NOKIA

Open MSS CM CFP5, Product Documentation, issue 01

Storing and Transfer of Charging Data

DN99611861
Issue 12-0-0

The information in this document applies solely to the hardware/software product (“Product”) specified herein, and only as specified herein. Reference to “Nokia” later in this document shall mean the respective company within Nokia Group of Companies with whom you have entered into the Agreement (as defined below).

This document is intended for use by Nokia's customers (“You”) only, and it may not be used except for the purposes defined in the agreement between You and Nokia (“Agreement”) under which this document is distributed. No part of this document may be used, copied, reproduced, modified or transmitted in any form or means without the prior written permission of Nokia. If You have not entered into an Agreement applicable to the Product, or if that Agreement has expired or has been terminated, You may not use this document in any manner and You are obliged to return it to Nokia and destroy or delete any copies thereof.

The document has been prepared to be used by professional and properly trained personnel, and You assume full responsibility when using it. Nokia welcomes your comments as part of the process of continuous development and improvement of the documentation.

This document and its contents are provided as a convenience to You. Any information or statements concerning the suitability, capacity, fitness for purpose or performance of the Product are given solely on an “as is” and “as available” basis in this document, and Nokia reserves the right to change any such information and statements without notice. Nokia has made all reasonable efforts to ensure that the content of this document is adequate and free of material errors and omissions, and Nokia will correct errors that You identify in this document. Nokia's total liability for any errors in the document is strictly limited to the correction of such error(s). Nokia does not warrant that the use of the software in the Product will be uninterrupted or error-free.

NO WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF AVAILABILITY, ACCURACY, RELIABILITY, TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, IS MADE IN RELATION TO THE CONTENT OF THIS DOCUMENT. IN NO EVENT WILL NOKIA BE LIABLE FOR ANY DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, DIRECT, INDIRECT, INCIDENTAL OR CONSEQUENTIAL OR ANY LOSSES, SUCH AS BUT NOT LIMITED TO LOSS OF PROFIT, REVENUE, BUSINESS INTERRUPTION, BUSINESS OPPORTUNITY OR DATA THAT MAY ARISE FROM THE USE OF THIS DOCUMENT OR THE INFORMATION IN IT, EVEN IN THE CASE OF ERRORS IN OR OMISSIONS FROM THIS DOCUMENT OR ITS CONTENT.

This document is Nokia proprietary and confidential information, which may not be distributed or disclosed to any third parties without the prior written consent of Nokia.

Nokia is a registered trademark of Nokia Corporation. Other product names mentioned in this document may be trademarks of their respective owners.

Important Notice on Product Safety

This product may present safety risks due to laser, electricity, heat, and other sources of danger.

Only trained and qualified personnel may install, operate, maintain or otherwise handle this product and only after having carefully read the safety information applicable to this product.

The safety information is provided in the Safety Information section in the “Legal, Safety and Environmental Information” part of this document or documentation set.

Nokia is continually striving to reduce the adverse environmental effects of its products and services. We would like to encourage you as our customers and users to join us in working towards a cleaner, safer environment. Please recycle product packaging and follow the recommendations for power use and proper disposal of our products and their components.

If you should have questions regarding our Environmental Policy or any of the environmental services we offer, please contact us at Nokia for any additional information.

Table of Contents

This document has 65 pages

List of Figures

List of Tables

Changes in the document

Changes between document issues are cumulative. Therefore, the latest document issue contains all changes made to previous issues.

Changes made between issues 12-0-0 and 11-0-0

Section *Compression of charging data files* was updated.

Changes made between issues 11-0-0 and 10-0-0

Section *Compression of charging data files* was updated.

Changes made between issues 10-0-0 and 9-1-4

Corrected the maximum number of CDRs and maxiumum file size in the following sections:

- Charging files on hard disks
- Recommended file size and amount
- Configuration of the VDS system

Changes made between issues 9-1-4 and 9-1-3

The section *Writing of TTTCOF to disk fails if TTTCOF file is locked on one disk* was added.

Changes made between issues 9-1-3 and 9-1-2

Editorial changes.

Changes made between issues 9-1-2 and 9-1-1

Editorial changes.

Changes made between issues 9-1-1 and 9-1

Editorial updates have been made throughout the document to reflect changes incurred by the Nokia portfolio.

Section 1.2.6 Diameter interface has been removed.

Changes made between issues 9-1 and 9-0

Feature names have been corrected according to the approved feature names.

1 Changes in Storing and Transfer of Charging Data

Charging records

Information on charging records in different call cases is given in *Generation and Contents of CDRs in Different Call Cases*, Interface Specification.

The coding of different CDR fields and the general description and format of CDR fields are described in CDR *Field Description*, Interface Specification.

Modifications to charging interface MML

Modifications to the charging interface MML are described in the following documents:

- *MML Changes in M14.0,* Interface Specification
- *Charging Handling,* Operating Instructions
- *Virtual Data Storing Device Handling,* IF Command Group
- *Detailed Charging Handling,* GT Command Group
- *Dynamic Data Handling,* GA Command Group
- *CDR and Report Storing Handling,* GE Command Group
- *Hot Billing Handling,* GB Command Group

2 Interface description

2.1 Introduction to Storing and Transfer of Charging Data

Storing and Transfer of Charging Data, Interface Specification contains information on transferring charging data from the MSS to the post-processing system. It also provides background information to understand the detailed description of the interface.

The following concepts are explained in this document:

- storing charging data on hard disks and on removable media

NOTICE: Logical file connections related to storing the charging data are only briefly discussed. See *I/O System Administration, Operating Manual* for more detailed information on logical files and the storing device handling.

- the format of charging files and control files
- transferring charging data from the MSS to the post-processing system, for example, to the Billing Center

NOTICE: Instructions on configuring the data transfer connection between MSS and the post-processing system is not in the scope of the document. See *OSI Guide* and *TCP/IP Functional Description* for the instructions.

- the immediate Charging Data Record (CDR) transfer interfaces offered by features *Feature 161: Hot Billing* and *Feature 1491: Immediate CDR Transfer for All Subscribers*

See *Table: Charging documentation* for finding information on another areas of charging.

Table 1 Charging documentation

Document title	Covered charging areas
Charging, Functional Description	general description of charging
Charging Handling, Operating Manual	tasks related to detailed charging and charging counters
Generation and Contents of CDRs in Different Call Cases, Interface Specification	call case examples with the corresponding CDR format, information on data fields and timestamps
CDR Field Description, Interface Specification	detailed description of the fields in CDRs

2.2 Interface description

The MSS contains one or more redundant Charging Unit (CHU) pairs that collect charging data, and maintain various counters and produce CDRs.

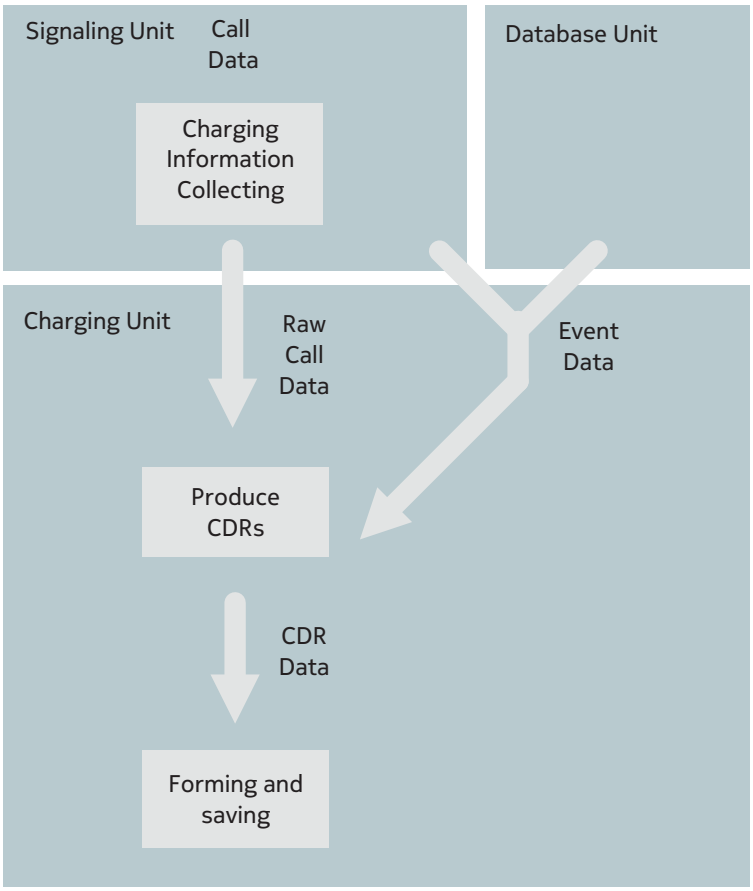
2.2.1 General

During the call or event, charging information is collected. The collected data is sent for forming. After a CDR is formatted, it is either transferred immediately using Hot Billing or stored on a local mass memory device and transferred later on with FTP, or it is transferred immediately to the post-processing system with GTP' using the Immediate CDR transfer for all subscribers feature.

CDRs are generated in the MSS. Within a network element, a CDR is generated in the CHU. A dynamic data formatter formats call data or event data using a customer-specific format description.

The functionality of charging data handling before storing or transfer is shown in *Figure: Functionality of charging data handling before storing or transfer*.

Figure 1 Functionality of charging data handling before storing or transfer



Charging Information Collection Function collects the charging information during the call. It transmits the collected data of the chargeable call to the CDR Data Collection Function. CDRs related to other chargeable events are mostly formed immediately after the event in question.

CDR Data Collection Function collects the required charging data from the raw data or event data, and sends it to the CDR Forming and Saving Function. It forms the CDRs according to the customer-specific definition of chargeable call types and events and stores them in the RAM of the CHU.

After the CDR is formatted, the functionality depends on the selected transfer method. There are two different methods for transferring the CDRs to the post-processing system:

- A traditional CDR transfer: the CDRs of Hot Billing subscribers are transferred immediately to the post-processing system. Other (non-Hot Billing) CDRs are collected into a charging block, which is stored on a local mass memory device. The CDR files on the hard disk can be transferred to the post-processing system using FTP. Alternatively, the CDR files can also be copied to a removable media, which can be manually transferred to the post-processing system.
- Immediate CDR transfer for all subscribers: You can use this function if *Feature 1491: Immediate CDR Transfer for All Subscribers* is activated. Several formatted CDRs are collected into a small charging block (GTP' package), which is transferred immediately to the post-processing system using the GTP' protocol. In case of unsuccessful immediate transfer to any configured destination, or if the transfer is not fast enough, the Fall Back Saving functionality stores the CDRs on a local mass memory device. The CDRs stored on the hard disk can be transferred using the FTP. Alternatively, the CDR files can also be copied to removable media. Removable media can be manually transferred to the post-processing system.

Note: Storing on hard disks and storing on removable media follows the same principles in both methods. However, the content of the charging files, the path, and the file names are different in the two modes.

Note: The GTP' interface is also used in both methods. In an immediate CDR transfer for all subscribers, it is the primary transfer interface and in a traditional CDR transfer it is the preferred transfer protocol of Hot Billing.

2.2.1.1 Logical file connections

The logical file is a method defined in the I/O system for hiding the actual I/O device from an application. The reading and writing tasks that are directed to a logical file are routed to a physical I/O device according to the control data contained in the file.

The MSS I/O system is responsible for special facilities needed for each device and essentially it handles the working states of the equipment. Thus, an application can direct its I/O task to a logical file and the I/O system transfers it further to all the devices that have been defined in the logical file concerned. In a fault situation, the I/O system directs the output to a possible spare device. A maximum of four objects can be connected to one logical file. The objects may be I/O devices or logical files.

The logical files are TTLOFI, CFCLOF, and HBLOFI. Each CHU contains these logical files. The TTLOFI is used to store charging data to the selected devices (hard disks and cartridge tapes). The CFCLOF is used when charging data is copied from the hard disk to the tape with the `IFC` command. The HBLOFI is used to store the HB data to selected devices (Hot Billing is an optional feature).

The logical files TRACIG, TRACSU, TRACIR, TRAZON, and TRATOT are used to store accounting data to selected devices. The default connection for all logical files is a VDS.

For more detailed information, see *I/O System Administration* and *Charging Handling*.

2.2.1.2 Several CHU pairs

Feature 641: Charging Capacity provides you the option of having more than one pair of CHUs in MSS. The functionality of charging is not affected by the number of CHU pairs. Adding more CHU pairs only improves the capacity. For example, when a second CHU pair is added, the storing capacity is doubled.

The charging data is stored to charging disk files within each CHU. Each CHU contains unique charging data, and the control files for the data transfer. A connection to the post-processing system has to be created for each CHU pair. More information on the CHU connections to a post-processing system can be found in *OSI Guide* and *TCP/IP Guide*.

The accounting counters are handled by a CHU-0–pair (other CHU-pairs send update messages to the CHU-0–pair when counters are increased).

