista pu-

helinkeskuksista, se muokkaa datan annettujen sääntöjen mukaan. ARM muuttaa tietueen sopivaan muotoon asiakkaan laskutussysteemiä varten.

ARM on modulaarinen kuten myös sen rakenne itsessään. ARM muodostuu lukuisista komponenteista ja pienemmistä moduuleista, joista jokainen on erikoistunut erityiseen tehtävään. Alisysteemin komponentit kommunikoivat keskenään muodostaen itsenäisen prosessiketjun (kuva 7). Jokainen prosessi vastaanottaa tietoa edellisestä ketjun osasta, ja kun komponentti on lopettanut materiaalin prosessoinnin, se etenee seuraavaan komponenttiin. Edellisestä komponentista lähtevä tieto muodostuu seuraavan komponentin tulevaksi tiedoksi. Standardien komponenttien lisäksi ARM sallii myös asiakkaan omat, erityiset komponentit. Nämä komponentit tyypillisesti suorittavat erityisiä tehtäviä, kuten tilastotietojen keräilyä. [4, s. 7.].



Kuva 7. MDS/ARM komponenttien muodostama ketju.[4]

Peruskomponentteja ovat File Reader (FR), Conversion and Validation (CV) ja File Writer (FW) (kuva 8). Suomenkieliset nimet voisivat olla tässä yhteydessä tiedoston lukija, tiedoston muokkaus ja tiedoston kirjoittaja.

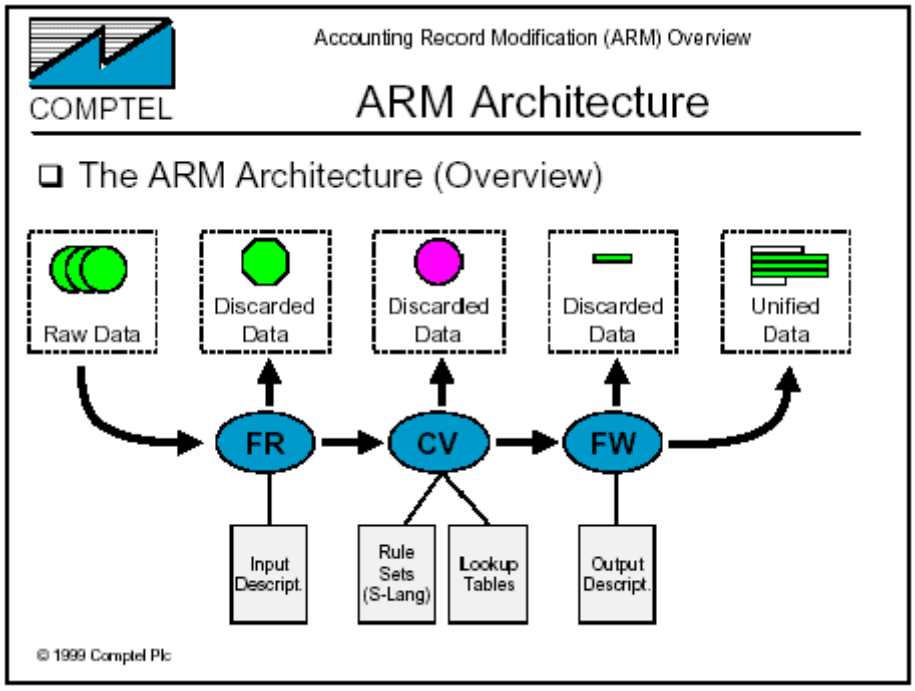
FR:n tehtävä on lukea ja dekoodata tuleva data. FR aloittaa lukemalla tulevan tiedon kuvaustiedoston ja tarkistamalla sen konfiguroinnin. Sen jälkeen FR lukee input–tiedoston ja dekoodaa dataa. Dekoodattu data on pilkotaan pienemmiksi tietopaketeiksi, kuten tietueiksi ja tietokentiksi. Kerätyt ja käännetyt tietueet

yhdistetään ja lähetetään seuraavaan prosessiin. FR jatkaa kunnes kaikki data on luettu tai virhe on havaittu [4,s. 7] .

CV-moduuli käyttää sisääntulevana tietona tiedoston lukijalta tulevaa dataa. CV muokkaa datan aluksi haluttuun muotoon, joko ASCII, binääriseen tai johonkin muuhun standardi-muotoon. Sen jälkeen data joko hylätään tai hyväksytään muokattavaksi. Erityiset hyväksymis- ja muutossäännöt ovat talletettu omaan tiedostoonsa. Tuloksena syntyy yhtenäistä dataa, joka jatkaa seuraavaan komponenttiin. Seuraava komponentti voi olla joko FW tai jokin asiakkaan oma erityinen komponentti [4, s.8].

Pitkien puheluiden yhdistäminen (Combining of Long Calls) on valinnainen ominaisuus, joka voidaan sisällyttää CV-moduuliin. Joskus puhelun kesto voi olla niin pitkä, että se ei mahdu yhteen CDR:ään (puhelutietue). Puhelu koostuu useista tietueista, osatietueista, jotka tallennetaan väliaikaiseen tiedostoon kunnes viimeinenkin puhelun osa on vastaanotettu. Sen jälkeen tietueet yhdistetään yhdeksi tietueeksi, ja data muokataan annettujen sääntöjen mukaan ennen lähetystä seuraavaan moduuliin [4, s. 8].

FW komponentti lukee prosessoidut tietueet, jotka se on vastaanottanut CV-moduulista. FW käyttää ulkoista tiedostoa (Output Description) määrittäessään CDR:ien rakennetta ja ulkoasua. Kun FW on vastaanottanut kaikki tietueet, se etsii vastaavaa muotoa tästä kuvaustiedostosta ja kirjoittaa lähtevän tietueen sen mukaisesti. Tietueet kirjoitetaan yhteen tai useampaan output –tiedostoon. [4, s. 8].



Kuva 8. ARM:n peruskomponentit: FR, CV ja FW. [9]

GRC-parametrit ovat muuttujia, jotka määrittelevät MDS/ARM konfiguraatiota. GRC-parametrien arvot, jotka sallivat komponenttien ominaisuudet määritellään asennusvaiheessa. GRC-parametrit tallennetaan GRC-tiedostoon, joka on jaettu GRC-ryhmiin. Jokaisella ohjelmalla on oma ryhmänsä, jota se käyttää ohjelmaa ajettaessa.

ARM:iin liittyvät tiedostot

ARM tuottaa ajon aikana erilaisia lokitiedostoja, jotka sisältävät informaatiota ajon suorituksesta. Lokitiedostot varastoidaan yleensä sille varattuun yleiseen hakemistoon, jonka alihakemistoiksi on luotu kunkin verkkoelementtityyppi.

ARM prosessin ajoon liittyy useita datatiedostoja; input-tiedosto, tulevan datan kuvaustiedosto, CV–tiedosto, lähtevän datan kuvaustiedosto, output-tiedosto sekä lokitiedosto hylätylle datalle.

Input-tiedosto sisältää verkkoelementeiltä vastaanotetun datan, jota jatkossa käsitellään ja muokataan laskutusjärjestelmää varten. Eri puhelinkeskuksista tuleva data on erimuotoista. Keskuksista tuleva raakadata koodataan ja määritellään tietyn formaatin, tulevan datan kuvauksen mukaan. Tulevan datan

kuvaustiedosto kuvaa tulevan datan kenttien ja tietueiden rakenteen. FR käyttää kuvaustiedostoa dekoodatessaan dataa.

CV–tiedosto sisältää parametreja ja sääntöjä, joita CV-komponentti käyttää ajaessaan vastaanotettua dataa FR:ltä. Säännöt ovat asiakkaan haluamat ja hyväksymät. Muutossäännöt (Conversion Rules) muokkaavat datasta juuri sellaisen kuin laskutusjärjestelmä vaatii. Säännöt perustuvat sekä tulevan datan että lähtevän datan -kuvaukseen. Kelvolliseksi toteamissäännöt (Validation Rules) ovat sääntöjä, joiden avulla kelpuutetaan tuleva data ARM:iin. Siinä määritellään tulevan datan kentille säännöt, joita noudatettuaan kentän tiedot hyväksytään prosessoitavaksi. Vaikka tulevaa dataa ei hyväksytäkään, sitä ei silti hylätä, vaan tietue menee systeemissä seuraavaan moduuliin.

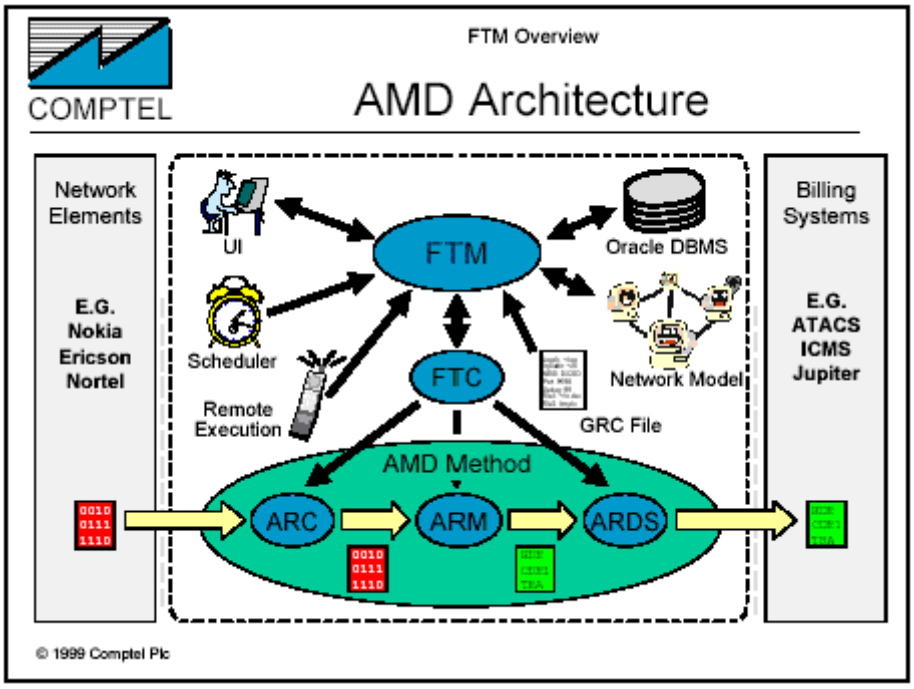
Lähtevän datan kuvauksessa määritetään jo prosessoidun datan muoto, kenttien nimet, pituudet, formaatti jne., jotta se olisi sopivaa laskutusjärjestelmään. FW käyttää tiedostoa ajaessaan dataa output-tiedostoon. Output-tiedosto sisältää prosessoidun, kelpuutetun datan. Haluttaessa käyttäjä voi määritellä syntyvän useita output–tiedostoja.

Lokitiedostot sisältävät tietoa datasta, jota ARM ei ole kyennyt tunnistamaan tai se on jostakin muusta syystä joutunut hylkäämään sen. Esimerkiksi tapauksessa jossa tietuetyypille ei ole kirjoitettu sääntöä, data kirjoitetaan määrittelemättömiin. Jokaisella komponentilla (FR, CV, FW) on omat lokitiedostonsa. Komponentti luo tiedoston määrittelemättömälle tai hylätylle datalle. Lisäksi komponentti luo tekstitiedoston, jossa on tietoa syystä, miksi data on kirjoitettu kyseiseen tiedostoon. Data on binäärimuotoista, ei luettavassa muodossa ilman erikoiskääntäjää, jotta komponentti ei pystyisi uudelleen koodaamaan dataa. ARM sisältää tämän erikoiskääntäjän, jota voidaan käyttää tarvittaessa [4, s. 9].

4.3.3 Prosessin valvonta –moduuli

FTM:ää voidaan kuvata AMD:n aivoina. Sen tarkoituksena on hoitaa puheludatan siirtoa verkkoelementistä tai toisesta tietokoneyksiköstä MDS systeemiin luotettavalla tavalla. Se voi myös kontrolloida useita muita prosesseja kuten tiedostojen liittämistä ja kopioimista. FTM ei salli tietojen katoamista ja se pystyy selvittämään virhetilanteita ja jossakin määrin elvyttämään niitä. Jos automaattinen elvytys ei ole mahdollista, systeemi lähettää manuaalisen hälytyksen, joka sisältää kaiken tarpeellisen tiedon, jotta systeeminhoitaja pystyy paikallistamaan virhetilanteen [4, s.5].

FTM valvoo tiedon keräys, muutos ja siirtoprosessien suoritusta yksinkertaisien metodien avulla. Metodit ovat joukko peräkkäin aseteltuja komentojonoja (step), jotka suoritetaan peräkkäin tai rinnakkain. Ensimmäinen komentojono kerää materiaalin ulkopuolisesta systeemistä, kuten puhelinvaihteesta ja toimittaa materiaalin seuraavalle komentojonolle. Tämä komentojono ehkä prosessoi materiaalin ja toimittaa joko seuraavalle komentojonolle tai ulkopuoliselle laitteelle, kuten asiakkaan laskutussysteemiin. Kuva 9. esittää FTM:n toimintaa [4, s.5].



Kuva 9. FTM kontrolloi datan kulkua AMD systeemissä [10]

FTM voi hoitaa monia yhtäaikaisia metodeja. Esimerkiksi kaksi metodia voivat olla keskinäisessä suhteessa toisiinsa niin, että ensimmäinen metodi muodostuu komentojonoista, jotka huolehtivat tietojen keräyksestä ja muutoksista. Sen muodostama tieto on toisen metodin tulevaa dataa. Toisen metodin komentojonot huolehtivat puhelun hinnoittelusta ja datan siirtämisestä laskutussysteemiin. Metodien ei aina tarvitse olla suhteessa toisiinsa, vaan siellä voi olla esimerkiksi kaksi itsenäistä puhelun keräystä samanaikaisesti. [11, s.6]

FTM muodostuu useista eri komponenteista: Method Manager, File Tranfer Controller (FTC) ja Method Scheduler. Method Manager vastaa yleisesti metodien suorituksesta. Se ottaa vastaan pyyntöjä joko ajastimesta tai käyttäjältä. Manager aktivoi metodin suorituksen pyyntöön perustuen. FTC vastaa prosessin aktivoimisesta ja monitoroinnista. Se käsittelee kaikki ulkoiset tapahtumat synnyttämällä komentojono–prosesseja. Komentojonot voivat olla joko unix-ohjelmia tai –skriptejä, joita FTM suorittaa. Ajastimen avulla voidaan käynnistää tiedoston siirtometodit automaattisesti. [11, s.9]

Metodit määritellään mallien (Templates) avulla. Mallit ovat yleisiä suunnitelmia, ja ne tarkentavat mitkä stepit kuuluvat mihinkin metodiin. Nämä suunnitelmat ovat käytössä kaikkiin uusiin metodeihin. Mallit ovat hyödyllisiä mm. tilanteessa, kun uusi olemassa olevan tyyppinen verkkoelementti yhdistetään verkkoon ja siihen määritellään metodi. Komentojonot yhdistetään toisiinsa linkkien avulla, jotka luodaan jokaisessa metodissa erikseen [11, s.13].

5 PROJEKTITYÖN TOTEUTUKSESTA YLEISESTI

Projektityö–nimitystä käytetään asiakastöistä, jotka kuuluvat mediaattoriohjelmiston tukipalvelun piiriin. Ympäri maailman sijaitsevat asiakkaat jättävät Change Request (CR) tai Problem Report (PR) pyyntöjä, jotka pyritään toteuttamaan mahdollisimman nopeasti. CR:t ovat kokonaisuuksia, jolloin muutoksia tehdään spesifikaatioon ja ohjelmakoodiin. PR:t ovat pienempiä kokonaisuuksia, jolloin muutoksia tehdään vain ohjelmakoodiin. Spesifikaatio on kirjoitettu oikein, mutta ohjelma ei toimi sen mukaisesti. PR:ien aikataulu on kiireellisempi. CR:t ja PR:t toteutetaan joko SAS, AMD tai AMD DB ympäristöissä.

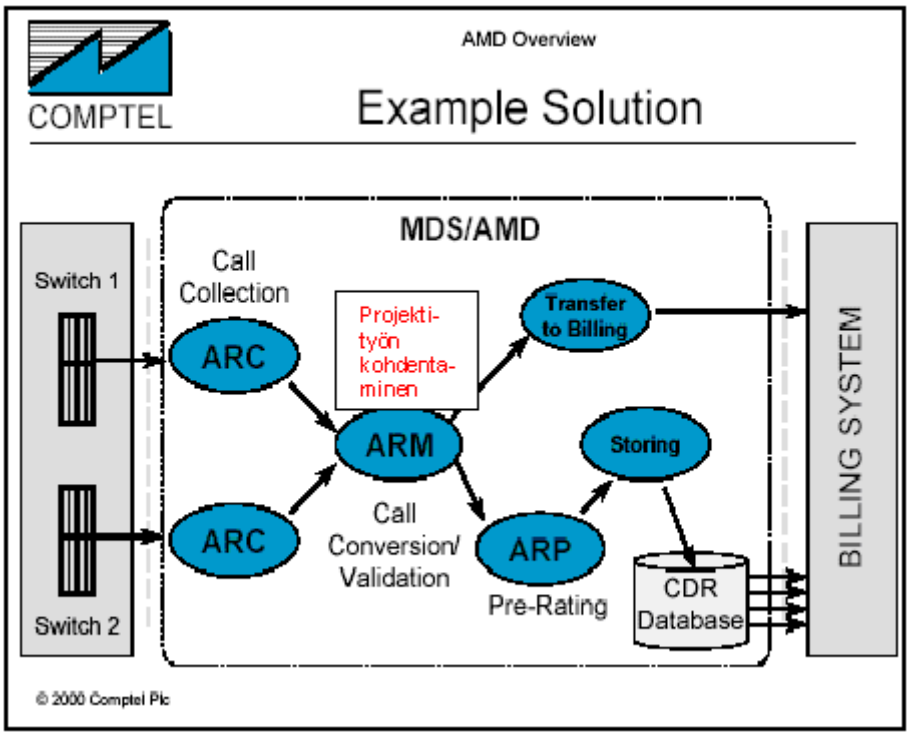
Työn suunnittelu aloitetaan joko spesifikaation kirjoittamisella tai päivityksellä. Asiakkaalle lähetetään dokumentti hyväksyttäväksi ennen ohjelmakoodin kirjoittamisen aloittamista. Asiakas voi joko lisätä haluamiansa tietoja dokumenttiin, tai hyväksyä sen sellaisenaan. Yleensä dokumentit kulkevat useamman kerran asiakkaan ja työn toteuttajan välillä ennen lopullista hyväksymistä. Dokumentin hyväksymisen jälkeen aloitetaan ohjelmakoodin kirjoitus. Koodi kirjoitetaan S-lang tai Expect –ohjelmointikielellä. Valmis koodi testataan ja kirjoitetaan testiraportti.

Ohjelmakoodista tehdään paketti lähetettäväksi asiakkaan testikoneelle ftp-yhteyttä käyttäen. Siihen liitetään erillisenä osana testiraportti sekä Release Notes, jotka lähetetään asiakkaalle sähköpostin välityksellä. Asiakas testaa tilatun työn, jonka jälkeen tehdään mahdolliset korjaukset tai muutokset. Kun asiakas hyväksyy lopullisen työn, työn toteuttaja asentaa ohjelmakoodin asiakkaan tuotantokoneelle.

Metodit asennetaan sekä asiakkaan testi- että tuotantokoneelle. Metodit rakennetaan pala palalta usein valmiita komentojonoja apuna käyttäen. Metodit suorittavat mm. puhelutietojen prosessoinnin ja siirtämisen laskutusjärjestelmään.

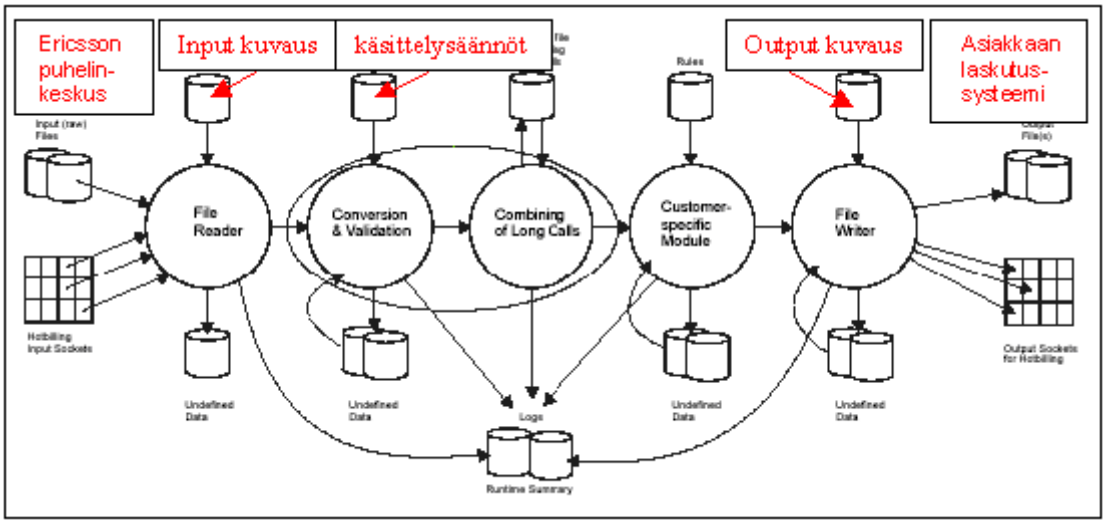
5.1 Projektityö MSC

Esimerkki projekti on suoritettu suurelle eurooppalaiselle teleoperaattorille Comptel Corporation toimesta. Asiakas tilasi uuden säännöstön GSM-puhelin (MSC) ja tekstiviestijärjestelmän (SMSC) laskutustansa varten. Tämä työ käsittää siitä GSM osuuden. Projektityö toteutetaan CR:nä ja se keskittyy AMD–ympäristöön ja siellä ARM–moduuliin (kuva 10).



Kuva 10. Projektityön toteutus tapahtuu AMD:n ARM–moduulissa.

Matkapuhelinkeskuksesta tulevat puhelut täytyy käsitellä tietyn formaatin mukaan, jotta ne soveltuisivat asiakkaan olemassa olevaan laskutusjärjestelmään (kuva 11). Aikaisemmin siihen järjestelmään ei ole tullut kyseisestä keskuksesta dataa laisinkaan. Työn tulevan datan kuvaus oli sama, mitä Comptel on jo aikaisemminkin käyttänyt. Siihen löytyi valmis tiedosto (Input Description), joten sitä ei tarvinnut luoda uudelleen. Lähtevän datan kuvaus luotiin kokonaisuudessaan uusi, josta johtuen myös käsittelysäännöt jouduttiin suunnittelemaan alusta lähtien.



Kuva 11. Ericssonin matkapuhelinkeskuksesta tuleva data käsitellään asiakkaan laskutussysteemiin sopivaksi.

6 PROJEKTITYÖN SUUNNITTELU

Työn suunnittelu aloitettiin, kun asiakkaalta saatiin virallinen tilaus. Suunnitteluvaihe käsitti lähinnä uuden MDS/ARM dokumentin tekemistä. Comptelilla on käytössä laatujärjestelmän mukaan mallipohjat, jonka pohjalle dokumentti kirjoitettiin. Asiakkaalta saatujen tietojen perusteella tehtiin ensimmäinen spesifikaatio -versio, joka lähetettiin asiakkaalle hyväksyttäväksi. Hyväksytyn spesifikaation versionumero on 1.0.1, ja sen sisällysluettelo löytyy liitteestä B.

MDS/ARM:iin tuleva data on peräisin Ericsson MSC –keskuksesta. Data on muodoltaan Common Charging Output ASN.1 –dokumentin mukainen, joka on yli sata sivua pitkä dokumentti. Seuraavassa kappaleessa siitä on esitelty pieni osa.