One call is handled in the same CHU unit, therefore, all CDRs related to the call generated in this particular MSS are stored on the same charging disk. There are two exceptions which are related to the following call scenario. For example: A calls B and a call is established. C calls A, A puts B on hold and answers to C. The call between A and B is handled, for example, by the CHU-0–pair and the call between C and A is handled, for example, by the CHU-1–pair. In this case, A has the following options:

- A can activate a multiparty call. The original calls are handled by original CHU-pairs, so CDRs related to these original calls are made in the different CHU-pairs. The HW CDR related to multiparty can be created in any CHU-pair (for example, in CHU-2–pair).
- A can make a Call Transfer. The original calls are handled by original CHU-pairs, so CDRs related to these original calls are made in different CHU-pairs. The Supplementary Service (SUPS) CDR and the Forwarded Call (FORW) CDR related to Call Transfer can be made in one of the CHUs handling the calls (in the above example it is CHU-1–pair).

2.2.1.3 The structure of Data Communication Network

The Data Communication Network (DCN) alternatives, including hardware requirements, are described in *OSI Guide* and *TCP/IP Guide*.

2.2.2 Traditional File Based CDR Transfer

After being generated, CDRs are saved and stored on a local storing device, usually on the hard disks of the CHU, or on removable media. They can be transferred later on to the BC using FTP.

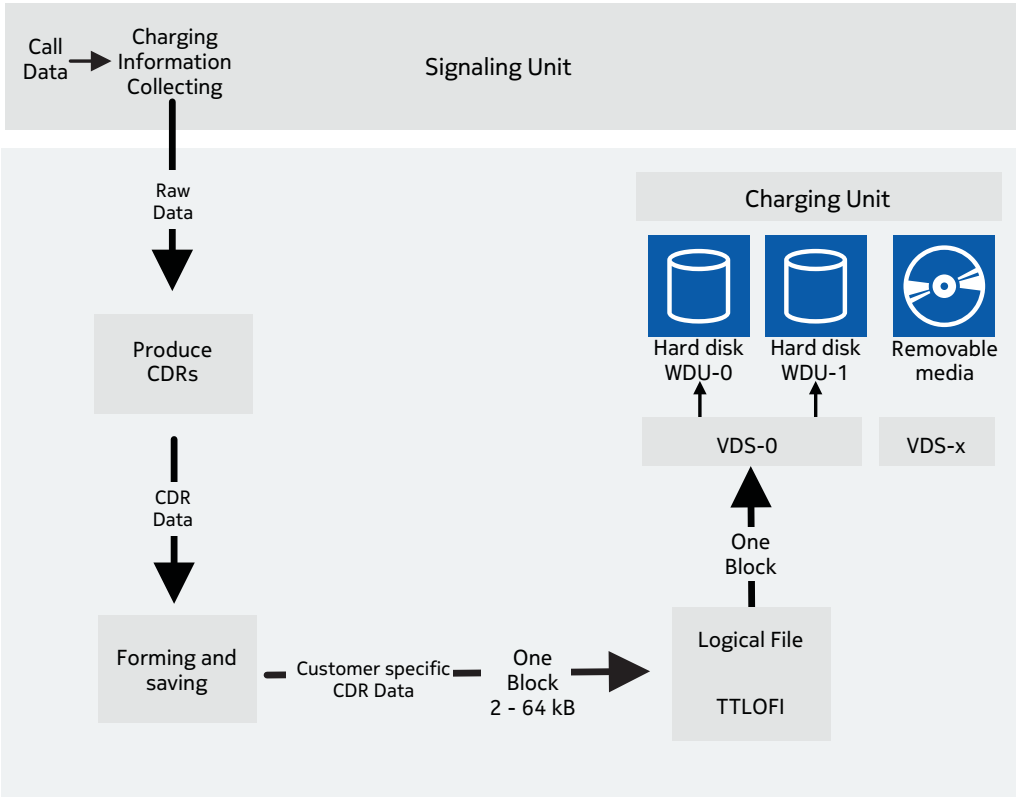
When the charging block fills up, its content is stored in a local mass memory device defined in the logical files by the operator. All charging blocks for which the storing task is not yet acknowledged are buffered in a big RAM buffer which can hold approximately 5-10 minutes worth of charging data when the switch is running within the design capacity load limits.

The RAM is divided into several blocks. One block is the amount of charging data written on the hard disks at a time. The block is transmitted further to the TTLOFI logical file when the block becomes full or the warm billing timer goes off. TTLOFI contains the information on the physical I/O devices where writing tasks are routed to. Usually, the two hard disks of the CHU are used. In addition to hard disks, the CDRs can be stored on removable media as well. On hard disks, several charging blocks are written to one charging file. The charging files can be transferred further from the hard disk by using FTP.

If optional *Feature 1821: Charging I/O Improvement* is in use, the CDR Forming and Saving Function collects charging blocks to one 256KB buffer before writing it to the hard disk. If this 256KB buffer does not become full with pre-defined time interval (GEV MML command, GE Command Group), the buffer is saved to the disk even if it is not full.

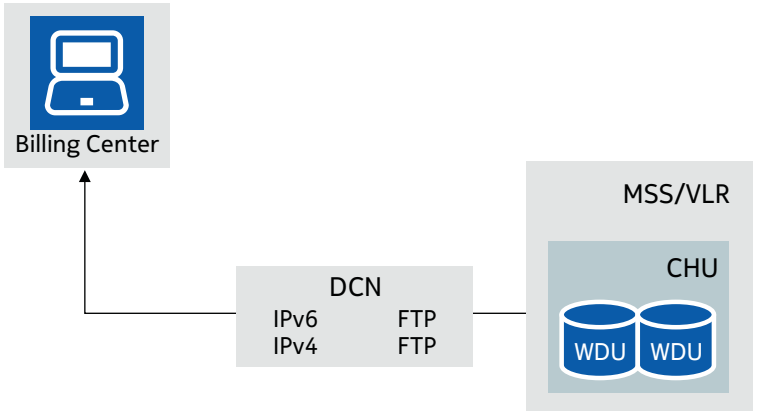
Note: The use of spare devices is highly recommended for storing charging data. You can make the connection by using one of the storing devices as a primary device and the other one as a secondary device. The secondary unit is used if the storing in the primary device does not succeed. For more information, see *I/O System Administration*.

Figure 2 Functionality of charging data saving



The transfer of charging data files from hard disks of the MSS network element to a post-processing system can be done using FTP. The general idea of transferring the charging data is shown on *Figure: Transfer of charging data from MSS to BC*.

Figure 3 Transfer of charging data from MSS to BC



DCN 2 is the only configuration.

Data between a network element and a post-processing system is transmitted through a Local Area Network (LAN) connection. When using FTP, only LAN connection can be used as the data transmission connection.

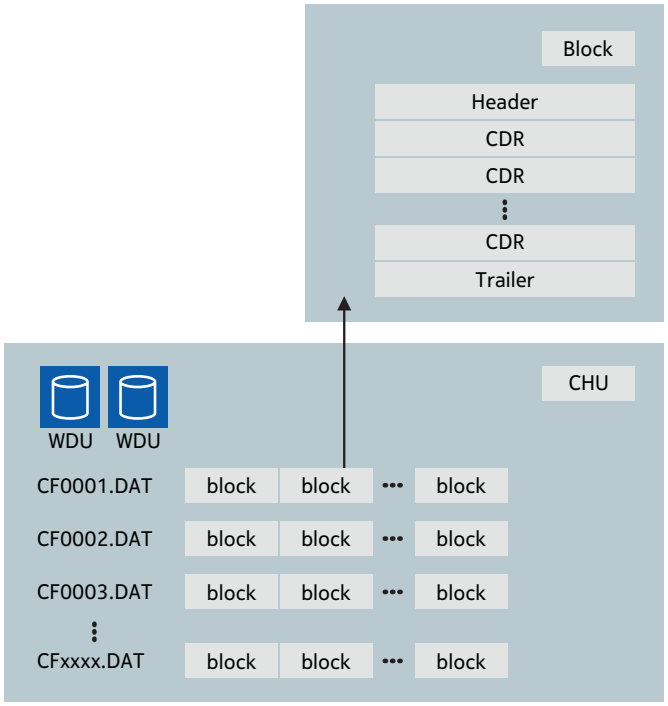
2.2.2.1 Charging files on hard disks

The CDRs are written on the disk files located in the root directory of the hard disks of the CHU. The files are named CFxxxx.DAT or CFxxxx.Z in case of compression, where the *xxxx* is an ordinal number. The files act as ring buffers, and the ordinal number can range from 0001 to 9999.

These files are filled with charging blocks containing the CDRs. Each block has a header, and a trailer record. The header is located at the beginning of the block, and the trailer at the end of the block.

The structure of the CDRs on the disk files is shown in .

Figure 4 The structure of charging disk files



The CDRs are numbered sequentially, except for the headers and trailers. The block size remains constant during operation, but it can be changed (requires system restart).

A header is generated at the beginning of each charging block. The header includes, for example, the block and the batch sequence number, which is the sequential number of the charging data batch (with a value between 1 and 65535). All blocks in one disk file have the same batch sequence number. The header also includes the sequence number of the first CDR in the block, and the length of the actual data in this block, and the timestamp. A description of a header record is shown in Table 2: Field description of a header record.

Table 2 Field description of a header record

Field Description	HEX Format	Field Content
Record length	29 00	41 bytes
Record type	00	0 = header
Block size	01	size of charge, block 01 = 8176 bytes
Tape block type	01 00	charging block
Data length in block	E2 1F	8162 bytes

Table 2 Field description of a header record (Cont.)

Field Description	HEX Format	Field Content
Exchange id	94 71 37 89 FF FF FF FF FF FF	49177398
First record number	01 00 00 00	1
Batch sequence number	85 05 03 00	30585
Block sequence number	01 00	1
Start time	53 03 23 09 08 00 20	2000.08.09, 23:03:53
Format version	4D 30 02 01 00 FF	Y00CUSMO ver 2.1–0

A trailer is generated at the end of each charging block. The trailer includes, for example, the timestamp and the sequence number of the last CDR in the block. The number of charging records in a block can vary because of the different length of records. A description of a trailer record is shown in Table 3: Field description of a trailer record.

Table 3 Field description of a trailer record

Field Description	HEX Format	Field Content
Record type	10	10 = trailer
Exchange id	94 71 37 89 FF FF FF FF FF FF	49177398
End time	13 04 23 09 08 99 19	2000.08.09, 23:04:13
Last record number	33 00 00 00	33

An example of a header and a trailer record in a charging data file is shown in Figure 5: A header and a trailer on a disk file. An example of charging data on a disk file is shown in An example of charging data on a disk file (8176B charging block). For detailed descriptions of the header, the trailer, and the other record types, see *CDR Field Description*, interface specification.

Figure 5 A header and a trailer on a disk file

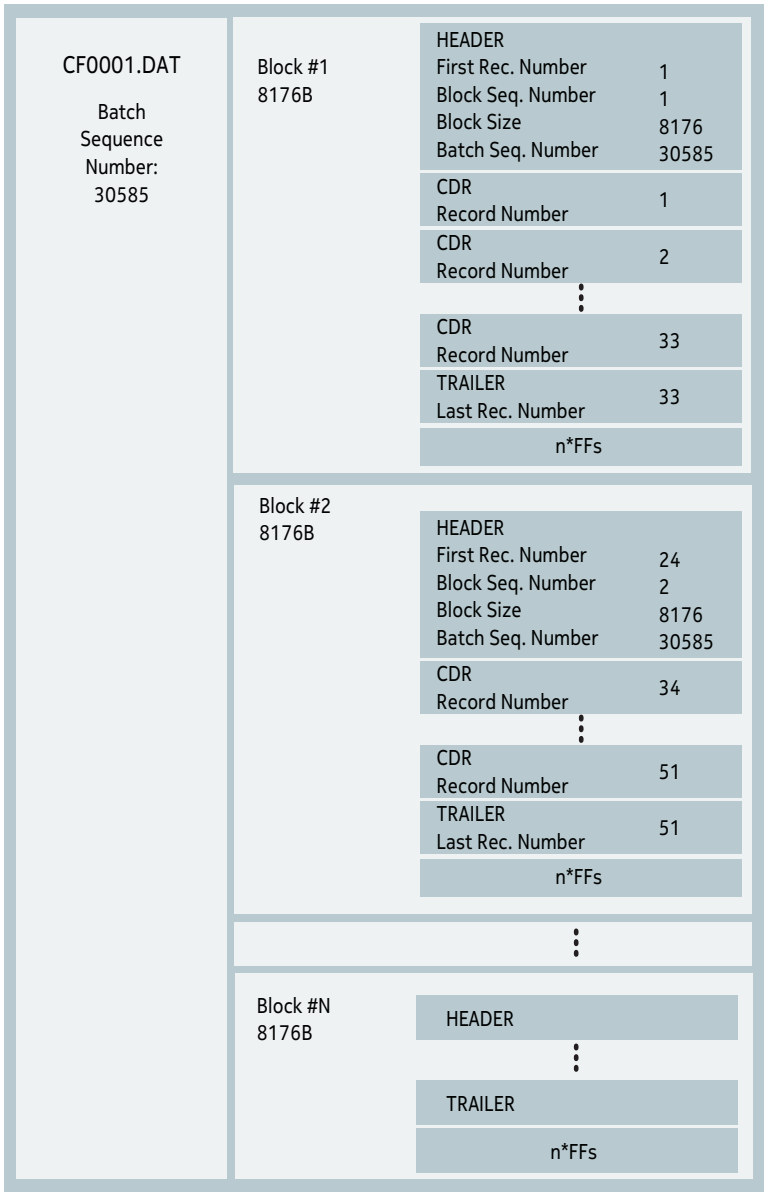
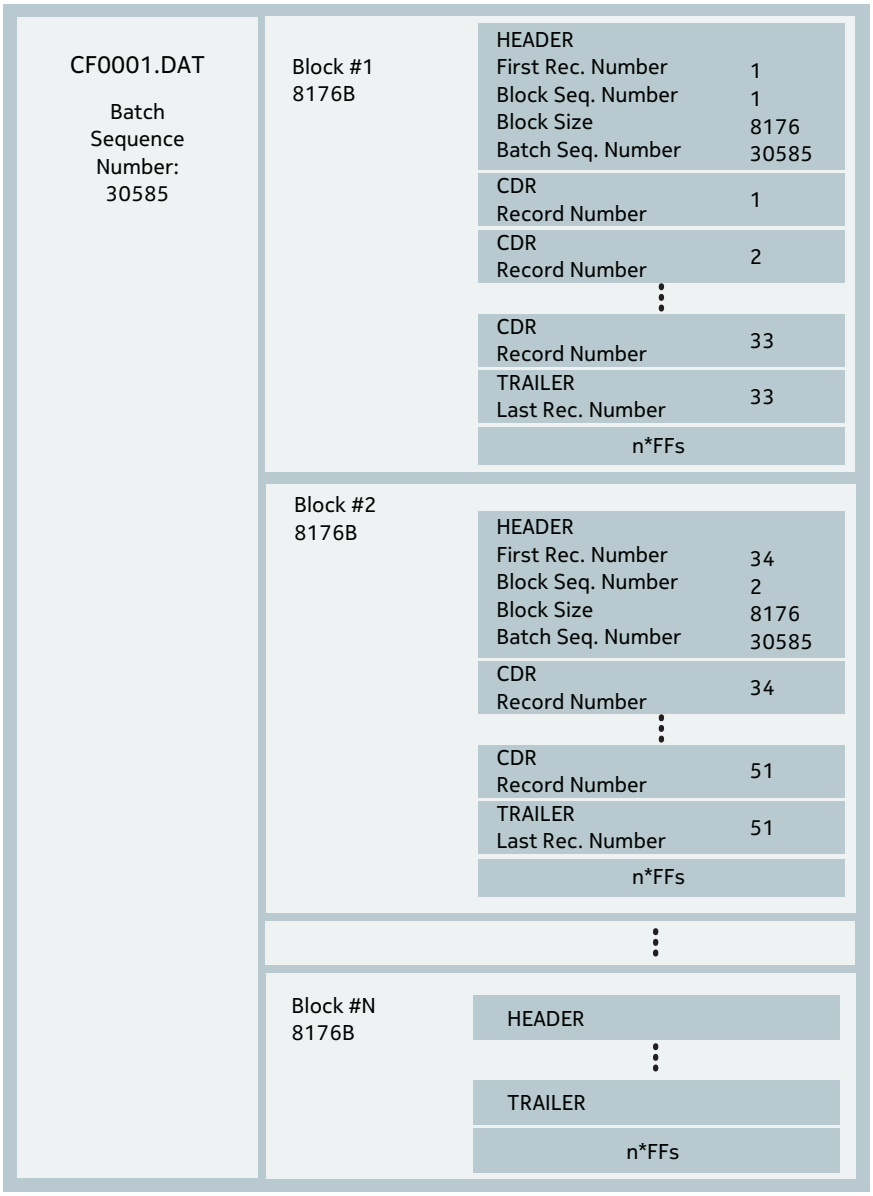


Figure 6 A header and a trailer on a disk file



An example of charging data on a disk file (8176B charging block)

```
       1  2  3  4  5  6  7  8   9 10 11 12 13 14 15 16

00000  29 00 00 01 01 00 E2 1F  94 71 37 89 FF FF FF FF  HEADER
00010  FF FF 01 00 00 00 85 05  03 00 01 00 53 03 23 05
00020  06 97 19 4D 30 04 01 00  FF 40 00 07 01 00 00 00  LOCA
00030  00 0C 24 FF FF FF FF FF  94 71 37 89 FF FF FF FF
00040  FF FF 62 02 53 18 11 51  00 F5 88 88 FF FF FF FF
00050  FF FF FF FF FF FF 88 88  94 71 37 89 FF FF FF FF
00060  FF FF 54 40 08 05 06 97  19 DF 00 01 02 00 00 00  MOC
00070  00 7F 5F 33 41 27 01 01  94 71 37 89 FF FF FF FF
00080  FF FF 00 00 00 62 02 53  18 11 51 00 F0 10 20 30
00090  40 50 60 70 F0 94 71 57  58 00 F0 FF FF FF FF 00
000A0  00 62 02 53 18 11 51 00  F1 10 20 30 40 50 60 70
000B0  F0 06 71 57 58 00 F1 FF  FF FF FF FF FF FF 00 04
000C0  10 77 85 05 10 FF FF FF  FF FF FF FF 88 88 82 03
```



```
000D0   94 71 37 89 FF FF FF FF   FF FF 88 88 82 03 88 88
000E0   82 03 88 88 82 03 01 00   02 00 00 11 10 00 FF 18
000F0   46 41 08 05 06 97 19 53   41 08 05 06 97 19 08 42
00100   08 05 06 97 19 00 00 00   00 03 00 FF FF FF 00 00
00110   15 00 00 00 02 00 FF FF   06 71 97 02 00 F1 FF FF
00120   FF FF FF FF FF FF FF FF   FF 05 06 71 57 58 00 F1
00130   FF FF FF FF FF FF FF FF   FF FF FF FF FF FF FF FF
00140   00 FF FF 33 41 27 01 01   9C 00 02 03 00 00 00 00  MTC
00150   51 3B 33 41 27 01 01 94   71 37 89 FF FF FF FF FF
    *
    *
    *
01FB0   19 12 00 00 00 11 01 01   FF FF FF FF 10 77 85 05
01FC0   10 FF FF FF FF FF FF FF   00 04 18 00 10 94 71 37  TRAILER
01FD0   89 FF FF FF FF FF FF 28   02 09 06 06 97 19 36 00
01FE0   00 00 FF FF FF FF FF FF   FF FF FF FF FF FF FF FF  FILLED WITH FF
01FF0   29 00 00 01 01 00 61 10   94 71 37 89 FF FF FF FF  HEADER
02000   FF FF 01 00 00 00 85 05   03 00 02 00 32 02 09 06
```

2.2.2.2 Warm billing

Feature 531: Warm Billing makes it possible to define a time interval after which the RAM buffer containing the charging data of ended calls is stored in the storing device that is used. If you define a regular interval at which the charging data is stored in a storing device, the charging block is stored at that interval even if it is not full. This can be, for example, relevant at night-time when there is not much call traffic in the exchange and the charging block is open for a longer period of time.

It is also possible to define a time interval for closing the charging data file. Using these two parameters together makes it possible to send charging data to the post-processing system considerably faster than normally at low traffic periods.

For more information, see *CDR and Report Saving Handling*, GE Command Group and *Virtual Data Storing Device Handling*, IF command group.

In case of Warm Billing, the number of subscribers is not restricted, and the number of charging records is not limited (see figures in section *Traditional File Based CDR Transfer*).

Note: Note, however, that for performance reasons the charging file cannot be closed because of this timer at busy hours, and the size of charging file must be reasonable. See also section *VDS storing capacity*.

2.2.2.3 Storing capacity

The size of the RAM block, the size of hard disks, and the data format affect the storing capacity.

CDRs are stored in charging blocks from the RAM to the storing device. Each block contains several CDRs. A larger RAM block results in a more effective functioning of the system because the number of storing events decreases. However, a larger RAM block leads to a longer time period before saving the charging data from the RAM on hard disks.

The maximum charging block size is 64 kB.

The size of the RAM block is N (integer) times 8176 bytes. N=1 (8176 Bytes) is the default value, though in new equipments it is recommended to use N=8 (65408 Bytes) due to performance reasons. (For example, capacity statements requires a 64kB charging block size). Other possible values are N=2 (16352 bytes) and N=4 (32704 bytes). For backwards compatibility, N=0 (2044 Bytes) is also supported. The RAM block size can be changed by using the relevant PRFILE parameter. The post-processing system must be able to handle all block sizes that may be received from the MSS.

The record types generated from different call cases and the content of each record type chosen by the operator affect the amount of data stored. Only the necessary fields have to be selected.

For more information on the MML commands used, see *GE - CDR and Report Saving Handling, Command Reference Manual*.

The capacity reference figures can be found in *Design Capacity* documents. These documents assume that a traditional file-based CDR transfer is used without Hot Billing, 64kB charging block is used, CDRs are stored only (and only once) on hard disks and all available CHU-pairs are in use.

The design capacity documents state the capacity figures only for maximum configuration. There are no official design capacity figures available for all configurations, for example, an MSS with only one CHU-pair. Rough estimates, however, can be based on the distribution figures.

The following is an example on how these reference figures are calculated.

1. Calls 8 000 000 BHCA, 3 CDRs/call = 24 000 000 CDRs/BH
2. Non-call-related services (for example, SMS) 8 000 000 SMS/BH, 1 CDR/SMS = 8 000 000 CDRs/BH
3. Total CDRs/BH is 24 000 000 CDRs/BH + 8 000 000 CDRs/BH = 32 000 000 CDRs/BH.
4. In a hardware configuration with 3 CHU pairs each CHU pair handles 24 000 000 / 3 = 8 000 000 call related CDRs/BH, and 8 000 000 / 3 = 2 666 667 non-call related CDRs/BH, altogether 10 666 667 CDRs/BH, which is 2222 + 741 = 2963 CDRs/s per CHU pair.
5. Assuming average of 350 Bytes/CDR for call related CDRs, and 220 Bytes/CDR for non-call related CDRs the amount of charging data in each CHU is 2222 * 350 Bytes + 741 * 220 Bytes, about 919 kB/s

2.2.2.4 Recommended file size and amount

The capacity of the buffer should be configured so that it can store at least three days worth of charging data. Almost all the storing capacity of the hard disks can be used for storing charging data. No other data should be stored to the hard disks of CHU. If *Feature 1491: Immediate CDR Transfer for All Subscribers* has been used and deactivated, the files created by Fall Back Saving of Immediate CDR Transfer must be removed before the full capacity of the hard disk can be used. Very small file sizes cause capacity problems, because of the overhead of handling the control files. On the other

hand, if the file size is set to a high value, files may be open for a long time during a low-traffic period delaying the transfer to the post-processing system, or the files may not be efficiently used if the files are forced to be closed periodically. The maximum number of charging files can be set up to 9999, and the maximum file size can be configured up to 32 MBs. These two parameters shall be set taking into consideration the size of the hard disk, the traffic profile, and the frequency of file transfer to the post-processing system.

When using a 300GB hard disk without file compression, the file size can be set to 32 MBs and the amount of files to 9300 in order to use the maximum capacity of the hard disk for charging data.

Note: It is recommended to configure the switch so that you use the maximum capacity of the hard drive for saving charging data, and minimize the frequency of sending the charging data to the post-processing system.

2.2.3 Hot Billing

Hot Billing (HB) enables the MSS to send the CDRs to the post-processing system automatically, and immediately after they are produced. It is possible either by transferring online using GTP', or by saving the HB CDRs with normal CDRs or in separate files and transferring them with FTP. Some CDRs cannot be identified as HB CDRs

Hot Billing is used along with traditional file based CDR transfer (see section *Traditional File Based CFR Transfer*) to offer an immediate CDR transfer for a small proportion of subscribers. A CDR is handled as a hot record if the charged subscriber has the hot billing supplementary service provisioned. The HB subscribers have a Nokia-proprietary Hot Billing supplementary service provisioned in the HLR. In order to use HB, both the HLR and MSS must be Nokia products. After a CDR is formatted, the CDRs of HB subscribers can be sent immediately to the post-processing system.

The CDR Forming and Saving Function sends the CDRs of the HB subscribers to the Hot Billing Saving Function. It is possible to define whether a copy of these HB CDRs is included within normal charging data.