6.1 Tulevan datan kuvaus

Tulevan datan kuvaustiedosto kuvaa saapuvan datan tietoja. Kaikki puhelutiedot tulevat keskuksista tietueina (CDR). Ne voivat olla yksittäisiä puhelutietueita (Single Records) tai useita yksittäisiä puhelutietueita peräkkäin tietyssä järjestyksessä (Composite CDR). Puhelutieto muodostuu taas yhdestä puhelumoduulista (Call Module) ja mahdollisesta tapahtumamoduulista (ei pakollinen). Puhelumoduuli on osa puheluun liittyvästä datasta. Tapahtumamoduuli on puhelutietoon liittyvä tapahtuma, kuten lisäpalvelu, jolla on itsenäinen ajankesto [12]. Liitteessä C on kuvattu CDR:n rakenne ja liitteessä D Composite CDR:n rakenne.

Puhelumoduulit, toiselta nimeltään tietuetyypit (Record Types), joita käsitellään input –kuvauksessa ovat taulukossa 1 ja 2 Liitteen C tietuetyypit ovat yksittäisiä puhelutietueita ja liitteen D tietuetyypit ovat yhdistelmiä kahdesta eri tietuetyypistä, ja liittyvät CAMEL–palveluun (ks. 3.2). Ericsson MSC –keskuksessa esiintyy myös muita tietuetyyppejä, joita ei asiakkaan laskutusjärjestelmään suodateta lainkaan. Niistä ei ole myöskään mainintaa taulukossa.

Taulukko 1. Ericsson MSC –keskuksesta tulevat tietuetyypit

Record Type
Transit
MS Originating
Roaming Call Forwarding
Call Forwarding
MS Terminating
MS Originating SMS in MSC
MS Terminating SMS in MSC
SS Subscriber Procedure

*Taulukko 2. Ericsson MSC–keskuksesta tulevat CAMEL tietue tyyp*it

Record Type
MS Originating – Transit
Call Forwarding – Transit

6.2 Lähtevän datan kuvaus

Asiakas antoi dokumentin, jonka mukaan lähtevän datan kuvaus muodostettiin. Sen mukaan MDS/ARM täytyi suunnitella tuottamaan yhtenäinen tiedosto, joka on ASCII-muotoa ja koostuu aloitus- (Header) ja lopetustietueesta (Trailer) sekä useista puhelutietueista (taulukko 3). Tietueiden täytyi olla kiinteän mittaisia, ja ne erotetaan toisistaan rivinvaihto-merkillä. Kentät ovat numeerista tietoa. Jos kenttien arvot ovat lyhempiä kuin kentälle annetut pituudet, ARM täyttää kentät joko nollilla tai tyhjillä merkeillä. Numerokentät tasataan oikealle ja kirjainkentät tasataan vasemmalle.

Taulukko 3. ARM–moduulista lähtevän datatiedoston rakenne

Header
Call Data Record
Call Data Record
…
Trailer

Lähtevän datan kuvaukseen tietokenttiä tulee Header:ssä 19, Trailerissa 33 ja CDR:ssä 63. Jokaiselle tietuetyypille tulee CDR:stä samanlainen ulostulo, mutta niiden käsittelysäännöt voivat vaihdella hyvinkin paljon.

6.3 Kenttien hyväksymissäännöt (Validation rules)

Hyväksymissäännöt ovat sääntöjä, jonka avulla kelpuutetaan tuleva data ARM:iin. Säännöt sijoitettiin koodissa ennen muutossääntöjä heti tietokenttien sisään lukemisen jälkeen. Kentän yksityiskohtaisesta hyväksymissäännöistä on esimerkkejä taulukossa 4.

Taulukko 4. esimerkkejä hyväksymissäännöistä

Field	Rules
CalledPartyNumber	Value ≠ regexp “[0-9#A-F]+”
calledSubscriberIMSI	If field present and value ≠ regexp “[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9F]+”

6.4 Muutossäännöt (Conversion Rules)

Keskuksilta tuleva data prosessoidaan tiettyjen sääntöjen mukaan ennen kuin se luetaan output–tiedostoon. Jokainen ARM:sta lähtevä tietueen kenttä (CDR Field Name) luodaan tulevan datan kentän tai kenttien (raw CDR fields) ja muutossäännön (Conversion rule or value description) avulla. Muutossäännöistä on esimerkki taulukossa 5.

Taulukko 5. Esimerkki muutossäännöistä, jossa Charged-Party on ARM:sta lähtevän kentän nimi ja CalledPartyNumber on ARM:iin tulevan kentän nimi

#	CDR Field Name	Related raw CDR fields	Type and Length	Conversion rule or value description
1	Charged-Party	CalledPartyNumber	A18	Remove NPI & TON Left aligned- trailing spaces If CalledSubscriberIMSI starts with 27602 then convert the beginning of the telephone number (the part without NPI & TON) 1. If the telephone number starts with 35569, convert to 069 2. If the telephone number starts with 69, convert to 069 Example: convert 14691234567 to 0691234567

Muutossäännöt luotiin jokaiselle tietueen tyypin kentälle erikseen. Esimerkiksi Charged-Party–kenttä käsiteltiin lähes jokaisessa tietueen tyypissä eri tavalla ja tuleva datan nimi vaihteli myös tietuetyypin mukaan.

6.5 Pitkien puheluiden yhdistäminen

Pitkien puheluiden yhdistäminen sisällytettiin tähän projektityöhön, kuitenkin niin, että toiminto voidaan valinnaisesti joko kytkeä päälle tai pois päältä GRC-parametrin avulla.

Pitkän puhelun tietueita voi siis esiintyä lukuisia, jolloin osatietueilla täytyy olla jokin tunniste. Ericsson keskuksesta tulevassa datassa kentän nimi IdentificationNumber, jonka perusteella ne voidaan yhdistää samaksi puheluksi. Osatietueiden eri osat tunnistetaan partialOutputRecNum ja lastPartialOutput–kenttien avulla. Näitä kenttiä ei esiinnyt yksittäisissä (Single) tietueissa, joissa puhelu muodostuu kokonaisuudessaan.

LastPartialOutput–kenttä esiintyy tulevassa datassa vain jos tietue on viimeinen osa pitkän puhelun tietueesta. Silloinkin se esiintyy tyhjänä. Kenttä ei

esiinny laisinkaan yksittäisessä tai pitkän puhelun välitiedostossa (Intermediate). PartialOutputRecNum–kenttä esiintyy pitkän puhelun eri tietueissa sisältäen tiedon siitä mitä osaa puhelusta se edustaa. Jos partialOutputRecNum on 1, se on puhelun ensimmäinen tietue, jos se on suurempi kuin 1 esim. 2 tai 3, se on puhelun toinen tai kolmas tietue. Taulukossa 6 on pitkien puheluiden muutossääntö.

Taulukko 6. Pitkien puheluiden käsittelysääntö

PartialOutputRecNum	Last Partial Output	Explanation
field not present	field not present	Single (not long), will not be combined
1	field not present	First intermediate, first part of a long call
>1	field not present	Intermediate (normal part of a long call), not last part
>1	null i.e. the field is empty)	Last intermediate, this is the last part of a long call

Kun ARM havaitsee pitkän puhelun osatietueen, se varastoi tietueen sille varattuun hakemistoon ja väliaikaiseen tiedostoon. Seuraavan osatietueen tullessa kohdalle se tekee samoin, kunnes viimeinenkin on saapunut. Sen jälkeen ARM laittaa tietueen eteenpäin prosessoitavaksi. Osatietueet voivat tulla myös väärässä järjestyksessä, jonka ARM myös havaitsee, ja pystyy yhdistämään tietueet oikein.

7 PROJEKTITYÖN TOTEUTUS

Kun asiakas oli hyväksynyt spesifikaation, aloitettiin varsinainen työn toteutus eli ohjelmakoodin kirjoitus. Lopulliseksi spesifikaatio -versioksi tuli 1.0.7. Viimeisin niistä kirjoitettiin juuri ennen ohjelmakoodin tuotantokoneelle siirtoa. Viimeisimmät muutokset liittyivät mm. lähtevien ja tulevien kenttien vaihtamisiin. Koodi kirjoitettiin spesifikaation mukaiseksi S-lang -ohjelmointikielellä. Työ toteutettiin Comptelin Unix-kehityskoneella.

7.1 Kehitysympäristön luominen

Aluksi täytyy muodostaa kehitysympäristö unix–koneeseen, jos sitä ei jo ole kyseiselle asiakkaalle muodostettu. Tässä tapauksessa kehitysympäristö oli jo luotu osittain valmiiksi, koska asiakkaalle on tehty kehitystyötä aikaisemminkin. MDS/ARM kehitysympäristöä käytetään luotaessa asiakkaalle juuri heidän tarpeiden mukaista MDS/ARM ohjelmistoa: tulevan ja lähtevän datan kuvaukset, muutossäännöt jne. Kehitysympäristö on ClearCase VOB. Myös jokaisen asiakkaan erityinen VOB täytyy luoda kaikille MDS asiakkaille. VOB on ikään kuin hakemistorakenne unix–koneessa. Unix–systeemin hoitaja luo jokaiselle asiakkaalle oman käyttäjätunnuksen tai toiselta nimeltään projektitunnuksen. Mkvob–komento luo ClearCase:n vaatiman tietokannan, mutta hakemistoon ei vielä pääse kirjautumaan. Mkdir–komennolla tehdään ns. asennuspiste, joka on –tag –option niminen hakemisto. Lopuksi VOB asennetaan komennolla mount. Alla esimerkki kehitysympäristön luomisesta.

```
ct mkvob –tag /vobs/cust/panmd /vobstore/cust/panmd.vbs
mkdir /vob/cust/panmd
ct mount /vobs/cust/panmd
```

Lisäksi täytyy luoda projektille ClearCase näkymä (view). VOB:ssa olevat hakemistot näkyvät käyttäjälle vain, jos hänellä on näkymä päällä. Näkymä luodaan komennolla mkview. Näkymä oli myös jo valmiina aikaisempien projektien seurauksena. Alla esimerkki näkymän luomisesta projektille.

```
ct mkview –tag panmd_develop
/viewstore/shared/panmd_develop.vws
```

Seuraavaksi projektille luodaan oma hakemisto ja kehityskonfigurointi. Make-Project–komento luo hakemistorakenteen, joka sisältää symboliset linkit MDS/ARM:n komponentteihin ja hakemistoihin. Tämä osa täytyi tehdä, koska tehtävässä luotiin uusi verkkoelementti ja verkkoelementti tyyppi. Niiden nimiksi annettiin molemmille sama (R80_VFAL). Seuraavalla komennolla luotiin kehitysympäristö, jossa on keskustyypin R80_VFAL säännöstö voitiin kehittää:

```
MakeProject –d /vobs/cust/panmd/arm –n R80_VFAL –e
R80_VFAL,
```

missä –d tarkoittaa parametria hakemistolle, -n parametria verkkoelementti tyypille ja –e parametria verkkoelementille. Säännöstö tulee sijaitsemaan alihakemistossa /vobs/cust/panmd/mds/arm/config/R80_VFAL. MDS/ARM GRC–tiedosto eli arm.rc täytyy nyt sisältää luodut verkkoelementin ja verkkoelementtityypin. Kehitysympäristöä voitiin testata, vaikka itse säännöstö oli vielä tekemättä. Se testattiin arm_startup–komennnolla, joka käynnisti MDS/ARM prosessin.

```
arm_startup –d mod_desc  -e R80_VFAL –i 222 –f
/tmp/mytest/R80_VFAL/testmaterial,
```

jossa –e määrittelee verkkoelementin, -i FTMID:n tälle prosessille, –d mod_desc vaaditaan historiallisesta syystä sekä –f määrittelee ajettavan datan hakemiston ja nimen. Arm_startup sisältää kaikki edellä mainitut komponentit; FR, CV ja FW.

7.2 Ohjelmakoodin kirjoitus

Ohjelmakoodi kirjoitettiin neljään eri tiedostoon, jotka ovat nimeltään slangcdr.sl, slangcv.sl, functions.sl ja variableList.sl. Lisäksi tarvittiin tulevan ja lähtevän datan kuvaus tiedostot R80_JPT_in.desc ja R80_VFAL_out.desc. Arm.rc –tiedostoon piti lisätä tähän uuteen verkkoelementtiin tarvittavat lisäykset, sekä pitkien puheluiden ohjaustiedosto lc.dsc.