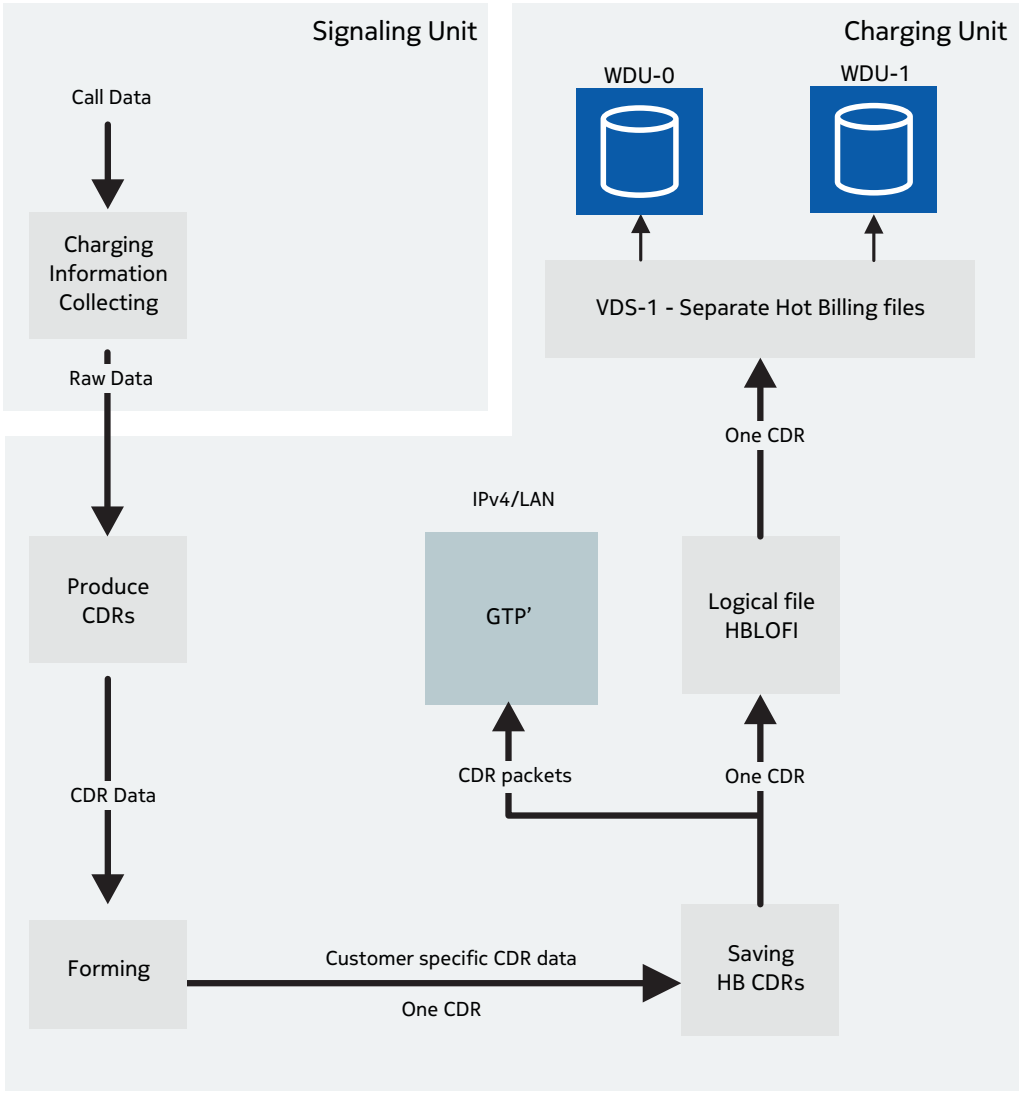
The HB CDRs can be stored and transferred in the following ways:

- Online transfer using the GTP' over (TCP/IP or) UDP/IP, with confirmation from the post-processing system. (All new implementations should use this CDR transfer method).
- Saving the hot CDRs together with normal CDRs and transferring them with FTP. (This is the only option which can be used simultaneously with the other options.)
- Saving the HB CDRs in separate files and transferring them with FTP via HBLOFI. (For testing purposes only.)

One or more of these options can be activated with parameters. It is recommended to apply the online transfer using the GTP' as the primary method. The hot CDRs are also expected to be stored in the normal charging files as backups in case of a possible loss of CDRs during the transfer as well as in case of link failure situations.

Storing the HB CDRs in separate files should only be used for testing purposes. The reason for this recommendation is that since the CDRs are stored in these files one by one, it slows down the storing of normal charging data.

Figure 7 Collecting and storing of Hot Billing charging records in CHU

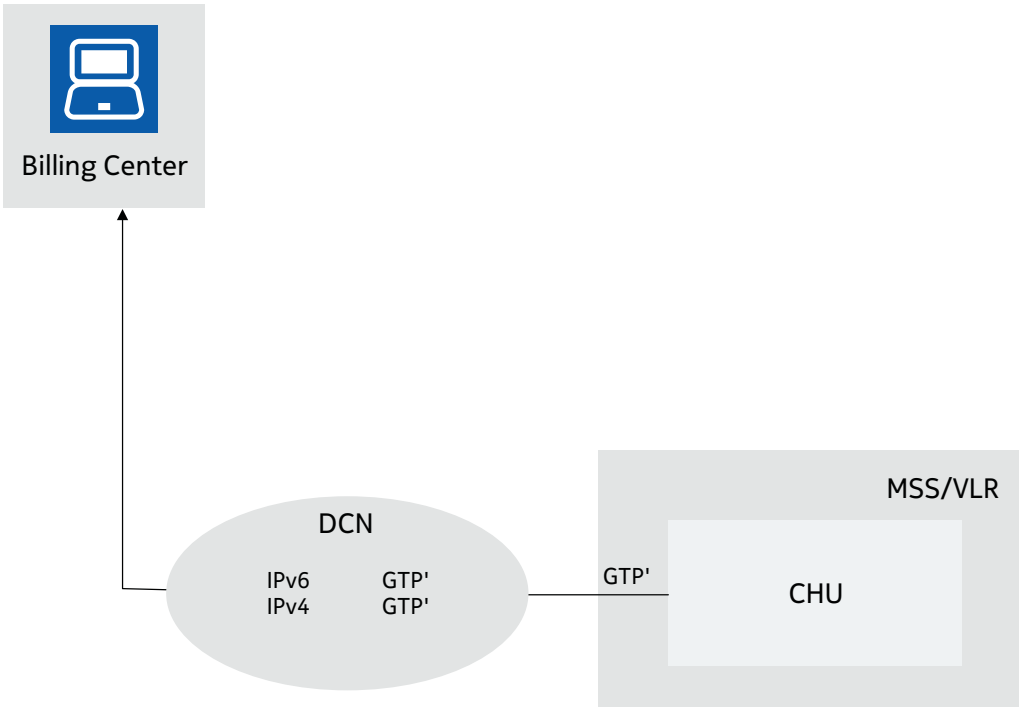


The Hot Billing saving Function stores the HB CDRs to the RAM of the CHU. A CDR is transmitted immediately either to the I/O system which directs it to the destination defined by HBLOFI or to the GTP' (via UDP/IP or TCP/IP socket interface). HBLOFI is a logical file that contains the information on the I/O devices to which the writing tasks are routed to. The operator can choose whether the HB CDRs are handled in separate HB files (used only for testing purposes), or the CDRs are only sent to a post-processing system.

Note: HBLOFI should not be connected so that the HB CDRs and normal charging blocks (connection of logical file TTLOFI) are stored on the same device.

The Hot Billing Saving Function can buffer the HB CDRs if the LAN connection does not work or is overloaded.

Figure 8 Transfer of Hot Billing data from MSS to BC



The transfer of the HB records is carried out by using the existing protocol services and connections. GTP' over LAN connections are used as the protocol interface. Descriptions of these three methods are presented later in the subsections.

Hot Billing Record Number

In those CDRs that can be handled as hot records, a field HOT_BILLING_RECORD_NUMBER is available. Operator can choose whether this field is included in the CDR format or not. This field has the sequential number of the hot CDR. It is also included in the normal CDRs where it has the value FFFFFFFFH. It can be used by the post-processing system to detect any possibly missing hot CDRs. It can also be used to distinguish the hot CDRs from the normal CDRs in the charging data files. The format of the field is a four-byte binary coded decimal (BCD), and the value range is between 1 and 99 999 999. The HOT_BILLING_RECORD_NUMBER will be initialised to 1 during a software upgrade or if the working (active) CHU in the MSS is restarted. The HB works even in case the HOT_BILLING_RECORD_NUMBER field is not selected to customer-specific CDR format, but HB CDRs can only be identified in the post-processing system if HOT_BILLING_RECORD_NUMBER is included in the CDR format.

Restrictions

Both the MSS and HLR must be Nokia products as the HB supplementary service is Nokia-proprietary extension.

There are some CDRs which cannot be identified as HB CDRs and which cannot be sent immediately to the post-processing system. These CDR types do not have HOT_BILLING_RECORD_NUMBER available in the CDR format.

Some supplementary services (depending on supplementary service code) generate a SUPS CDR which cannot be identified as a HB CDR and the CDR is not sent immediately to the post-processing system.

Transfer capacity of Hot Billing

Hot Billing is designed to be used only for a limited number of subscribers, not for all subscribers. The exact limit depends on many variables, such as the number of CHU pairs, the interface in use, the structure of the network, the size and format of the CDRs, and parameters controlling the generation of CDRs in the different call cases. The intermediate charging interval may also have an impact: the shorter the interval, the more CDRs are produced per call on the average. The following capacity calculations are based on the assumption that the CDR amounts and sizes are within the limits stated in the *Design Capacity* documents.

Sending individual charging records to the post-processing system entails an extra load, which must be taken into account when defining the maximum number of HB subscribers. Collecting all of the charging data through the HB is not possible (In case immediate CDR transfer is needed for all subscribers, see section *Immediate CDR Transfer for all subscribers*.) The capacity of the HB is mainly restricted because of the following factors:

- sending method (GTP')
- CPU load of charging unit
- capacity of RAM buffer

The restrictions for the transfer methods are described in the protocols section. The HB has an increasing effect on the CPU load of the CHU.

The RAM is reserved (CHU) for buffering the HB records. HB RAM buffer size is 16 MB. The RAM buffers are originally designed to hold charging data for five minutes when a switch is running within design capacity load limits. You must be aware of the buffer size before provisioning.

General alarms of Hot Billing

There are two types of alarms: general alarms which apply to all transfer methods and alarms which apply only to the used transfer method. The general alarms are as follows:

- When sending charging data fails, the alarm `2233: FAILURE IN SENDING HOT BILLING DATA` is set by the application. In this case either the online link is down, or the storing of a CDR to the HB CDR files on the CHU's disk has failed (or both), depending on the configuration.
- If the RAM buffer for the HB CDRs reaches the limit defined with the GBA MML command, the alarm `2234: HOT BILLING RAM-BUFFER IS FILLING UP` is generated. It means that for some reason more HB CDRs have been placed in the RAM buffer than the system is able to send to the logical file HBLOFI because of a link failure or a temporary burst of HB CDRs. In the latter case, the alarm should be automatically cancelled after a while.
- If the application starts overwriting the not sent CDRs in the RAM buffer it generates the alarm `2235: HOT BILLING DATA HAS BEEN LOST`. In this case the situation described above has lasted long enough to fill the RAM buffer. Normally it means that the link is down for some time. After this, the overwritten CDRs cannot be sent as hot CDRs.

The alarms which apply only to a selected transfer method are described in the section of each transfer method.

2.2.3.1 Storing Hot Billing CDRs on hard disk (for testing purposes only)

Due to performance reasons, separate HB disk files should be used only for testing purposes. The CDRs are stored one-by-one to the mass memory device, which is a very inefficient method, and causes a high load on the device drivers and on the mass memory equipments. It also has a strong decreasing effect on the total throughput of the charging unit.

The VDS-1 device (application name GSMBIL) is reserved for this purpose. First the amount and size of files BF0nnn.DAT (or BF0nnn.Z in case of compression) have to be defined with the MML command `IFF`, IF Command Group. The optimization mode can also be defined with the same command. As the HB CDRs are stored on the device one-by-one, the optimization mode PF512K slightly increases the capacity as compared to the default 'OFF' value. Note, however, that even with this optimization mode, this storing/transfer method of the HB has a strong impact on the performance of the normal charging data storing. This is why the storing/transfer method should only be used for testing purposes. The HB files are into their own directory (GSMBIL). Other relevant parameters for these files (alarm limits, and so on) can be set with other commands in the 'IF' command group the same way as for normal charging data files.

Storing in these files is enabled with the following parameters:

- Parameter `SENDING_STATE` in the HBPARA file. This parameter can be set with the MML command `GBS`, GB Command Group.
- Parameter `CURRENT_I/O_DEVICE`. With this parameter the logical file HBLOFI is connected to the VDS of the CHU. Use the MML command `IIS`, II Command Group. Use the VDS-1 since the VDS-0 is used for storing normal CDRs.

The separate HB disk files are located on the hard disks of the CHU. The disk files are located in the GSMBIL directory and they are named 'BF0xxx.DAT', or 'BF0xxx.Z' (in case of compression), where the *xxx* is the ordinal number. The files act as ring buffers and the ordinal number can range from 001 to 999. The related parameters can be handled with the MML commands of the IF Command Group.

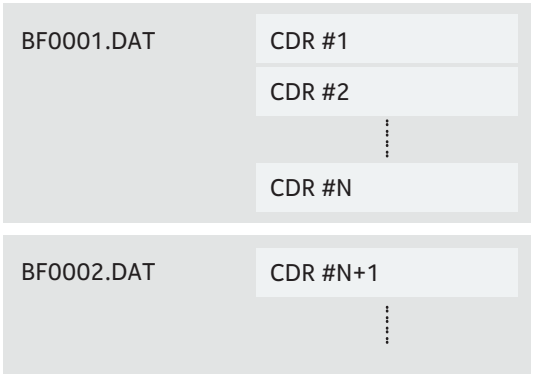
These files can be transferred to the post-processing system with the same mechanism as in the transfer of normal charging data. A separate pair of control files 'TTSCOF01.IMG' and 'TTTCOF01.IMG' is also used.

The HB disk files do not have a header or a trailer. If the length of a CDR is larger than the free space in the disk file, the CDR is written into the next file.

If it is necessary to find specific HB CDRs (that have not been transferred online because of a link failure, for example), the right files can be found by checking the timestamps of the files with the MML command `IFO`, IF Command Group. The post-processing system will then find the right CDRs from the file.

The structure of HB CDRs on the disk files is shown in .

Figure 9 The structure of hot billing CDRs on disk files



An example of HB charging data on a separate HB disk file is shown in .

An example of hot billing charging data on disk file

```
        1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16

0000    BE 00 01 05 40 45 10 00 1F 5D 32 41 10 00 91 58 MOC
0010    92 20 00 01 F0 FF FF FF FF 00 00 00 01 02 03 04
0020    04 05 06 07 08 FF FF FF FF FF FF FF FF 01 02 03
0030    04 05 06 FF FF FF FF 00 03 FF FF FF FF FF FF FF
0040    FF FF FF FF FF FF FF FF FF 00 01 02 03 04 FF FF
0050    FF FF FF FF FF FF 00 FF 11 16 33 F1 FF FF FF FF
0060    FF FF FF FF B8 22 3A 27 58 92 20 00 01 F0 FF FF
0070    FF FF B8 22 3A 27 FF FF FF FF FF FF FF FF FF FF
0080    FF FF FF FF FF FF FF FF 04 01 18 00 00 11 FF FF
0090    00 00 00 14 02 17 02 94 19 01 14 02 17 02 94 19
00A0    04 14 02 17 02 94 19 16 14 02 17 02 94 19 11 00
00B0    00 00 00 00 00 00 00 03 02 00 00 FF FF 00 8D 00 MTC
00C0    02 06 40 45 10 00 34 36 41 40 51 00 83 58 92 20
00D0    20 00 01 F0 FF FF FF FF 00 00 00 FF FF FF FF FF
00E0    FF FF FF FF FF 01 02 03 04 05 06 07 08 FF FF FF
00F0    FF FF FF FF FF 01 02 03 04 05 06 07 08 09 10 11
0100    12 00 03 11 01 06 01 B8 22 C2 4E 58 92 20 00 01
0110    F0 FF FF FF FF B8 22 C2 4E 00 11 FF FF 00 00 35
0120    13 02 17 02 94 19 35 13 02 17 02 94 ……
```

2.2.3.2 Hot Billing CDR transfer using COTS (for backwards compatibility)

The HB CDRs can also be sent through the Connection-Oriented Transport Service (COTS) interface immediately from an exchange to a post-processing system. This method is also faster than the transfer via the logical file HBLOFI, and also provides immediate confirmation by receipt from the post-processing system. When using COTS, HB CDRs can be sent to a post-processing system by using LAN as the transfer method.

Note: COTS uses an OSI stack when the LAN interface is used. This means that an OSI stack support is required from routers. Implementation can be used with or without acknowledgement. When acknowledgement is used, the acknowledgement window is 1, which makes this protocol inefficient when compared to the GTP'.

When the COTS is taken into use, the interface for HB data between the MSS and the post-processing system can be set to support the connection with an immediate confirmation. The operator can choose whether the COTS transfer path is acknowledged or not. If the interface is equipped with an immediate confirmation, the post-processing system must be able to confirm all received data.

As a CDR leaves the exchange, a counter is attached to it. The counter is then detached by the post-processing system before the CDR is processed. This counter is then sent back to the exchange with the confirmation to verify each sent hot record with its respective received confirmation. The counter is an ascending number which is independent of any possible sequence number inside the hot record data. The value of the field is presented in doubleword format. When the post-processing system confirms the received data, the same counter has to be included in the confirmation message. Furthermore, the confirmation message from the post-processing system includes a status byte, which tells if the received data was correct or not.

From the capacity point of view, the greatest capacity restriction is the time taken by the post-processing system before it sends the confirmation. Capacity can be significantly increased by transferring the records without confirmation. However, if there is no acknowledgement and a problem arises in the post-processing system, charging data could be lost during the delay before the problem is noticed in the normal way.

If a higher sending capacity is required, the MML command GBT, GB Command Group can be used to define that records should be sent with or without confirmation by the post-processing system.

The capacity is sufficient and hot records cannot be lost if a LAN is used and the post-processing system (the BC, for example) confirms all received hot records. An advantage of using online data transfer with an immediate confirmation through the COTS is that hot records can be transferred safely without having to save them with the normal charging data.

2.2.3.2.1 Requirements for using COTS

There must be a LAN connection between the MSS and the post-processing system and the post-processing system must be able to receive data packages and confirm all received data packages.

2.2.3.2.2 Parameters

The original transfer method through the I/O system via the logical file remains to be the primary method of transfer. The data transfer path through the COTS can be set with the MML command GBT, GB Command Group. The HB CDRs are transferred via the COTS interface, one at a time, and the post-processing system continues to confirm all sent CDRs.

In the Hot Billing Parameter File, there are two parameters for each CHU pair which can be set with the MML command GBT:

Transfer path	The path for sending hot records. Possible values are HBLOFI and COTS.
Application entity name	The name specifying the remote application entity.

If the exchange is equipped with only one CHU pair, only one transfer path and application entity name pair can be defined.

2.2.3.2.3 Fault conditions

Alarm 'HOT DATA CONNECTION CHANGED TO DEFAULT PATH' (2700) is set if the COTS transfer path fails and the transfer path is automatically changed to the default I/O services through the logical file HBLOFI. This alarm is cancelled after the COTS transfer path has been successfully defined.

2.2.3.2.4 Capacity

The CDR transfer capacity increases as compared to the traditional hot record transfer through the HBLOFI. The most significant capacity restriction is the time taken by the post-processing system before it sends the confirmation. The load of the LAN network does not appear to present any significant restrictions. Transfer capacity can, however, be significantly increased by transferring records without asking for confirmation. When the value of the acknowledgement state parameter in the GBT command is set to 'OFF', no confirmations are returned by the post-processing system and therefore additional capacity is available for sending hot records.

2.2.3.3 Hot Billing CDR transfer using GTP'

All new HB implementations should use the GTP' for CDR transfer.

The HB CDRs can be sent to a post-processing system over the LAN when using the GTP' over UDP/IP (or over TCP/IP) as a transfer method. Both IPv4 and IPv6 are supported. The supported protocol is the GTP' version 2. Backwards compatibility with versions 0 and 1 is not supported. When the GTP' is used, it is possible to use a higher transfer rate because several records can be sent in the same packet and several packets can be sent before waiting for an acknowledgement from the post-processing system.

The normal storing procedure between the MSS and the post-processing system is carried out with Data Record Transfer Request and Data Record Transfer Response messages. The Data Record Transfer Request message contains the information element Packet Transfer Command which indicates the nature of the message, such as Send Data Record Packet. The information element Data Record Packet in Transfer Request contains the actual data records.

The HB using the GTP' feature also includes duplicate packet prevention scheme. If the post-processing system does not acknowledge the HB CDR package, the CDR package is sent to another post-processing system. To prevent both of these post-processing systems from handling these CDRs, the second CDR package is marked as 'possible duplicate'. The MSS automatically interrogates whether the first post-processing system has processed the CDR package or not. This interrogation takes place automatically when the first time system sends an alive request to the MSS or sends an acknowledgement (echo response) to an echo request made by the MSS. If the CDR package has already been processed in the first post-processing system, the MSS informs the second post-processing system to discard the CDR package. If the CDR package is not processed in the first post-processing system, the MSS informs the second post-processing system to go ahead with processing the CDR package. If the first post-processing system has not acknowledged the live inquiry or has not sent the live notice to the MSS within 25 hours of the sending of the CDR package, the MSS considers the CDR package lost. In this case the MSS does not send any information to the second post-processing system. If the second post-processing system does not receive any information from the MSS on how to act with the 'possible duplicate' CDR packages within 25 hours, it may, for example, continue with processing those CDRs (though the implementation of the post-processing system should keep in mind that there is still a slight possibility that the CDR is already processed by the first post-processing system in this case).

When the GTP' is used as transfer protocol of Hot Billing, the interface is compatible with the GTP' version 2 specification. In Hot Billing usage interface also accepts the GTP' version 0 and 1, but only with a 6 byte GTP' header.

For more information on the GTP' protocol, see *3GPP TS 32.215 Charging data description for the Packet Switched (PS) domain* and for more information on the configuration of TCP/IP for Hot data transfer, see *Hot Billing Handling*, GB command group.

2.2.3.3.1 Parameters

Use the MML command GBE to define the parameters of IP transfer used in all CHU pairs of the exchange. Use this command to define the IP parameters and the command GBX to set the IP addresses used in the exchange before setting the IP specific addresses of the CHU pairs with the GBT command. For more information, see GB Command Group.

The parameter values you define for transfer using GTP' over TCP/IP or over UDP/IP are used by all the CHU pairs.

Hot CDRs are first saved in packets, then the packets are sent to the post-processing system. The sending window parameter defines the number of packets that can be sent before the acknowledgement is required from the post-processing system if the acknowledgement status is enabled. If the acknowledgement status is disabled, the packets are sent continuously as long as the transfer path is functioning. If the acknowledgement status is disabled, the packets are sent continuously as long as the transfer path is functioning.

If the acknowledgement status is enabled, then you also have to define the acknowledgement waiting time . If the acknowledgement is not received within the specified maximum waiting time, the system re-sends all hot CDRs that were sent since the previous successful acknowledgement has been received. The packets are normally sent when they are full. However, when volumes are low, the delay before a packet is full could be unacceptable. Therefore the maximum delay before a packet is sent should also be defined. If a packet has not been sent yet at the end of this delay period, it is sent to the post-processing system even if the packet is not full, provided that it contains at least one hot CDR.

Example

Configuring the IP parameters

Define the IP transfer parameters to be used by all CHU pairs. One GTP' packet includes MANY CDR's, hot records can wait 200 seconds before the packet is sent even if it is not full, the acknowledgement status is set to YES, the size of the sending window is 10 and acknowledgement waiting time is 15 seconds.

```
ZGBE:CDRS=MANY,DEL=200,ACK=YES,WIN=10=,WAIT=15;
```

Example

Defining the IP address

Use the GBX command to define the IP addresses used in the exchange before selecting CHU-pair specific IP addresses and their priority order with the GBT command.

Only one IP address can be defined with one command.

Define the IP address and port of index 2 to 121.221.56.65 and 3000.

```
ZGBX:2,"121.221.56.65",2000:;
```

Further information

For more information, see *Hot Billing Handling*, GB command group and *Charging Handling*.

2.2.3.3.2 Fault conditions

If acknowledgement is not received from the post-processing system, the system changes the IP address, sets the alarm 'IP ADDRESS FOR HOT BILLING DATA CHANGED' (0090) and re-sends all packets starting from the last acknowledgement received. If a packet is acknowledged as faulty, the system re-sends the contents of all packets starting from the faulty packet. If all IP addresses fail or a connection cannot be set up, the system sets the alarm 'HOT BILLING DATA TRANSFER USING IP FAILED' (3114) and tries to connect every 15 seconds. Once a connection is established the alarm is cancelled.

2.2.3.3.3 Capacity

This transfer method is fast because groups of hot CDRs are sent in packets instead of single CDRs. In addition, if acknowledgement is required, up to 10 packets can be sent before the acknowledgement from the post-processing system. A maximum of 10 different IP addresses can be defined for the post-processing system. If there is more

than one CHU pairs, any combination of the defined addresses can be set for each CHU pair; their order of priority can also be defined. Different CHU pairs can use different transfer methods. The packets containing the hot CDRs are sent to the post-processing system either when the current packet is full or when the defined maximum waiting time has expired. You can determine the number of CDRs in one packet by setting the size of the packets in kilobytes.

2.2.3.4 Storing HB CDRs in normal charging files

The hot CDRs should also be stored in the normal charging files as backups in case of a possible loss of CDRs during the transfer as well as in link failure situations. If the HB CDRs are stored in the normal charging files, the storing is enabled with the following parameter:

- STORING_METHOD in the CHPARA file. This is set with the MML command GTN, GT Command Group.

The hot CDRs stored in the normal charging files are the same as the ones sent online (or stored in separate files).

To be able to differentiate the hot CDRs from the normal CDRs in the normal charging data files, the field HOT_BILLING_RECORD_NUMBER must be included in the CDRs. In hot CDRs, this field holds the sequential number of the hot CDR. In normal CDRs this field is empty (FFFFFFFFH).

If it is necessary to find specific HB CDRs (they have not been transferred online due to a link failure, for example), the right files can be found by checking the timestamps of the files with the MML command IFO, IF Command Group. Then the post-processing system has to find the right CDRs from the file.