Arm.rc–tiedosto sisältää kunkin verkkoelementin nimet ja tyypit. Sieltä löytyvät myös GRC-parametrit sekä output–tiedostojen hakemistot ja nimet. Arm.rc–tiedostoon lisätyt rivit näkyvät kuvassa 12.

```
program: arm_fr
R80_VFAL:               $(DESCDIR)/R80_JPT_in.desc
                        # FR configuration for NE type R80_VFAL

program: arm_fw
R80_VFAL:               $(DESCDIR)/R80_VFAL_out.desc
                        # FW configuration for NE type R80_VFAL
header:                 $(ARMHOME)/output/$(NEID)/HDR.$(FTMID)
unified:                $(ARMHOME)/output/$(NEID)/UNIFIED.$(FTMID)

program: arm_CV
R80_VFAL:               $(DESCDIR)/slangcv.sl   # CV configuration
for NE type R80_VFAL
variablelist_L:         $(DESCDIR)/variableList.sl
functions_L:            $(DESCDIR)/functions.sl
R80_VFAL_long_call_combine: 1
                        # Accepted values are 1 (enable) or 0 (dis-
able)
jupiter_retail_visitors:1
                        # Accepted values are 1 (enable) or 0 (dis-
able)
rcf_VFAL_Jupiter: 1 # Accepted values are 1 (enable) or 0 (disable)
```

Kuva 12. Arm.rc:hen lisätyt rivit.

Tulevan datan kuvaus löytyi valmiina. Siihen täytyi ainoastaan muodostaa linkki unixin käskyllä, jotta se saatiin käyttöön myös tässä kyseisessä hakemistossa. Tulevan datan kuvaus on 29–sivuinen, ja siitä on kolme sivua esitetty liitteessä E. Tulevan datan kuvaustiedoston nimi on R80_JPT_in.desc. Liitteessä olevista sivuista käy ilmi mm. transit–tyyppisessä tietueessa olevien datakenttien nimet. Esimerkiksi calledSubscriberIMSI on kentän nimi, joka esiintyy jokaisessa tietuetyypissä (ks. 3.1).

Seuraavaksi luotiin tiedosto nimeltä slangcv.sl, siten että ohjelma saatiin ajettua MDS/ARM:ssa. Slangcv.sl–tiedostoon mm. kirjoitettiin raakadatasta mahdollisesti löydettävät tietuetyyppien määrittelyt. Slangcv.sl–tiedostoon kirjoitetaan myös GRC-parametrien määrittelyjä, muuttujien määrittelyjä, ajon aikaisia tulostuksia sekä Header- ja Trailer-tietueiden luomiset sekä niiden tietokenttien ulostulokomennot.

Slangcdr.sl–tiedostoon toteutettiin tietueiden ja niiden kenttien käsittelyt. Ensimmäiseksi kirjoitettiin ARM:iin saapuvien kenttien lukemiset. Alla on esimerkki käskystä kenttien hakemisesta prosessoitavaksi:

```
calledPartyNumber = CVGetInDataField_String("calledPartyNumber");
```

Kutakin kenttää kutsutaan vain kerran kyseisen puheludatan käsittelyn aikana. Sen jälkeen tutkitaan kentän data, onko se hyväksymissääntöjen mukaista. Kuvassa 13. on esimerkki dateForStartofCharge–kentän hyväksymissäännöstä.

```
if (dateForStartofCharge[0])
 {
    !if(string_match(dateForStartofCharge, "[0-9][0-9][0-1][0-9][0-
3][0- 9]",1))
    {
        () CVDiscardCurrentRec("Erroneous Date for Start of Charge");
        discarded++;
        return;
    }
 }
```

Kuva 13. Esimerkki hyväksymissäännöstä

Esimerkissä tutkitaan aluksi onko kenttä dateForStartofCharge olemassa, ja jos on, tutkitaan kentässä tulevan datan formaatti. Jos datan formaatti ei ole kuusi merkkiä pitkä niin, että ensimmäinen numero on väliltä 0 – 9, ja toinen 0 – 9 ja kolmas 0 – 1 ja neljäs 0 – 9 ja viides 0 – 3 ja kuudes 0 – 9, niin data hylätään, ja ohjelma palaa alkuun tutkimaan seuraavan tietueen ensimmäistä kenttää. Jos formaatti on edellisen mukainen, kenttä hyväksytään käsiteltäväksi.

Käsittelysääntöjen (Conversion and Validation Rules) rakentaminen jokaiselle tietuetyypille erikseen vaati eniten aikaa koko toteutusvaiheessa. Liitteessä F on esimerkki tariffClass–kentän käsittelysäännöstä. Kenttää käsiteltiin tietuetyypeittään (Record_type).

Esimerkissä käsitellään kaikkien muiden tyyppien tariffClass–kenttää paitsi sSSubsProc -tietuetyypin. TariffClass–kenttä on siis luettu jo ohjelman alussa sisään, ja se on heksaluku muotoa. Kenttä muokataan nyt heksaluvusta alphanumeeriseksi ja sen jälkeen alphanumeerinen luku desimaaliluvuksi. Sen jälkeen aletaan tutkia kenttää tietuetyypeittäin, ja tehdään muutokset speksin mukaan. Ensin tutkitaan tietuetyyppi transit. Jos tariffClass on pienempi kuin 150 tai suurempi kuin 180, tietue jätetään käsittelemättä. Sitten tutkitaan onko kenttä 180, ja jos on tutkitaan chargeableDuration-kentän arvo. Jos se on pienempi tai yhtä suuri kuin 10, tietue jätetään käsittelemättä. Muissa tapauksissa transit–tyyppisen tietueen tariffClass-kenttä tulostetaan sellaisenaan desimaalilukuna. Käsittelysääntö jatkuu muiden tietuetyyppien kohdalla vastaavasti.

Liittessä G on esitelty pieni osa pitkien puheluiden käsittelystä. Siinä tutkitaan löytyykö tietueesta partialOutputRecNum–kenttää, ja jos löytyy muuttuja partialRecNumberPresent asetetaan ykköseksi, ja partialOutputRecNum-kenttä muutetaan numeerisesta desimaaliluvuksi. Jos kentän arvon pituus on suurempi kuin yksi, se hylätään. Jos partialOutputRecNum–kenttää ei esiinny, muuttuja PartialRecNumberPresent asetetaan nollaksi. Tutkitaan esiintykö lastPartialOutput–kenttä. Jos lastPartialOutPresent esiintyy, asetetaan se ykköseksi, muuten nollaksi. Sen jälkeen tutkitaan mikä osatietueen osa tietue on.

ARM:sta ulostulevaa tiedostoa varten täytyi luoda aivan uusi lähtevän datan kuvaus –tiedosto (liite H). Siinä on kuvattu kaksi tiedostoa ja kolme eri tietuetyyppiä. ARM tuottaa kaksi erillistä tiedostoa, otsikko- ja datatiedoston (File header ja File unified). Syy miksi otsikko tietuetta ei sisällytetty suoraan datatiedoston alkuun on siinä, että otsikkotietueeseen haluttiin viimeisimmän puhelun päivämäärä. Se saadaan palautettua vasta sitten kun kaikki puhelu-

tietueet on luettu, ja jos otsikkotietue halutaan tulostaa alkuun, se täytyi ensin tulostaa omaan tiedostoonsa, ja sitten yhdistää se lopuksi datatiedoston alkuun. Datatiedoston loppuun tulostetaan suoraan lopetustietue. Otsikko- ja datatiedostot yhdistetään yhdeksi tiedostoksi metodeissa.

Seuraavaksi luotiin uusi lähtevän datan –tietue nimeltä DETAIL_REC. Se luotiin komennolla:

```
() = CVStartNewOutRec("DETAIL_REC");
```

Prosessoidut datakentät voidaan nyt lisätä datatietueeseen seuraavanlaisella komennolla:

```
() = CVAddOutDataField_String("CHARGED_PARTY",calledPartyNumber, 0,
ASCII_TYPE);
```

Kun ohjelman koodausvaihe oli suoritettu, aloitettiin testisuunnitelman laatiminen. Käsitys siitä oli jo muovautunut koodauksen aikana.

8 PROJEKTITYÖN TESTAUS

Testausvaihetta vaikeutti asiakkaalta saadun testimateriaalin puute. Testimateriaalina kuitenkin käytettiin dataa, joka tulee Ericssonin keskuksesta. Aineistoa oli niukasti, eikä sieltä löytynyt kaikkia mahdollisia tapauksia, joita ohjelman testaus olisi vaatinut.

Testiraportin suunnittelussa käytettiin Comptelin valmista mallipohjaa. Testiraportin pääotsikot ovat seuraavanlaiset:

```
1  About This Document
2  Overview of the Tests
3  Acceptance of the Executed Tests
4  Test Scenarios
```

Kohta 4 sisältää varsinaiset testitapaukset. Testitapaukset jaettiin kolmeen osaan: ulostulon tarkistukseen, otsikko-, data- ja lopetustietueiden muutossääntöjen testaukseen sekä pitkien puheluiden testaukseen (Output Layout, Header, Data and Trailer Records unified layouts and conversions ja Long Call Combining).

Ulostulon tarkistuksessa jokainen ulostuleva kenttä tarkistettiin erikseen. Jos kentän pituus sekä tasaus oikeaan tai vasempaan oli dokumentin mukainen, testi merkittiin hyväksytyksi testiraporttiin (taulukko 7).

Taulukko 7. Testiraportissa kohta ulostulon tarkistus

ID	Test Case Description (field name)	OK/Fail Date
83	Customer-No	OK 26.6.01/LE

Otsikko-, data- ja lopetustietueiden muutossääntöjen testaus sisältää myös kaikkien ulostulevien kenttien testauksen. Ulostulevien kenttien muutossäännöt muuttuvat tietuetyypeittäin, joten kaikkien tietuetyyppien kaikki kentät testattiin erikseen, ja merkattiin testiraporttiin. Taulukossa 8 on CallForwarding–

tyyppisen tietueen Charged-Party–kentän testaus, jossa sisään tuleva kenttä on RedirectingNumber. Sen arvo on 14944300251. Muutossääntöjen jälkeen, kun data lähtee ARM:sta, se on muutettu luvuksi 944300251. Muutossääntö on dokumentin mukainen, joten testaus merkittiin hyväksytyksi.

Taulukko 8. CallForwarding–tyyppisen tietuetyypin Charged-Party–kentänkäsittelysäännön testaus

ID	Test Case Description	Key fields and values for test data	Expected results and items to be checked, comments of test results	OK/Failed Date
171	Charged-Party	RedirectingNumber "14944300251"	944300251 3510 00111012170100000100480000000000000000000000000000000000 000000000000000 0P N0000	OK 27.6.01/LE

Pitkien puheluiden testaus suoritettiin hyvin pienellä aineistolla. Aineistosta löytyi vain yksi testauskohde, jossa puhelu sisälsi kaksi osatietuetta. Ensimmäinen tietue sisälsi PartialOutputRecNum–kentän, joka oli arvoltaan ”1”, ja toinen tietue sisälsi PartialOutputRecNum–kentän arvoltaan ”2” sekä lastPartialOutput–kentän arvoltaan tyhjä. Näin saatiin testattua, että muutossääntö toimi tältä osin ja yhdisti osatietueen yhdeksi tietueeksi. Tietueet tulivat ARM:iin oikeassa järjestyksessä, siten että ensin tuli ensimmäinen osa puhelusta ja vasta sitten viimeinen osa puhelusta. Testitapausta, jossa tietueet olisivat tulleet siten, että ensin olisi tullut puhelun viimeinen tietue, ja vasta sen jälkeen puhelun ensimmäinen tietue ei aineistosta löytynyt. Asiakkaan niin halutessa, tapaus testataan myöhemmässä vaiheessa. Taulukossa 9 on esitetty pitkien puheluiden testaus.

Taulukko 9. Testiraportin kohdasta pitkien puheluiden testaus

ID	Test Case Description	Key fields and values for test data	Expected results and items to be checked, comments of test results	OK/Failed Date
285	All partials found			
285.1	First and last partials found. Time slip between partials is ≤ max_difference.	PartialOutputRecNum "1" PartialOutputRecNum "2" lastPartialOutput ""	Partials combined as long call	OK 29.6.01/LE

Testitapaukset hyväksyttiin kohdassa Log of Acceptance (taulukko 10). Hyväksymisen suorittaa testaaja itse. Lopuksi koko testiraportin hyväksyy projektipäällikkö kohdassa Acceptance Record (taulukko 11) ennen asiakkaalle luovuttamista.