2.2.4 Immediate CDR Transfer for all subscribers

> *CDRs are transferred immediately to the BC using GTP'. If immediate transfer is not possible, the Fall Back Saving functionality guarantees that CDRs are not lost.*

This transfer method is used when *Feature 1491: Immediate CDR Transfer for All Subscribers* is activated.

When the charging block fills up, its content is immediately transferred to the post-processing system. All charging blocks for which the immediate transfer task or the Fall Back Saving storing task is not yet acknowledged are buffered in a large RAM memory buffer, which can hold charging data of approximately 5 minutes when switch is running at the limits of the design capacity. If the RAM buffer fills up to the alarm limit or the RAM buffer contains data which is older than 15 minutes, the Fall Back Saving functionality will be activated.

Figure 10 Functionality of immediate CDR transfer for all subscribers

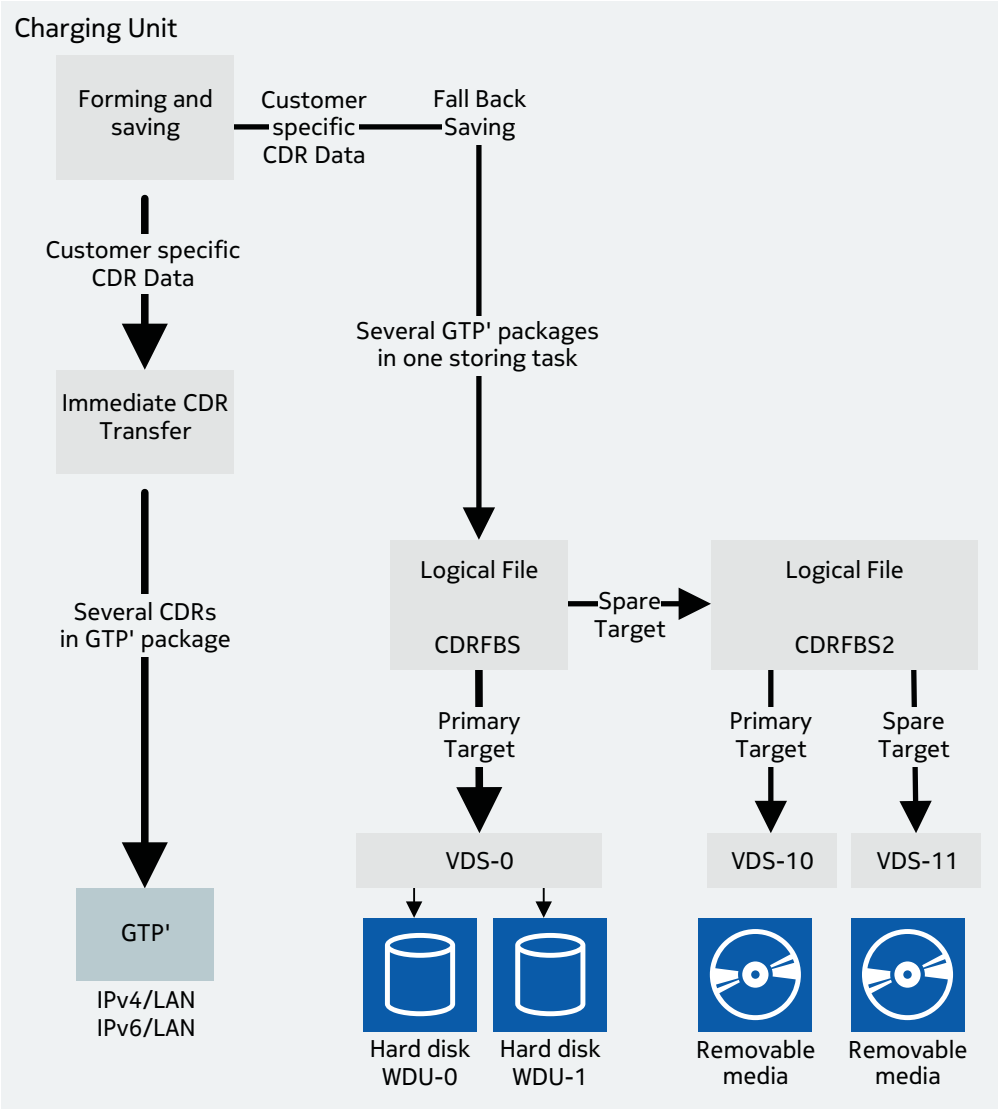
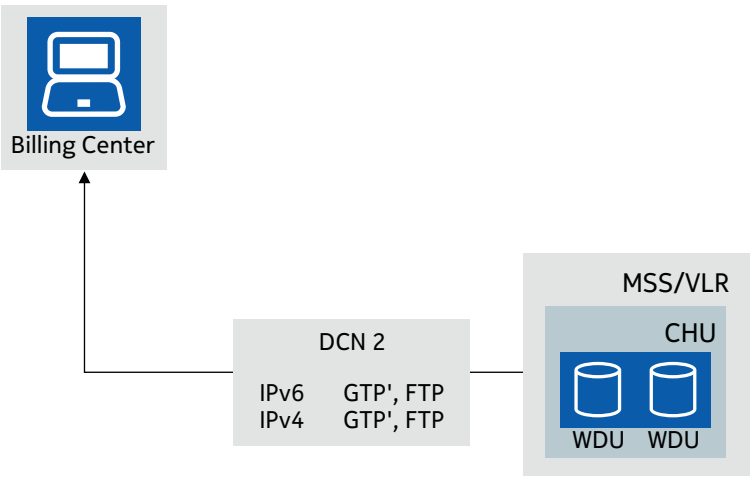


Figure 11 Immediate CDR Transfer from MSS to BC



For more information, see *Feature 1491: Immediate CDR Transfer for All Subscribers, Feature Description*.

2.2.4.1 Immediate Transfer

Normally CDRs are transferred immediately to the post-processing system using the GTP' interface. The functionality of this interface follows the GTP' specification. The GTP' is a CDR transfer protocol which is standardised in the GPRS network. The same protocol is the preferred transfer protocol of *Hot Billing*.

For more information about the GTP', see *GTP' Interface*.

2.2.4.2 Fall Back Saving

In case the immediate transfer fails to all destinations, or the immediate transfer is too slow, the CDRs are stored on local mass memory equipment. This functionality is called a Fall Back Saving (FBS). For optimal performance, the FBS stores multiple GTP' packages using one storing task. The data is stored by means of a logical file CDRFBS.

Normally, the CDRFBS is connected to a VDS-8 device. The amount of files depends on the capacity of hard disks. A recommended file size for a VDS-8 device is 256 MBs (262144 kB). For example, when using 300 GBs disks on ATCA, 1140 files is a reasonable value.

Note: The files of a VDS-0 device should be deleted from the disk when *Feature 1491: Immediate CDR Transfer for All Subscribers* is activated, so that there is enough room for the FallBackSaving files in the hard disks of the CHU.

As an alternative, it is possible to write the FBS files directly to a removable media. In this case, the primary connection of the CDRFBS is made to the VDS-10 device, and a CHG-type spare connection is created to the VDS-11 device. VDS-10 should be configured to write on the UMS-0, and VDS-11 should be configured to write on the UMS-1. The file size should be the same as with hard disks (10MB) and the amount of files depends on the capacity of the removable media to be used.

Note: Since the CHU pairs cannot access each other's UMS devices, the selected UMS devices must have a mass storage device plugged-in both in the WO-EX and SP-EX units. Due to the same reason, only the WO-EX CHU unit (which obviously changes with switchover) writes the data to its own USB device.

The files created by the FBS contain GTP' packages with GTP' headers and Information Elements. The length of the GTP' packages varies. There is no additional fillings which would make the package to have a fixed length. (In other words the content of these FBS files is not identical with the traditional CDR files.)

An example of charging data on FBS disk file

```
1  2  3  4  5  6  7  8   9 10 11 12 13 14 15 16
```



```
00000  4E F0 01 28 12 34 7E 01  FC 01 23 02 0B 40 40 00 GTP', LOCA
00010  3E 07 01 00 00 00 00 0C  24 FF FF FF FF FF 94 71
00020  37 89 FF FF FF FF FF FF  62 02 53 18 11 51 00 F5
00030  88 88 FF FF FF FF FF FF  FF FF FF FF 88 88 94 71
00040  37 89 FF FF FF FF FF FF  54 40 08 05 06 97 19 00 MOC
00050  DD 01 02 00 00 00 00 7F  5F 33 41 27 01 01 94 71
00060  37 89 FF FF FF FF FF FF  00 00 00 62 02 53 18 11
00070  51 00 F0 10 20 30 40 50  60 70 F0 94 71 57 58 00
00080  F0 FF FF FF FF 00 00 62  02 53 18 11 51 00 F1 10
00090  20 30 40 50 60 70 F0 06  71 57 58 00 F1 FF FF FF
000A0  FF FF FF FF 00 04 10 77  85 05 10 FF FF FF FF FF
000B0  FF FF 88 88 82 03 94 71  37 89 FF FF FF FF FF FF
000C0  88 88 82 03 88 88 82 03  88 88 82 03 01 00 02 00
000D0  00 11 10 00 FF 18 46 41  08 05 06 97 19 53 41 08
000E0  05 06 97 19 08 42 08 05  06 97 19 00 00 00 00 03
000F0  00 FF FF FF 00 00 15 00  00 00 02 00 FF FF 06 71
00100  97 02 00 F1 FF FF FF FF  FF FF FF FF FF FF FF 05
00110  06 71 57 58 00 F1 FF FF  FF FF FF FF FF FF FF FF
00120  FF FF FF FF FF FF 00 FF  FF 33 41 27 01 01 4E F0 GTP'
00130  05 86 12 35 7E 01 FC 05  80 09 0B 40 40 00 9A 02 MTC
00140  03 00 00 00 00 51 3B 33  41 27 01 01 94 71 37 89
00150  FF FF FF FF FF ……
```

2.2.4.3 Capacity

The assumption is that this feature does not limit the amount of charging data from the figures presented for traditional file-based CDR transfer. The official capacity figures are released in the *Design Capacity* documents. It is recommended to adjust the size of the sending window parameter according to the amount of traffic in the exchange, for example, by increasing the size from default value during high traffic period.

2.2.5 GTP' Interface

> *A GTP' interface is used as a CDR transfer interface in case of immediate CDR transfer for all subscribers, and it is the only interface of Hot Billing.*

The interface is a standard interface in the GPRS networks. It has been adopted to circuit-switched network elements as it has been found to be a very reliable and an efficient protocol for the immediate CDR transfer. In MSS, a GTP' interface is compatible with the GTP' version 2 specification. The interface is not backwards compatible with versions 0 and 1. The following sections present a few requirements which are not presented in standards and a few interpretations of the standard. These sections do not include full description of the GTP' interface, which can be found from the GPRS standards (*3GPP TS 32.215 v5.2.0* was used in design of the *Feature 1491: Immediate CDR Transfer for All Subscriber*). In some parts, the GTP' definition refers to the GTP definition which can be found from GPRS standards (*3GPP TS 29.060 v5.2.0* was used in the design of the *Feature 1491: Immediate CDR Transfer for All Subscriber*).

2.2.5.1 Maximum size of GTP' packet

The maximum size of the GTP' package can be defined. As the preferred transfer protocol under the GTP' is the UDP, the size of the GTP' package defines whether or not the IP package is fragmented or not. To avoid problems which may occur because of IP packet fragmentation, the size of GTP' package must be defined so that the Maximum Transmission Unit (MTU) is not exceeded.

In an Ethernet link the MTU is 1500 bytes, this must include the IP header, the UDP header and the GTP' package. The best possible size of the GTP' package differs as the protocol overhead differs. The size of an IP header is different between IPv4 and IPv6, and the possible use of IPsec causes header extension, the size of which differs depending on mode (transport/tunnel) and encryption algorithm (for example, 3DES, AES). The following table defines the optimal GTP' packet sizes.

Table 4 Optimal GTP' packet size

Protocol definition	Protocol overhead	Max GTP' packet size with Ethernet (dec)	Max GTP' packet size with Ethernet (hex)
IPv4 + UDP	28	1472	0x05C0
IPv6 + UDP	48	1452	0x05AC
IPv6 + ESP Transport (3DES) + UDP	86	1414	0x0585
IPv6 + ESP Transport (AES) + UDP	102	1398	0x0576
IPv6 + ESP Tunnel (3DES) IPv4 + UDP	106	1394	0x0572
IPv6 + ESP Tunnel (AES) IPv4 + UDP	122	1378	0x0562
IPv6 + ESP Tunnel (3DES) IPv6 + UDP	126	1374	0x055E
IPv6 + ESP Tunnel (AES) IPv6 + UDP	142	1358	0x054E

The `MAX_GTP_PACKET_SIZE` (001:0156) PRFILE parameter defines the maximum size of a GTP' packet in bytes. The default value is 0x054E. If the used protocol allows a greater GTP' packet size without IP fragmentation, the value of this parameter can be increased for optimal performance.

If the value defined is too high, the IP packages will be fragmented. In optimal environments, this does not cause problems, though in many cases the IP packet fragmentation causes problems which are almost impossible to monitor and detect. If IP fragmentation is not a problem in the used network environment, the maximum GTP' packet size is 32768 (0x8000) bytes.

2.2.5.2 Sequence Number

The sequence number in the GTP' header is used to detect duplicate packages. The post-processing system must keep a record on which packages (GTP' sequence numbers) are already received and which are not. If the acknowledgement is lost, the GTP' package with the same sequence number (and without duplicate package indicator) can be sent several times to the same post-processing system. The possible duplicate handling scheme presented in the GTP' standard takes place only when the post-processing system to which the data is delivered changes. The range of sequence numbers is from 0 to 65535, after 65535 sequence number will wrap to 0. The post-processing system will keep in mind which is the next expected sequence number.

If the received sequence number is the expected sequence number or any of 32767 following numbers, the package is accepted and the next expected sequence number will be the number following this package's sequence number. The package (the sequence number) is also marked as a received package and all the packages (sequence numbers) which are skipped are marked as not received packages. If the sequence number does not match this range (it is 'older' than the next expected sequence number), the post-processing system will check if this package is already received. If the package is already received, the new instance of this package will be discarded and a cause code #253 REQUEST_ALREDY_FULFILLED will be sent in the acknowledgement. If the package (the sequence number) was not received, the package will be accepted (and a response will be sent), and the package will be marked as received to the boolean array.

With this method, the GTP' packages may come in an out of order sequence, unique packages are accepted and duplicate packages are discarded. Only restriction is that the out of order is limited within 32767 packages. For example, packages may come in the following sequence and still be accepted: #0, #32767, #1, #2 ... #32766, #32768, #32769 ... #65535, #0, #2 ... #32767, #1, #32768.

In some rare conditions some sequence numbers may be skipped, thus the sequence number in the GTP' header cannot be used to detect whether all packages are received or not. The post-processing system may check whether all CDRs are received by checking the record_number from the actual CDR data (not part of GTP' specification).

2.2.5.3 Node Alive Request with TCP

The UDP is the preferred protocol under the GTP'. The GTP' itself handles end-to-end acknowledgement so, in practice, the TCP does not offer any additional value compared to the UDP. If the post-processing system requires that the TCP must be used, it should be noted that the TCP connection is always established from the MSS to the post-processing system. The MSS does not listen to the TCP connection requests, which is why the TCP connections cannot be established from the post-processing system side.

In practise this means that, when the TCP is used it is almost impossible for the post-processing system to send a NodeAliveRequest after restart (of the post-processing system). This is not a problem as the MSS uses EchoRequests to check when the link is re-established. See section *Re-establishment of a broken link*. However, for compatibility, the post-processing system has to accept that it cannot establish TCP connection to the MSS and send NodeAliveRequest at startup.

2.2.5.4 Re-establishment of a broken link

The MSS sends regular EchoRequests to links from which it has not received acknowledgements. If the MSS receives an EchoResponse or a NodeAliveRequest, the link is considered to be up and running.

2.2.5.5 Clear up of hanging possible duplicate packages

If the CDRs are delivered with a possible duplicate indicator, the post-processing system will wait for instructions on whether the CDRs are to be cancelled or released for billing. To ensure that the post-processing system does not wait for too long for possibly non-existing instructions, the CDRs should be automatically cancelled or released for billing if instructions are not delivered within 24 hours (if there has been a failure which has lasted over 24 hours, it may cause duplicate CDRs). The MSS tries to create a connection to the first post-processing system to check if the CDR is already handled there or not. If the MSS cannot contact the first post-processing system within 25 hours, the MSS assumes that the CDR is already released by the second post-processing system and the MSS will stop the duplicate removal handling.

2.2.5.6 Data Record Format

The data record format is a proprietary format, and the same format is used as in normal charging data with two exceptions (see section *CDR format*). The data record format identifier in the Data Record Packet IE is marked with the value 11.

2.2.5.7 Data Record Format Version

The data record format is a proprietary format, thus the interruption of the Data Record Format Version field of the Data Record Packet IE is also a proprietary. The version field contains the version of the CDR format without a customer_id. The value is in big-endian byte order and the most significant 4 bits represent version, the next 6 bits represent edition, and the least significant 6 bits represent correction.

2.2.5.8 CDR format

The CDR format is identical to the normal charging data, with the following exceptions:

- The length field is defined in the GTP' standard. The length of the CDR is a part of the Data Record Packet IE, and it is presented in big-endian byte order. The value of the length field contains only the length of the CDR, and it does not contain the length of the length field.
- The type of the CDR (the first byte of the actual CDR) is identical to the normal charging data. The value is BCD-coded. In some cases, the same post-processing system may receive data from the GPRS network elements and the MSS. In such cases, the MSS is able to add the value 50 to the CDR type field when forming the CDR. This is controlled by the RECORD_TYPE_BASE_VALUE (001:0149) PRFILE parameter.

2.2.5.9 Redirection Request

All Cause IE values received in a RedirectionRequest are handled in the same way, that is, there is no separate way of handling certain Cause IE values, and all Cause IE values are considered valid. The address of the Recommended Node IE in a RedirectionRequest can include either an IPv4 or an IPv6 address. An alternative address of the Recommended Node IE received in a RedirectionRequest is not taken into account, but discarded.

If the address of the Recommended Node IE received in a RedirectionRequest is defined in the priority list of the configured IP nodes, the CDR transfer will be redirected to that node. If the received IP address is not defined or it can not be found in the priority list, the CDR transfer will be redirected to the IP address of next priority in the priority list. This is why it is necessary to define at least two IP addresses in the priority list.

2.2.5.10 Redirection Response

If the address of the Recommended Node IE received in a RedirectionRequest can not be found from the priority list of the configured IP nodes, a RedirectionResponse will be sent with the cause code #255 REQUEST_NOT_FULFILLED in the Cause IE.

2.2.6 Using hard disks as storing devices

> *The process of storing can have four different states. When the charging data file is in FULL state, it is ready to be transferred. After transferring, the file can be reused for storing.*

Charging blocks are written in a hard disk file until the file reaches the predefined size, after which the file is closed. A certain time period (for example 30 minutes) can also be set after which the charging file is closed. A new charging file is opened after the first block has been written. If the file does not exist on the hard disk, the system will create it.

Charging data files are controlled with the charging control named TTSCOF and TTTCOF. The TTSCOF contains information on states, timestamps, and on the storing status of data files. The TTTCOF contains a timestamp which shows the date and time when the VDS disk file is transferred.

A charging data file can be in four different states, which are presented in *Table: The possible states of charging disk files* .

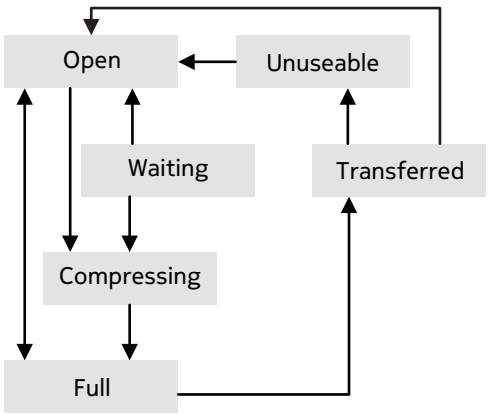
Table 5 The possible states of charging disk files

State	Explanation
OPEN	The data is being written on the file. Only one file can have OPEN state at a time.
FULL	The file is ready to be transferred to the post-processing system.
TRANSFERRED	The file has been transferred to the post-processing system and can be used again (overwritten) when necessary.
(WAITING)	Obsolete. In previous releases, this was used to queue files to file compression process. Now blocks are compressed using ZLIB library before block is stored on disk and so the compression queue is now obsolete.
(COMPRESSING)	Obsolete. In previous releases, this was used to identify file which is currently been compressed by file compression process.
UNUSEABLE	The file is unuseable because the hard disk is full and there is no space to create this file.

When the charging data file becomes full, the TTSCOF is updated with a state FULL, and a timestamp of the file closing time. The timestamp field in the TTSCOF indicates the last time when the state of a charging disk file has been changed from 'OPEN' to 'FULL', or from 'TRANSFERRED' to 'OPEN'. When the state of the charging disk file changes from 'FULL' to 'TRANSFERRED', the timestamp of the TTSCOF is not updated. The possible state transitions are shown in *Figure: State transitions of charging disk files*.

Note: The state of charging disk file can also be changed from 'FULL' to 'OPEN' by allowing the overwriting (not recommended). For more information, see *IF - Virtual Data Storing Device Handling, Command Reference Manual*.

Figure 12 State transitions of charging disk files



When the first block is stored on a disk file, the file is set to 'OPEN' state. Normally, the previous state of the opened file has been TRANSFERRED (*1). The files are used in sequential order, though it is possible to enable skipping of FULL files. In this case, the system searches for the next file which is in 'TRANSFERRED' state. If skipping is not enabled or there are no TRANSFERRED files, the system may also open FULL file (*1a) if overwriting is enabled (not recommended).

If the creation of the file is unsuccessful because there is not enough space on the disk or in the directory structure, the system finds the next file which already exists on the disk. All skipped (non-existing) files are marked as UNUSEABLE (*1b). During the next

round, the system tries to open the first UNUSEABLE file (*1c). If it is not successful, all the succeeding UNUSEABLE files are skipped (without the time consuming existence checking).

When the size of the file reaches a predefined limit or the file has been in 'OPEN' state for a predefined maximum time, an OPEN file is closed and marked as 'FULL' (*2). When the transferring timestamp is newer than the storing timestamp, the FULL file is marked as 'TRANSFERRED' (*3) .

Note: The post-processing system can also cancel the previous acknowledgement by setting a transferring timestamp which is older than the storing timestamp. If the TRANSFERRED file has an older transferring timestamp than the storing timestamp, the file is marked to 'FULL' state (*4).

The state of the disk files can be changed manually with an MML command. When the state is changed from 'TRANSFERRED', or 'OPEN', or from 'UNUSEABLE' to 'FULL', the timestamp field of the TTSCOF is updated with the current time. When the state is changed from 'FULL' or from 'UNUESABLE' to 'TRANSFERRED', the timestamp field is filled with '00's.

The updating of the TTSCOF file state information by comparing the timestamps of TTTCOF and TTSCOF is not performed immediately after TTTCOF is transferred to disks of MSS. The timestamps are compared in the following cases:

- The charging disk file is changed.
- The `IF0`, `IFS`, `IFF`, `IFP` MML commands are used.
- After a CHU switch-over when the first data block is stored.
- After a CHU restart when the first data block is stored.
- The minimum timer goes off (an interval of this timer can be set using `IFH` command).

2.2.6.1 Compression of charging data files

Data compression can be used to compress the charging data produced in the MSS before transferring it using FTP. From the charging point of view, the compressed data can be either normal charging, hot billing test files or accounting data, provided that the data has been saved by using a VDS device. The data has to be decompressed in the destination equipment before it is used.

The compression is done with the public domain ZLIB compression library. The MSS writes a GZIP compatible header and trailer to the compressed data files. The ZLIB uses the Lempel-Ziv algorithm, which is also used in the ZIP and the PKZIP. The compression of a charging block is done in the memory before storing on the disk. The extension of a compressed file is '.Z'. In case of file compression, it is possible to store also uncompressed data on the hard disk as a backup. According to the tests, the compression ratio for normal charging data is about 20 percent. This means that the transfer capacity is about five times faster than without compression.

Note: The compression procedure is processor resource intensive, but compressed data can be transferred much faster. The compression of charging data files uses a compression level that is optimized for speed, and therefore CPU utilization is not significant during normal operation of the MSS.

Note: The storage of compressed files is more effective with an introduced parameter change. For more information, see
IF - Virtual Data Storing Device Handling, Command Reference Manual.

2.2.6.2 Charging control files

Two charging control files, TTSCOF and TTTCOF, are needed to transmit the control information concerning charging files between the MSS and the post-processing system. These files are located in the hard disks of each CHU. For the location of these files, see section *Configuration of the VDS system*.

The size of the control files is dynamic and depends on the amount of the data files that have been defined. Both control files contain one record for each charging disk file. The number of the last record is the same as the number of the last charging data file (for example*CF1234.DAT*).

TTSCOF (VDS Device Data Storage Control File)

The TTSCOF controls the charging data storing and it is located in the hard disks of the CHU. The TTSCOF is updated only by the VDS of the CHU.

For each VDS device, there is one TTSCOF. The TTSCOF name is *TTSCOFxx.IMG*, where the *xx* is the number of the VDS device (see Table 9: VDS System).

The first record (record number zero) of the TTSCOF is reserved for the internal use of the VDS device driver (VIDAST) and the structure can be changed by Nokia without further notice. Therefore, the operator's application should not use the first record.

Note: File numbering starts from one instead of zero. Consequently, the data file number is the same as the record number. Thus, for example, the data related to the disk file CF1234.DAT can be found in the record number ‘1234’.

The TTSCOF records contain nine bytes and the record is divided into the following fields:

- file state information (1 byte)
- timestamp (7 bytes)
- storing status flag (1 byte)

The first byte contains the information on the state of the charging data file. Every time the state of a charging disk file changes, the state concerning this file is updated. The possible values of the file state are presented in Table 6: The states of charging disk files. The explanations of states are presented in .