Taulukko 10. Testiraportin kohdasta hyväksyntä

Log of Acceptance of the Above Scenario			
X	Accepted / Accepted with Comments / Rejected	Comments	Date & Initials
	Accepted		29.6.01/LE

Taulukko 11. Koko testiraportin hyväksyy projektipäällikkö

Test Acceptance Sign-Off (Filled in by the Person who Approves the Tests)	
	Accepted
	The tests are fully accepted
X	The tests are accepted with the following comments
Comments/Additional Information	
Vodafone Albania specific data was not available. Existing and modified data was used. Some cases could not be tested at all because of lack of data.	
Date	Signature
29.6.2001	Heikki Holopainen

.

9 PROJEKTITYÖHÖN LIITTYVÄT METODIT

Metodit tehtiin ensimmäisessä vaiheessa asiakkaan testikoneelle, ja lopuksi Comptelin toimesta asiakkaan tuotantokoneelle. Metodit määritellään mallien avulla, ja komentojono-prosessit suorittavat skriptejä ja unixin komentoja (ks. 4.3.3). Näitä malleja ja komentojono-prosesseja löytyi asiakkaan testikoneelta jo valmiina. Skriptien ja unixin komentojen pienillä muutoksilla sekä komentojono-prosessien linkittämisillä saatiin aikaiseksi toimivat metodit.

Jo ensimmäisen testiraportin yhteydessä asennettiin ensimmäinen metodi asiakkaan testikoneelle. Se sisälsi vain kaksi komentojonoa: ARM ajon käynnistyminen ja tiedostojen liittäminen yhteen (otsikko- ja datatiedoston liittäminen). Kun testausvaihe eteni siihen vaiheeseen, että ruvettiin suunnittelemaan siirtoa tuotantokoneelle, asiakkaan pyynnöstä lisättiin vielä neljä muuta komentojonoa metodiin; keskuksesta dataa keräävä komentojono, DW:hen dataa kopioiva komentojono, output–tiedoston nimeävä komentojono sekä output–tiedoston siirtävä komentojono. Metodi perustuu malliin (Template) nimeltä ARM File Process to Jupiter_VFAL, joka on esitelty taulukossa 12. Mallin luonnin yhteydessä annettiin parametreja, jotka näkyvät taulukossa 13. Parametreista suurin osa on pakollisia (Obligatory), ja käyttäjä antaa niiden arvot metodin käynnistyksen yhteydessä.

Taulukko 12. Metodi perustuu malliin ARM File Process to Jupiter_VFAL, jossa on kuusi komentojonoa

Step #	Step Name
1	ARC AXE FTAM Collection
2	ARC Copy Files To DW
3	ARM 4 File Conversion and Validation
4	Join Jupiter_VFAL output files
5	ARM File Rename Jupiter_VFAL
6	ARD Jupiter Retail transfer

Taulukko 13. Mallin luontivaiheessa annetut parametrit

Name	Comment	Default Value	Obligatory / Optional
AUDIT	Use data flow audit trail (‘on’/’off’)	None	Obligatory
END_TIME_CHECK	ARM to check or not end times for CDRs (0/1)	None	Optional
REC_SEQ_CHK	ARM to check or not sequence numbering for CDRs (0/1)	None	Optional
arm_output_file	The temporary name of the file after the ARM process	/mds/amd/arm/output/$NEID/joined_output_file.$ftm_id	Obligatory
arm_raw_file	The name of the raw file	$infile	Obligatory
do_backup	ARM to do backup of raw and converted files or not (0/1)	None	Optional
Jupiter_user	Userid for the jupiter host	None	Obligatory
exehost	Host where method steps are executed	None	Obligatory
Infile	The name of the file to be processed	None	Obligatory
NEID	Switch symbolic name e.g. TP1PNA	None	Obligatory
RCFILE		/mds/.mds.rc	Obligatory

Taulukoissa 14, 16, 18, 20, 22 ja 24 on esitelty komentojonojen sisällöt. Osa niistä ovat valmiita unixin ohjelmia, ja osa skriptejä, joiden sisältöä ei ole esitelty sen tarkemmin tämän työn puitteissa. Komentojonon suoritukseen tarvittavat parametrit on lueteltu taulukoissa 15, 17, 19, 21, 23 ja 25. Jos parametri on Internal parameter, se on FTM:n muodostama parametri. Jos parametri on Method parameter, parametriin tulee metodissa annettu parametrin arvo.

Komentojono ARC AXE FTAM Collection (taulukko 14) kerää puheluaineiston Ericssonin keskuksesta, ja tuo sen MDS/ARM tuotantokoneeseen. Komentojonon suoritukseen tarvittavat parametrit on esitelty taulukossa 15.

Taulukko 14. Komentojono ARC AXE FTAM Collection suorittaa puheluaineiston keräilyn keskuksesta unixin skriptillä axefcp

Command	Parameters
/mds/arc/bin/axefcp	-i $ftm_id -e $NEID -u $axe_user_id

Taulukko 15. Komentojono ARC AXE FTAM Collection sisältää taulukon mukaiset parametrit

Parameter	Explanation	Value
-i $ftm_id	FTM generated job sequence number 0-99999	Internal parameter
-e $NEID	Symbolic switch id, TP1PNA, LR1LAR, ...	Method parameter
-u $axe_user_id	Switch user name	method parameter

Komentojono ARC Copy Files to DW kopioi keskukselta tulevan tiedoston Data Warehouse:iin (DW). Lisäksi, että puheluaineisto käsitellään ARM:ssa, se kopioidaan tilastotietoja, kehitystä yms. varten Data Warehouse:iin. Siellä se käsitellään DW:n säännöstön mukaan. Kopioiva skripti on esitelty taulukossa 16 ja tarvittavat parametrit taulukossa 17.

Taulukko 16. Komentojono ARC Copy Files to DW sisältää skriptin join_dw_files.ksh, joka suorittaa puhelutiedoston kopionnin DW:hen

Command	Parameters
/mds/arc/bin/join_dw_files.ksh	-i $ftm_id -e $NEID

Taulukko 17. Komentojono ARC Copy Files to DW sisältää taulukon mukaiset parametrit

Parameter	Explanation	Value
-i $ftm_id	FTM generated job sequence number 0-99999	internal parameter
-e $NEID	Remote host (switch) symbolic name symbolic switch id, TP1PNA_DW, LR1LAR_DW, ...	method parameter

Komentojono ARM 4 File Conversion and Validation suorittaa ARM prosessoinnin puheluaineistolle, ja se käynnistyy unixin ohjelmalla arm_startup (taulukko 18). Arm_startup:iin liittyvistä parametreista (taulukko 19) on myös mainittu aikaisemmin kohdassa työn toteutus.

Taulukko 18. Komentojono ARM 4 File Conversion and Validation sisältää unixin ohjelman arm_startup

Command	Parameters
/mds/amd/arm/bin/arm_startup	-i $ftm_id -e $NEID -d mod_desc -f $infile

Taulukko 19. Komentojono ARM 4 File Conversion and Validation sisältää taulukon mukaiset parametrit

Parameter	Explanation	Value
-i $ftm_id	FTM generated job sequence number 0-99999	internal parameter
-e $NEID	Symbolic switch id, TP1PNA, TP1PNA_DW, ...	method parameter
-d mod_desc	Static parameter	'mod_desc'
-f $infile	Filename with leading path of the file to process	method parameter

Otsikko- ja datatiedoston liittäminen yhdeksi tiedostoksi (ks 7.2) tapahtuu komentojonossa Join Jupiter_VFAL output files (taulukko 20). Taulukossa 21 on esitetty tarvittavat parametrit.

Taulukko 20. Komentojono Join Jupiter_VFAL output files sisältää unixin skriptin filejoin

Command	Parameters
/mds/ftm/bin/filejoin	-i $ftm_id -e $NEID

Taulukko 21. Komentojono Join Jupiter_VFAL output files sisältää taulukon mukaiset parametrit.

Parameter	Explanation	Value
-i $ftm_id	FTM generated job sequence number 0-99999	internal parameter
-o $joind_output_file	File to be joined	method parameter
-f $output_file	File out from filejoin	method parameter

Yhdistetty otsikko- ja datatiedosto sisältää nyt asiakkaan halutun tietuerakenteen (ks. taulukko 3). Tiedoston nimi muokattiin vielä asiakkaan laskutussysteemiin sopivaksi komentojonossa ARM File Rename Jupiter_VFAL (taulukko 22). Sen parametrien arvot on esitetty taulukossa 23.

Taulukko 22. Komentojono ARM File Rename Jupiter_VFAL sisältää unixin skriptin arm_file_rename_Jupiter_VFAL.ksh

Command	Parameters
/mds/amd/arm/bin/arm_file_rename_Jupiter_VFAL.ksh	-i $ftm_id -e $NEID –f $arm_output_file –t $arm_raw_file -b $do_backup

Taulukko 23. Komentojono ARM File Rename Jupiter_VFAL sisältää taulukon mukaiset parametrit

Parameter	Explanation	Value
-i $ftm_id	FTM generated job sequence number 0-99999	internal parameter
-e $NEID	symbolic switch id, TP1PNA_DW, LR1LAR_DW, ...	method parameter
-f $arm_output_file	The output file name of the ARM process	Method parameter
-t $arm_raw_file	The name of the original raw file	Method parameter
-b $do_backup	Decides if the files should be backed up or not	method parameter

Metodin viimeisenä komentojonona on ARM:n output–tiedoston siirtäminen asiakkaan haluamaan laskutuskoneeseen. Komentojono ARD Jupiter Retail Transfer suorittaako. tehtävän (taulukko 24). Sen parametrit on esitetty taulukossa 25.

Taulukko 24. Komentojono ARD Jupiter Retail Transfer sisältää unixin skriptin jpxfer_retail.ksh

Command	Parameters
/mds/ard/bin/jpxfer_retail.ksh	-i $ftm_id -e $NEID -u $jupiter_user_id

Taulukko 25. Komentojono ARD Jupiter Retail Transfer sisältää taulukon mukaiset parametrit

Parameter	Explanation	Value
$ftm_id	FTM generated job sequence number 0-99999	internal parameter
$NEID	Symbolic switch id, TP1PNA_DW, LR1LAR_DW, ...	method parameter
$Jupiter_user_id	Destination user name	method parameter

Lopuksi asetettiin itse metodille ARM File Process to Jupiter_VFAL parametrit (taulukko 26). Jos parametri on muotoa User, sen arvon voi käyttäjä antaa itse tai arvoksi tulee oletusarvo. Jos parametri on muotoa Fixed, arvo tulee oletusarvokentästä.

Taulukko 26. Metodin ARM File Process to Jupiter_VFAL:lle annetut parametrit

Name	Default Value	Change at startup
AUDIT	"on"	User
END_TIME_CHECK	"0"	User
REC_SEQ_CHK	"0"	User
arm_output_file	/mds/amd/arm/output/$NEID/joined_output_file.$ftm_id	Fixed
Exehost	"localhost"	Fixed

Name	Default Value	Change at startup
NEID	"<NEID>"	User
arm_raw_file	“$infile”	User
Infile	Value defined for every run	User
Jupiter_user	“bmdftam”	User
RCFILE	/mds/.mds.rc	Fixed
Do_backup		User

10 YHTEENVETO

Työn aihe oli mielenkiintoinen ja hyödyllinen, koska se liittyi palkkatyöhöni. Itse asiassa tämä oli ensimmäinen itsenäinen asiakastyöni, jonka olen Ebsolut Oy:ssä oloaikanani tehnyt. Aihe antoi heti alkuun kokonaisvaltaisen käsityksen tulevastakin työstä, koska tämä esimerkki projekti sisälsi kaikki työvaiheet: suunnittelu, toteutus, testaus ja metodit. Projektiluontoisena työt voivat olla esimerkiksi vain koodimuutos tai testausprojekti.

Olen tehnyt kaikki työvaiheet itsenäisesti. Ilman apua en olisi kuitenkaan selvinnyt, vaan olen voinut pyytää neuvoa kokeneilta työtovereiltani. Alkutiedot saatuani asiakkaalta, työ tuntui aluksi aika vaikealta toteuttaa. Projektipäällikön ja työtovereideni neuvokkuuden ansiosta pääsimme eteenpäin, ja kykenin työskentelemään itsenäisesti projektin aikana.

Työn tavoitteena oli tutustua mediaattoriohjelmistoon, joka käsittää laskutustietojen keruu- sekä palveluiden aktivointitoiminnot. Sitä käytetään laajalti kiinteissä, matkapuhelin-, satelliitti- ja dataverkoissa kaikkialla maailmassa. Projektityön tavoitteena oli toimittaa eurooppalaiselle teleoperaattorille ohjelmisto, joka käsittelee puhelinkeskuksesta tulevaa dataa ja siirtää sen laskutusjärjestelmään.

Projektityön suunnitteluvaihe oli lähinnä dokumentin kirjoittamista. Dokumentin sisältö muokattiin yhdessä operaattorin kanssa hänen tarpeittensa mukaan. Dokumentin sisältöön jouduttiin kuitenkin palaamaan sekä toteutus- että testausvaiheessa. Se ei ole kuitenkaan uuden säännöstön suunnittelussa mitenkään normaalista poikkeavaa.

Projektityön toteutusvaihe käsitti ympäristön luomisen ja ohjelmakoodin kirjoittamisen. Ympäristö luotiin VOB:iin Comptelin unix-kehityskoneelle. Ohjelmakoodi kirjoitettiin S-lang–ohjelmointikielellä. Ohjelmakoodi kirjoitettiin hyväksymissäännöille, muutossäännöille ja lähtevän datan kuvaukselle. Hyväksymissäännöt valikoivat halutun datan käsittelyyn. Puhelutietojen muutossäännöt muokkaavat tulevan datan asiakaslaskutukseen sopivaksi. Muutossääntö-

jen koodaus oli työvaiheista haastavin ja vei eniten aikaa. Lähtevän datan kuvaus antaa prosessoidulle datalle formaatin, jotta operaattorin laskutusohjelma pystyy käsittelemään laskutustietoja.

Projektityön testausvaihe sisälsi testauksen, jonka mukaan kirjoitettiin testiraportti. Operaattorilta saatiin testidataa, joka ei kuitenkaan vastannut aivan sitä, mitä puheluaineisto tulisi olemaan. Testiraportista käy ilmi, mitkä tapaukset on testattu, ja millä aineistolla. Aineisto ei sisältänyt kaikkia suunniteltuja testitapauksia.

Metodi asennettiin sekä testikoneelle että tuotantokoneelle. Metodi koostuu useista komentojonoista, jotka ovat joko unixin komentoja tai skriptejä. Metodi suorittaa puheluaineiston keräilyn keskuksesta, aineiston kopioimisen Data Warehouse:iin, ARM–prosessin käynnistyksen, ARM:sta ulostulevien tiedostojen yhteenliittämisen, tiedoston nimen muuttamisen ja tiedoston siirron laskutusjärjestelmään.

Projektityö antoi kokemusta myös sähköpostin käyttämisestä tiedonvälityksessä. Sähköpostin muotoilu vieraalla kielellä kohteliaaksi ja asialliseksi kiireellisessä aikataulussa ei aina ollut helppoa. Asiakkaan vaatimukset kasvoivat projektin loppuvaiheessa, joihin piti pystyä vastaamaan joko myönteisesti tai kielteisesti kohteliaisuutta unohtamatta. Kirjeenvaihtoa asiakkaan kanssa kävi suurimmalta osin kuitenkin projektipäällikkö.

Projektityö toteutui sovitussa aikataulussa. Ongelmana pidin ainoastaan sitä, että aikataulu tuntui olevan häilyvä, eikä sitä oltu tarkasti määritelty työn loppuvaiheessa. Euroopassa aikataulu käsitys tuntuu olevan erilainen. He haluavat, että lupaamme mahdottomia. Jos aikataulu myöhästyy ei haittaa niin paljoa, kuin se jos ei luvata. Suomalainen haluaa pysyä sovitussa aikataulussa, vaikka se on mahdotonkin.

Olen tyytyväinen tekemääni insinöörityöhön, ja sen kautta saamaani kokemukseen. Ohjelmisto on asennettu asiakkaan tuotantokoneeseen, eikä palautetta sen toimimattomuudesta ole saapunut. Tulevissa projekteissa tulen var-

masti tekemään monia asioita toisin. Esimerkiksi ohjelmakoodia kannattaa ja voi kirjoittaa niin, että sitä on seuraavissa projekteissa helppo muuttaa. Ilman harjoitusta ja kokemusta se ei kuitenkaan onnistu. Tämän työn aikana harjoitusta ja kokemusta tuli monelta osa-alueelta.

LÄHDELUETTELO

1 Antti Kiviluoto, Mikko Hautala, Ari Haapaniemi. Internet artikkeli teletekniikan harjoitustyöstä. [WWW-dokumentti]. <http://keskus.hut.fi/opetus/s38118/s00//tyot/15/>. (Luettu 29.8.2001).

2 Jyrki Penttinen, Tauno Wacklin.1999. GSM-Tekniikka Järjestelmän toiminta, palvelut ja suunnittelu, 1. painos. Porvoo. WSOY. 349 s. ISBN 951-0-24099-0.

3 Comptel Corporation. Introduction to the MDS Accounting Mediation Device (MDS/AMD). Monistesarja.

4 Comptel Corporation. 1999. MDS/ARM Release 3.6 Functional Description. Document Version 4.0.0. 21 sivua. Monistesarja.

5 John E. Davis. 1996. S-Lang Programmer’s Guide. Version 0.1. Monistesarja.

6 Comptel Corporation. MDS/SAS Training Material. 105 sivua. Monistesarja.

7 Introduction to the MDS Accounting Record Collection (MDS/ARC).

8 Introduction to the MDS Accounting Record Pre-Rating (MDS/ARP).

9 Comptel Corporation. 1999. Introduction to the MDS Record Modification (MDS/ARM). Monistesarja.

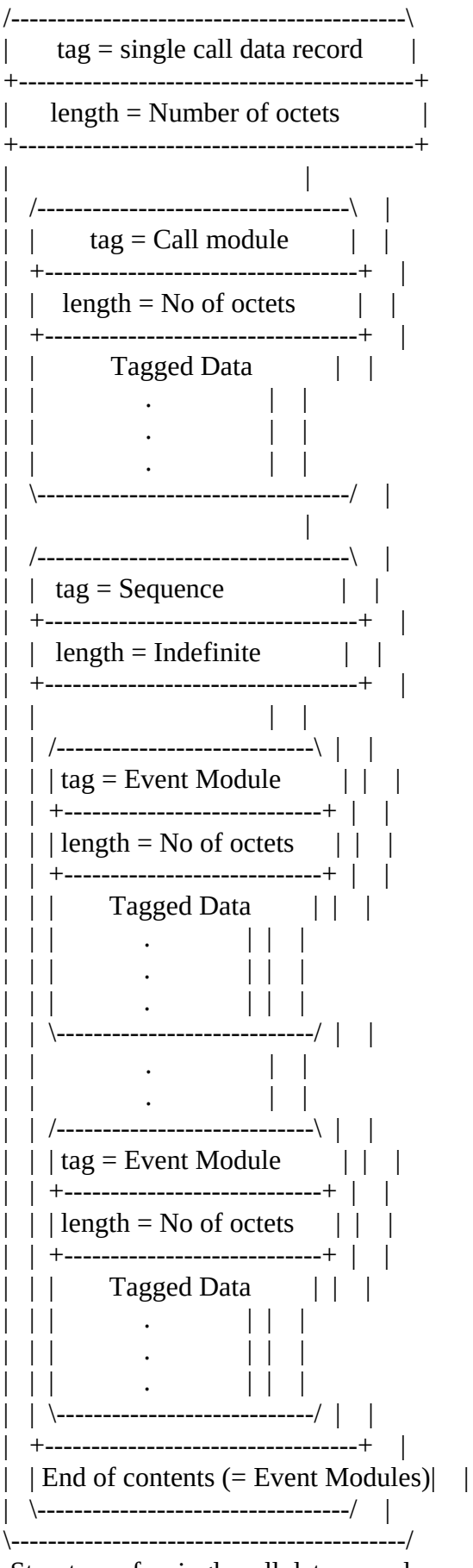
10 Comptel Corporation. Introduction to the MDS File Transfer Manager (MDS/FTM). 17 sivua. Monistesarja.

11 Comptel Corporation. MDS/FTM Release 3.3.7 Functional Description. 25 sivua. Monistesarja.

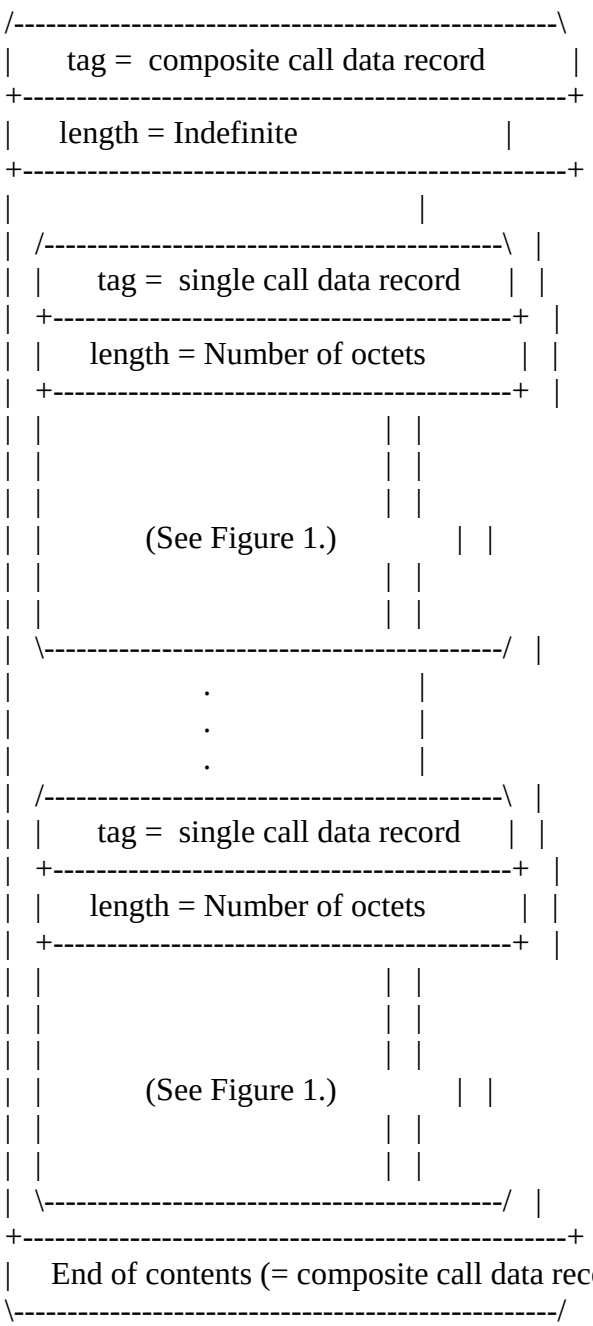
12 Comptel Corporation. COMMON CHARGING OUTPUT ASN.1 SS R7.0. Muutettu 26.4.2000. Monistesarja.

Taulukko 1. MS:n perustoimintoja ja –ominaisuuksia, jotka ovat pakollisia (P), vapaaehtoisia (V) tai pakollisia vain silloin kun liityntä käyttäjän ja puhelimen välillä on olemassa (P, kun MMI) [2, s. 63].

toiminto tai ominaisuus	**pakollisuus**
kutsutun numeron näyttö	P, kun MMI
puhelunmuodostuksen merkinannon ilmaisu	P, kun MMI
maa- ja verkkotunnuksen näyttö	P, kun MMI
palvelutason näyttö	P, kun MMI
maan ja verkon valinta	P
IMEI-koodi	P
SMS:n vastaanotto ME:llä, tallennus SIM:lle ja SMS:n ilmaisu	P
ilmaisu SIM-muistipaikkojen täyttymisestä lyhytsanomille	P
hätänumeroon soitto, mikäli TE tukee puhepalvelua	P
DTMF-toiminta puhelun aikana	P
tilaajan identiteetin hallinta (yhteydet mahdollisia, kun IMSI SIM:llä)	P
salausalgoritmien A5/1 ja A5/2 tuki	P
numeronäppäimistö (0…9, +, * ja # on voitava kuitenkin antaa)	V
lyhytsanoman näyttö	V
DTE/DCE-rajapinta	V
ISDN:n S-rajapinta	V
kansainvälinen ”+” –merkki, jolla korvataan ulkomaan suuntanumero	V
on/off-katkaisin	V
aliosoitteistus hakemistonumerolle	V
SMS-CB DRX (ei-haluttujen viestityyppien suodatus)	V
palveluntarjoajan ilmaisu (SIM:lle tallennettu 16-merkkienen tunnus)	V
lisäpalveluiden ohjaus	V
lyhytvalintanumero	V
soittomahdollisuus vain SIM-korttiin ennaltaohjelmoituihin numeroihin	V
lähtevien puheluiden esto	V
DTMF-ohjausmerkkien erotin	V
hakemistonumeroiden valinta lyhytsanomille	V
viimeksi valitun numeron toisto	V



Structure of a single call data record



. Structure of a composite call data record

```
#*******************************************************************************
# PROJECT: MDS/ARM HP Panafon
# Input description of R8.0
#
# $Source: /vobs/cust/panmd/mds/arm/config/R80_JPT/R80_JPT_in.desc $
# @(#)$Id: R80_JPT_in.desc /main/8 2001/07/20 13:32:09 khuhta $
#
# Copyright (c) Oy Comptel Ab 2000
#*******************************************************************************


# Ericsson AXE ASN.1 (BER) Call data modules

#byteorder MSB/LSB

Set {

      (Block) EOV     *0:'EOV' 80 2 {}
      (Block) EOF     *0:'EOF' 80 2 {}
      (Block) VOL     *0:'VOL' 80 2 {}
      (Block) HDR     *0:'HDR' 80 2 {}
      (Block) UHL     *0:'UHL' 80 2 {}
      (Block) UTL     *0:'UTL' 80 2 {}


      Block ChargingBlock * 2048 2 {

      Set {

            Block   GSMPLMNCDM+ BER:0xA0 BER 2 {

            Set {

            SetRecord transit=      BER:0xA0 BER 2 {
                  Field tAC                     BER:0x80 BER 2 ostr
                  Field callIdentificationNumber  BER:0x81 BER 2 ostr
                  Field recordSequenceNumber      BER:0x82 BER 2 ostr
                  Field typeOfCallingSubscriber   BER:0x83 BER 2 bin
                  Field callingPartyNumber        BER:0x84 BER 2 bcds
                  Field calledPartyNumber         BER:0x85 BER 2 bcds
                  Field calledSubscriberIMSI      BER:0x86 BER 2 bcds
                  Field disconnectingParty        BER:0x87 BER 2 bin
                  Field dateForStartofCharge      BER:0x88 BER 2 nstr
                  Field timeForStartofCharge      BER:0x89 BER 2 nstr
                  Field timeForStopofCharge       BER:0x8A BER 2 nstr
                  Field chargeableDuration        BER:0x8B BER 2 nstr
                  Field interruptionTime          BER:0x8C BER 2 nstr
                  Field timeFromRegSeizuToStofChg BER:0x8D BER 2 nstr
                  Field chargedParty              BER:0x8E BER 2 bin
                  Field originForCharging         BER:0x8F BER 2 ostr
```

```
                Field tariffClass               BER:0x90 BER 2 ostr
                Field tariffSwitchInd           BER:0x91 BER 2 bin
                Field numberOfMeterPulses       BER:0x92 BER 2 bin
                Field exchangeIdentity          BER:0x93 BER 2 ascii
                Field mSCIdentification         BER:0x94 BER 2 bcds
                Field outgoingRoute             BER:0x95 BER 2 ascii
                Field incomingRoute             BER:0x96 BER 2 ascii
                Field miscellaneousInformation  BER:0x97 BER 3 ostr
                Field originatedCode            BER:0x98 BER 2 ostr
                Field iNMarkingOfMS             BER:0x99 BER 2 bin
                Field callPosition              BER:0x9A BER 2 bin
                Field eosInfo                   BER:0x9B BER 2 ostr
                Field internalCauseAndLoc       BER:0x9C BER 2 ostr
                Field originalCalledNumber      BER:0x9D BER 2 bcds
                Field redirectingNumber         BER:0x9E BER 2 bcds
                Field RedirCounter              BER:0x9F1F BER 2 ostr
                Field redirectingDropBackNum    BER:0x9F20 BER 2 bcds
                Field redirectingDropBack       BER:0x9F21 BER 2 ostr
                Field restartDuringCall         BER:0x9F22 BER 2 ostr
                Field restartDuringOutInd       BER:0x9F23 BER 2 ostr
                Field iCIOrdered                BER:0x9F24 BER 2 ostr
                Field outputForSubscriber       BER:0x9F25 BER 2 bin
                Field lastPartialOutput         BER:0x9F26 BER 2 ostr
                Field partialOutputRecNum       BER:0x9F27 BER 2 bin
                Field relatedCallNumber         BER:0x9F28 BER 2 bin
                Field faultCode                 BER:0x9F29 BER 2 ostr
                Field subscriptionType          BER:0x9F2A BER 2 ostr
                Field incompleteCallDataInd     BER:0x9F2B BER 2 ostr
                Field incompleteCompositeCDR    BER:0x9F2C BER 2 ostr
                Field switchID                  BER:0x9F2D BER 2 ostr
                Field networkCallRef            BER:0x9F2E BER 2 ostr
                Field discSueSystemRec          BER:0x9F2F BER 2 ostr
# the field  forloppDurOutInd is closed in R7
#                Field forloppDurOutInd         BER:0x9F30 BER 2 ostr
                Field forloppRelDurCall         BER:0x9F31 BER 2 ostr
# New Fields that will be opened for R7
                Field translatedNumber          BER:0x9F32 BER 2 bcds
                Field bCSMTDPData1              BER:0x9F33 BER 2 ostr
                Field bCSMTDPData2              BER:0x9F34 BER 2 ostr
                Field bCSMTDPData3              BER:0x9F35 BER 2 ostr
                Field bCSMTDPData4              BER:0x9F36 BER 2 ostr
                Field bCSMTDPData5              BER:0x9F37 BER 2 ostr
                Field bCSMTDPData6              BER:0x9F38 BER 2 ostr
                Field bCSMTDPData7              BER:0x9F39 BER 2 ostr
                Field bCSMTDPData8              BER:0x9F3A BER 2 ostr
                Field bCSMTDPData9              BER:0x9F3B BER 2 ostr
                Field bCSMTDPData10             BER:0x9F3C BER 2 ostr
                Field gSMCallReferenceNumber    BER:0x9F3D BER 2 ostr
                Field mSCAddress                BER:0x9F43 BER 2 bcds
                Field reroutingIndicator        BER:0x9F79 BER 2 ostr
                Field redirectionCounter        BER:0x9F1F BER 2 bin
```

```
SetRecord mSOriginating=        BER:0xA1 BER 2 {
        Field tAC                       BER:0x80 BER 2 ostr
        Field callIdentificationNumber  BER:0x81 BER 2 ostr
        Field recordSequenceNumber      BER:0x82 BER 2 ostr
        Field typeOfCallingSubscriber   BER:0x83 BER 2 bin
        Field callingPartyNumber        BER:0x84 BER 2 bcds
        Field callingSubscriberIMSI     BER:0x85 BER 2 bcds
        Field callingSubscriberIMEI     BER:0x86 BER 2 bcds
        Field calledPartyNumber         BER:0x87 BER 2 bcds
        Field disconnectingParty        BER:0x88 BER 2 bin
        Field dateForStartofCharge      BER:0x89 BER 2 nstr
        Field timeForStartofCharge      BER:0x8A BER 2 nstr
        Field timeForStopofCharge       BER:0x8B BER 2 nstr
        Field chargeableDuration        BER:0x8C BER 2 nstr
        Field interruptionTime          BER:0x8D BER 2 nstr
        Field timeFromRegSeizuToStofChg BER:0x8E BER 2 nstr
        Field chargedParty              BER:0x8F BER 2 bin
        Field originForCharging         BER:0x90 BER 2 ostr
        Field chargingCase              BER:0x91 BER 2 ostr
        Field tariffClass               BER:0x92 BER 2 ostr
        Field tariffSwitchInd           BER:0x93 BER 2 bin
        Field exchangeIdentity          BER:0x94 BER 2 ascii
        Field mSCIdentification         BER:0x95 BER 2 bcds
        Field outgoingRoute             BER:0x96 BER 2 ascii
        Field incomingRoute             BER:0x97 BER 2 ascii
        Field miscellaneousInformation  BER:0x98 BER 2 ostr
        Field originatingLocationNumber BER:0x99 BER 2 ostr
        Field timeForTCseizureCalling   BER:0x9A BER 2 nstr
        Field cellIDFor1stCellCalling   BER:0x9B BER 2 ostr
        Field cellIDForLastCellCalling  BER:0x9C BER 2 ostr
        Field gSMTeleServiceCode        BER:0x9D BER 2 ostr
        Field gSMBearerServiceCode      BER:0x9E BER 2 ostr
        Field firstRadioChannel         BER:0x9F20 BER 2 bin
        Field callPosition              BER:0x9F21 BER 2 bin
        Field eosInfo                   BER:0x9F22 BER 2 ostr
        Field internalCauseAndLoc       BER:0x9F23 BER 2 ostr
        Field restartDuringCall         BER:0x9F24 BER 2 ostr
        Field restartDuringOutInd       BER:0x9F25 BER 2 ostr
        Field numberOfMeterPulses       BER:0x9F26 BER 2 bin
        Field c7ChargingMessage         BER:0x9F27 BER 2 ostr
        Field c7FirstCHTMessage         BER:0x9F28 BER 2 ostr
        Field c7SecondCHTMessage        BER:0x9F29 BER 2 ostr
        Field applicationIndicator      BER:0x9F2A BER 2 bin
        Field rFPowerClassCalling       BER:0x9F2B BER 2 bin
        Field dTMFUsed                  BER:0x9F2C BER 2 ostr
        Field iCIOrdered                BER:0x9F2D BER 2 ostr
        Field outputForSubscriber       BER:0x9F2E BER 2 bin
        Field iNMarkingOfMS             BER:0x9F2F BER 2 bin
        Field lastPartialOutput         BER:0x9F30 BER 2 ostr
        Field partialOutputRecNum       BER:0x9F31 BER 2 bin
        Field cUGInterlockCode          BER:0x9F32 BER 2 ostr
        Field cUGIndex                  BER:0x9F33 BER 2 ostr
```

```
!if (record_type == sSSubsProc)
{
        variable tClass = Hex2Dec(tariff_Class);
        tariffClass = atol(tClass);
        if(record_type == transit)
       {
                if(orelse{tariffClass < 150}{tariffClass > 180})
                {
                       omitted++;
                       return;
                 }
                if(tariffClass == 180)
                {
                       if(chargeableDuration_sec <= 10)
                       {
                             omitted++;
                             return;
                        }
                }
                () = CVAddOutDataField_Int("TARIFF_CLASS", tariffClass,
ASCII_TYPE);
        }
       else if(orelse{record_type == mSOriginating}{record_type == cF})
       {
                if(orelse{tariffClass == 143}{tariffClass ==
146}{tariffClass ==
               147}{tariffClass == 148})
                {
                       omitted++;
                       return;
                }
                if(andelse{tariffClass == 49}{chargeableDuration_sec <=
10})
                {
                      tariffClass = 48;
                }
                () = CVAddOutDataField_Int("TARIFF_CLASS", tariffClass,
ASCII_TYPE);
       }
       else if(record_type == roamingCF)
       {
                if(andelse{tariffClass == 49}{chargeableDuration_sec <=
10})
                {
                      tariffClass = 48;
                 }
                 () = CVAddOutDataField_Int("TARIFF_CLASS", tariffClass,
ASCII_TYPE);
        }
        else if(orelse{record_type == mSTerm}{record_type ==
mSTermSMSinMSC})
        {
                 () = CVAddOutDataField_String("TARIFF_CLASS","099" ,0,
                 ASCII_TYPE);
        }
        else if(record_type == mSOrigSMSinMSC)
        {
                 () = CVAddOutDataField_String("TARIFF_CLASS","090" ,0,
                 ASCII_TYPE);
        }
        else if(orelse{record_type == composite_mSOrig}{record_type ==
        composite_cF})
```

```
        {
                if(tariffClass == 198)
                {
                        variable tClass =
Hex2Dec(CVGetInDataField_String("tariffClass"));
                        tariffClass = atol(tClass);
                        if (iNMarkingOfMS == 5)
                        {
                            ()CVAddOutData-
Field_String("SWITCH_ID","CA",0,ASCII_TYPE);
                        }
                 }
                () = CVAddOutDataField_Int("TARIFF_CLASS", tariffClass,
ASCII_TYPE);
        }
}
```

```
if(partialOutputRecNum[0])
        {
                PartialRecNumberPresent = 1;
                PartialRecNumber = atol(partialOutputRecNum);

                if(strlen(partialOutputRecNum) > 1)
                {
                        () = CVDiscardCurrentRec(Sprintf("Erronous Partial
Output Record-
                        Number %s", partialOutputRecNum,1));
                        discarded++;
                        return;
                }
        }
        else
        {
                PartialRecNumberPresent = 0;
                PartialRecNumber = 0;
        }
        lastPartialOutput = CVGetInDataField_String("lastPartialOutput");
        if      (CFncErrorFlag == ERROR)
        {
                LastPartialOutPresent = 0;
        }
        else
        {
                LastPartialOutPresent = 1;
        }

        if (andelse {PartialRecNumberPresent == 0} {LastPartialOutPresent
== 0})
        {
                 partial_ind = 0; % not long call
        }
        if (andelse{PartialRecNumberPresent == 1}{PartialRecNumber == 1}
{LastPartialOutPresent == 0})
        {
                partial_ind = 1; % first part of long call
                partial_counter++;
        }
        if (andelse {PartialRecNumberPresent == 1}{PartialRecNumber > 1}
{LastPartialOutPresent == 0})
        {
                partial_ind = 2; % intermediate
                partial_counter++;
        }
        if (andelse{PartialRecNumberPresent == 1}{PartialRecNumber >
1}{LastPartialOutPresent == 1})
        {
                partial_ind = 3; % last part of long call
                partial_counter++;
        }
```

```
R80_VFAL_out.desc" [Read only] 143 lines, 6952 characters
File header
{
	Record HEADER *
	{
		Field CUSTOMER_NO		*  8 ascii	L/' '  ' '
		Field EARLIEST_TCOM_START_DATE  *  6 ascii	R
		Field TRANSFER_CUTOFF_DATE	*  6 ascii	R
		Field PROCESS_DATE		*  6 ascii	R
		Field PROCESS_TIME		*  6 ascii	R
		Field TRANSFER_BATCH_NO		*  4 ascii	R	'0'
		Field TRANSFER_BATCH_SUFFIX	*  2 ascii	R/'0'  '0'
		Field INVOICE_NO		*  8 ascii	L/' '  ' '
		Field CUSTOMER_NAME		*  30 ascii	L/' '  ' '
		Fixed SOURCE_COUNTRY_CODE	*  * ascii	'ALB'
		Fixed SOURCE_NET_PROV_CODE	*  * ascii	'VF'
		Field FILLER1			*  45 ascii	L/' '  ' '
		Field SERVICE_PROVIDER_TYPE	*  1 ascii	L/' '  ' '
		Fixed RECORD_TYPE_A		*  * ascii	'H'
		Field FILE_VERSION_NO		*  4 ascii	R/'0'  '0'
		Field FILLER2			*  8 ascii	L/' '  ' '
		Field POSITION_OF_UNITS		*  2 ascii	R/'0'  '0'
		Field TOTAL_NUMBER_OF_UNITS	*  2 ascii	R/'0'  '0'
		Field FILLER3			*  208 ascii	L/' '  ' '
		Fixed Newline			*  1 bin	0x0a
	}
}

File unified
{
	Record DETAIL_REC *
	{
		Field CHARGED_PARTY		*  18 ascii	L/' '
		Field NON_CHARGED_PARTY		*  18 ascii	L/' '
		Field CATEGORY			*  2 ascii	L/' '  ' '
		Field DISCOUNT_CATEGORY		*  1 ascii	L/' '  ' '
		Field TCOM_START_DATE		*  6 ascii	R
		Field TCOM_START_TIME		*  6 ascii	R
		Field CHARGEABLE_DURATION	*  6 ascii	R/'0'  '0'
		Field FREE_CALL_REASON_CODE	*  1 ascii	R	'0'
		Field TARIFF_CLASS		*  3 ascii	R/'0'  '0'
		Field PRICED_UNITS		*  9 ascii	R/'0'  '0'
		Field WHOLESALE_PRICE		*  9 ascii	R/'0'  '0'
		Field NON_DISCOUNT_VALUE	*  9 ascii	R/'0'  '0'
		Field TRANSFER_BATCH_NO		*  4 ascii	R/'0'  '0'
		Field TRANSFER_BATCH_SUFFIX	*  2 ascii	R/'0'  '0'
		Field CHARGED_PARTY_IND		*  2 ascii	L/' '  ' '
		Field REROUTING_IND		*  1 ascii	L/' '  ' '
		Fixed NETWORK_PROVIDER_CODE	*  * ascii	'VF'
		Field SWITCH_ID			*  2 ascii	L/' '  ' '
		Field TOLL_TICKET_BATCH_NO	*  6 ascii	R/'0'  '0'
		Field RETAIL_PRICE		*  9 ascii	R/'0'  '0'
```

```
Field FORCED_PROCESSING_IND     *  1 ascii      L/' '   ' '
        Field FREE_PERIOD_IND          *  1 ascii      R/'0'   '0'
        Field ORIGIN                   *  1 ascii      L       'P'
        Field TARIFF_TYPE              *  1 ascii      L/' '   ' '
        Field TRANSMISSION_TYPE        *  1 ascii      L/' '   ' '
        Field SERVICE_PROVIDER_TYPE    *  1 ascii      L/' '   ' '
        Field SUBSCRIPTION_TYPE_PREFIX *  1 ascii      L/' '   ' '
        Field FILLER1                  *  4 ascii      L/' '   ' '
        Fixed RECORD_TYPE_A            *  * ascii      'N'
        Field RETAIL_NON_DISC_VALUE    *  9 ascii      R/'0'   '0'
        Field WHOLESALE_BOOK_NUMBER    *  3 ascii      R/'0'   '0'
        Field WHOLESALE_BOOK_TYPE      *  1 ascii      L/' '   ' '
        Field RETAIL_BOOK_NUMBER       *  3 ascii      R/'0'   '0'
        Field RETAIL_BOOK_TYPE         *  1 ascii      L/' '   ' '
        Field CAUSE_FOR_FORWARDING     *  2 ascii      L/'0'   '00'
        Field NUMERIC_FILLER           *  9 ascii      R/'0'   '0'
        Fixed COUNTRY_CODE             *  * ascii      'ALB'
        Field DATA_VOLUME              *  9 ascii      R/'0'   '0'
        Field CHARGING_METHOD          *  1 ascii      L/' '   ' '
        Field SERVICE_TYPE             *  2 ascii      R/'0'   '00'
        Field SERVICE_ID               *  4 ascii      L
        Field ADVICE_OF_CHARGE         *  1 ascii      R/'0'   '0'
        Field TELECOM_DIRECTION        *  2 ascii      L       ' '
        Field DEBIT_CREDIT_IND         *  1 ascii      L/' '   ' '
        Field MOBILE_STN_CLASS_MARK    *  1 ascii      L       '1'
        Field CALL_REFERENCE           *  22 ascii     L/' '   ' '
        Field LOCAL_CALL_START_DATE    *  6 ascii      R
        Field LOCAL_CALL_START_TIME    *  6 ascii      R
        Field TIME_ZONE_ID             *  4 ascii      L/' '   ' '
        Field SEASON_ID                *  1 ascii      L/' '   ' '
        Field IMSI                     *  15 ascii     L/' '   ' '
        Field WSALE_PRICE_PRE_LOYALTY  *  9 ascii      R/'0'   '0'
        Field WSALE_LOYALTY_DISCOUNT   *  9 ascii      R/'0'   '0'
        Field LOYALTY_DISCOUNT_RATE    *  6 ascii      L/' '   ' '
        Field RETAIL_PRICE_PRE_LOYALTY *  9 ascii      R/'0'   '0'
        Field RETAIL_LOYALTY_DISCOUNT  *  9 ascii      R/'0'   '0'
        Field RETAIL_POST_LOY_BY_WSALE *  9 ascii      R/'0'   '0'
        Field FILLER2                  *  31 ascii     L/' '   ' '
        Field WSALE_PRICE_PRE_BUNDLE   *  9 ascii      R/'0'   '0'
        Field RETAIL_PRICE_PRE_BUNDLE  *  9 ascii      R/'0'   '0'
        Field BUNDLE_TYPE              *  2 ascii      L/' '   ' '
        Field CALL_BUNDLE_VOLUME       *  8 ascii      R/'0'   '0'
        Field PERIOD_BUNDLE_VOLUME     *  8 ascii      R/'0'   '0'
        Fixed newLine                  *  1 bin        0x0a
    }


    Record TRAILER *
    {
        Field CUSTOMER_NO              *  8 ascii      L/' '   ' '
        Field EARLIEST_TCOM_START_DATE *  6 ascii      R
        Field LATEST_TCOM_START_DATE   *  6 ascii      R
```

```
        Field PROCESS_DATE              *  6 ascii      R
        Field TOTAL_DURATION            *  10 ascii     R/'0'   '0'
        Field TOTAL_PRICED_UNITS        *  10 ascii     R/'0'   '0'
        Field TOTAL_WSALE_PRICE         *  12 ascii     R/'0'   '0'
        Field TOTAL_TELE_SERVICES       *  8 ascii      R/'0'   '0'
        Field PROCESS_TIME              *  6 ascii      L
        Field TOTAL_NON_DISCOUNTABLE    *  12 ascii     R/'0'   '0'
        Field TRANSFER_BATCH_NO         *  4 ascii      R/'0'   '0'
        Field TRANFER_BATH_SUFFIX       *  2 ascii      R/'0'   '0'
        Field INVOICE_NO                *  8 ascii      L/' '   ' '
        Field TOTAL_RETAIL_PRICE        *  12 ascii     R/'0'   '0'
        Field FILLER1                   *  17 ascii     L/' '   ' '
        Fixed RECORD_TYPE_A             *  * ascii      'T'
        Field TOTAL_RETAIL_NON_DISC_VAL *  12 ascii     R/'0'   '0'
        Field POSITION_OF_UNIT          *  2 ascii      R/'0'   '0'
        Field TOTAL_NUMBER_OF_UNITS     *  2 ascii      R/'0'   '0'
        Field RECORDS_IN_UNIT           *  8 ascii      R/'0'   '0'
        Field TOTAL_SUPP_SERVICES       *  8 ascii      R/'0'   '0'
        Field TOTAL_DATA_VOLUME         *  12 ascii     R/'0'   '0'
        Field TOTAL_WSALE_PRICE_DC      *  1 ascii      L/' '   ' '
        Field TOTAL_NON_DISCOUNTABLE_DC *  1 ascii      L/' '   ' '
        Field TOTAL_RETAIL_PRICE_DC     *  1 ascii      L/' '   ' '
        Field TOTAL_RETAIL_NON_DISC_DC  *  1 ascii      L/' '   ' '
        Field TOTAL_DETAIL_RECORDS      *  8 ascii      R/'0'   '0'
        Field TOTAL_SERVICE_CENTRES     *  8 ascii      R/'0'   '0'
        Field TOTAL_WSALE_LOY_DISC      *  12 ascii     R/'0'   '0'
        Field TOTAL_RETAIL_LOY_DISC     *  12 ascii     R/'0'   '0'
        Field TOTAL_WSALE_PRE_BUNDLE    *  12 ascii     R/'0'   '0'
        Field TOTAL_RETAIL_PRE_BUNDLE   *  12 ascii     R/'0'   '0'
        Field FILLER2                   *  112 ascii    L/' '   ' '
        Fixed newLine                   *  1 bin        0x0a
    }


}
```