Table 6 The states of charging disk files

State	Value
OPEN	00H
FULL	01H

Table 6 The states of charging disk files (Cont.)

State	Value
TRANSFERRED	02H
(WAITING)	03H (Obsolete)
(COMPRESSING)	04H (Obsolete)
UNUSEABLE	05H

The next seven bytes contain the timestamp showing the time when the charging data file is saved. The structure of the timestamp is presented in Table 7: The structure of the timestamp.

Table 7 The structure of the timestamp

Offset (byte)	Unit of measure	An example: 23rd October 1999 10:35:42
0	Seconds	42 BCD
1	Minutes	35 BCD
2	Hours	10 BCD
3	Days	23 BCD
4	Months	10 BCD
5	Years (last two digits)	99 BCD
6	Years (first two digits)	19 BCD

The last byte in the record contains information on the storing status of the charging disk file. The post-processing system, for example, can use the information to find out the location (WDU-0 or WDU-1) of the saved charging disk file and see if the charging disk file has been compressed. The values of the storing status flag are shown in Table 8: The meanings of values of storing status flag.

Table 8 The meanings of values of storing status flag

Offset (bit)	Explanation
0	This bit is set when the original data is successfully written on the WDU-0.

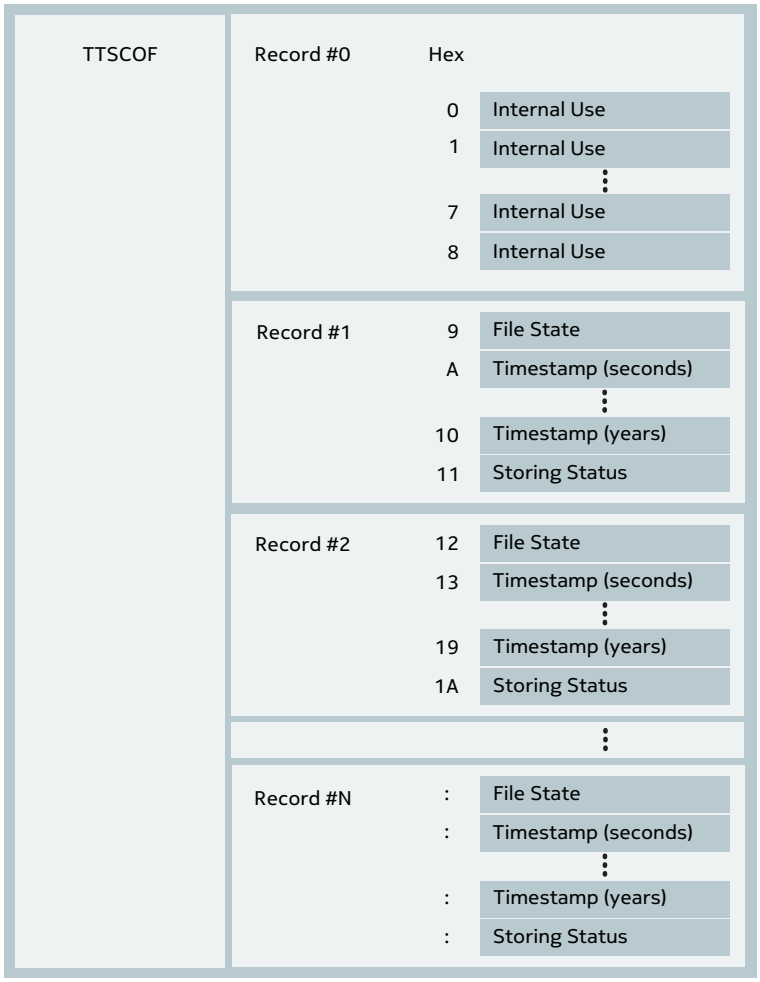
Table 8 The meanings of values of storing status flag (Cont.)

Offset (bit)	Explanation
1	This bit is set when the original data is successfully written on the WDU-1.
2	This bit is set when the compressed data is successfully written on the WDU-0.
3	This bit is set when the compressed data is successfully written on the WDU-1.
4	It is reserved for future use.
5	This bit is set when the back up of the file is successfully copied with the `IFC` MML command to the backup media.
6	This bit is set when the file is skipped due to incorrect state. The sequential order of files has been lost.
7	This bit is set when the file has been in FULL state before it has been set to OPEN. Untransferred data has been overwritten.

The default values of the bits are zeros '0'. The bit is changed to one when a charging data file has reached a certain storing status. For example, the value '00001111' indicates that both compressed and uncompressed charging disk files are stored to both disks (WDU-0 and WDU-1).

All records in the TTSCOF except the record number zero (as mentioned earlier) have the structure as shown above. The entire structure of the TTSCOF is shown in Figure 13: The structure of TTSCOF. The file is divided into records of nine bytes. The first nine bytes belong to the record number zero, the following nine bytes to record number one, and so on.

Figure 13 The structure of TTSCOF



An example of TTSCOF is presented in An example of the contents of a TTTCOF. The contents of the TTSCOF contain as many records as the number of the defined charging data files.

An example of a TTSCOF

```
1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16

0000: 09 00 FF FF FF FF FF FF FF 01 20 15 13 11 12 99
0010: 19 0F 01 16 28 13 11 12 99 19 0F 01 35 55 13 11
0020: 12 99 19 0F 01 11 23 14 11 12 99 19 0F 01 03 48
0030: 14 11 12 99 19 0F 01 09 17 15 11 12 99 19 0F 01
0040: 48 29 12 11 12 99 19 0F 01 16 58 15 11 12 99 19
0050: 0F 00 33 59 15 11 12 99 19 0F 02 56 14 11 11 12
0060: 98 19 0F 02 28 15 11 11 12 99 19 0F……
09 00 FF FF FF FF FF FF FF
```

Record 0:

For internal use.

```
01 35 55 13 11 12 09 20 0F
```


Record 3:

The charging data file ‘CF0003.DAT’ has been set to the 'FULL' state on 11th Dec 2009, 13:55:35. Both the compressed and decompressed data have been saved to both disks.

```
00 33 59 15 11 12 09 20 0F
```

Record 9:

The charging data file ‘CF0009.DAT’ is in the 'OPEN' state and was opened on 11th Dec 2009, 15:59:33. The file has been successfully initialised on both disks.

Others:

'CF0001.DAT’, ‘CF0002.DAT’, ‘CF0004.DAT’ - ‘CF0008.DAT’ are in 'FULL' state. ‘CF0010.DAT’ and ‘CF0011.DAT’ are in 'TRANSFERRED' state.

TTTCOF (VDS Device Data File Transfer Control File)

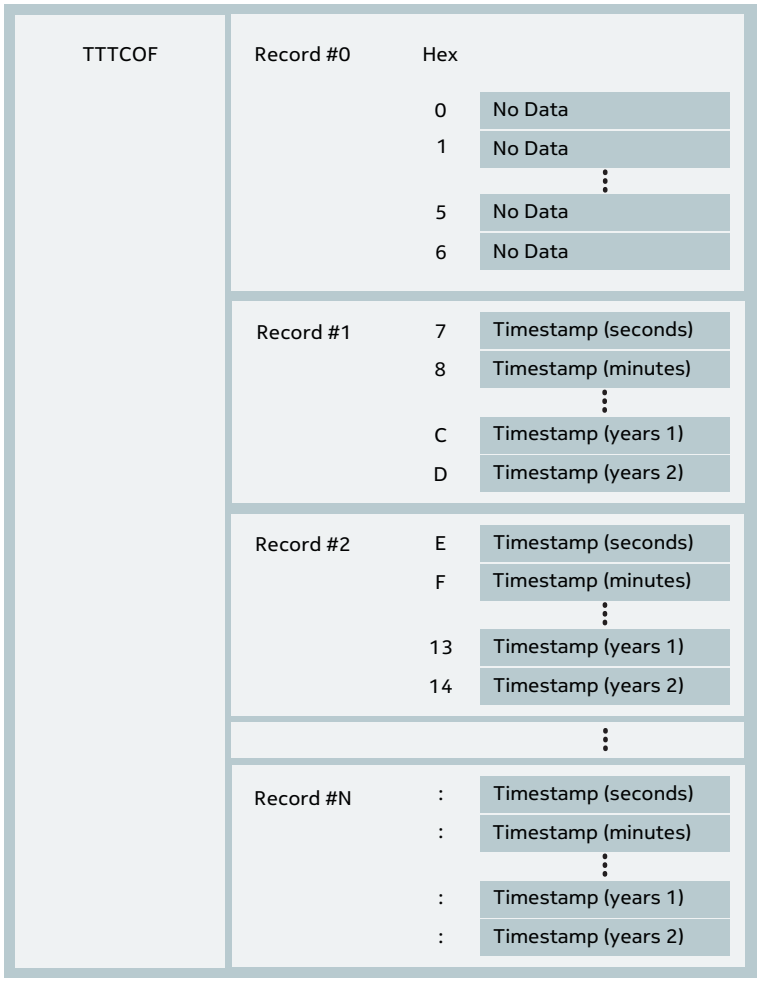
The TTTCOF controls the charging data transfer, and it is located in the hard disks of the CHU.

For each VDS device, there is one TTTCOF. The name of the TTSCOF is 'TTTCOFxx.IMG', where the *xx* is the number of the VDS device (see Table 9: VDS System).

The TTTCOF record contains seven bytes, and the record contains a timestamp with an identical structure with the timestamp of TTTCOF. The structure of the timestamp is presented in Table 7: The structure of the timestamp and it shows the time when the charging disk file has been transferred to the post-processing system. The TTTCOF is updated only by the post-processing system.

The entire structure of the TTTCOF is shown in Figure 14: The structure of TTTCOF. The file is divided into records of seven bytes. The first record (record number zero) of the TTTCOF contains no data.

Figure 14 The structure of TTTCOF



An example of the contents of a TTTCOF contains records of charging data files from ‘CF0001.DAT’ to ‘CF0011.DAT’. The TTTCOF contains as many records as the number of the defined charging data files.

An example of the contents of a TTTCOF

```
1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16

0000: 00 00 00 00 00 00 00 02 10 11 11 12 99 19 27 10
0010: 11 11 12 99 19 55 10 11 11 12 99 19 31 11 11 11
0020: 12 99 19 03 12 11 11 12 99 19 36 12 11 11 12 99
0030: 19 10 13 11 11 12 99 19 42 13 11 11 12 99 19 15
0040: 14 11 11 12 99 19 56 14 11 11 12 99 19 28 15 11
0050: 11 12 99 19……
00 00 00 00 00 00 00
```

Record 0:

Contains no data.

```
55 10 11 11 12 09 20
```

Record 3:

The charging data file ‘CF0003.DAT’ has been transferred to the post-processing system on 11th Dec 2009, 11:10:55.

```
15 14 11 11 12 09 20
```

Record 9:

The charging data file ‘CF0009.DAT’ has been transferred to the post-processing system on 11th Dec 2009, 11:14:15.

2.2.6.3 Configuration of the VDS system

The Man-Machine Interface (MMI) is used to display and change the configuration of the VDS system. The command group for this purpose is `IF`.

The following tasks, for example, can be done with the command group:

- changing the charging disk file states
- setting alarm limits and alarm mode for a disk
- defining the overwrite mode
- defining the size and amount of the files on disk
- defining time controls for changing disk files
- copying data files from disk to a removable media unit
- displaying control files and the file status map
- interrogating disk file information
- enabling or disabling the TTTCOF reading from hard disk
- defining the reading interval
- outputting data concerning the status of the hard disk and the parameter values
- controlling compression.

More information on the VDS System is presented in Table 9: VDS System.

Table 9 VDS System

Logical file	VDS-device and application	Amount of files	File size
TTLOFI	VDS-0 GSMCHA	10...9999	4kB...32MB
(HBLOFI)	VDS-1 GSMBIL	10...999	10kB... 1,9MB
TRACIG	VDS-2 ACFCIG	5... 25	96kB... 128kB
TRACSU	VDS-3	5... 25	96kB... 128kB

Table 9 VDS System (Cont.)

Logical file	VDS-device and application	Amount of files	File size
	ACFCSU		
TRACIR	VDS-4 ACFCIR	5... 25	384kB... 2MB
TRAZON	VDS-5 ACFZON	5... 25	384kB... 1MB
TRATOT	VDS-6 ACFTOT	5... 25	1kB... 8kB
MSSACCOUNT	VDS-7 GENE07	10... 999	4kB... 256kB
	VDS-8 GENE08	10...9999	4kB...32MB
	VDS-9 GENE09	10...9999	4kB...32MB
	VDS-10 GENE10	10...9999	4kB...32MB
	VDS-11 GENE11	10...9999	4kB...32MB
	VDS-12 GENE12	10...9999	4kB...32MB
	VDS-13 GENE13	10...9999	4kB...32MB
	VDS-14 GENE14	10...9999	4kB...32MB
	VDS-15 GENE15	10...9999	4kB...32MB

VDS devices has preset directories and file names. They are presented in Directories and file names used by VDS-devices.

Table 10 Directories and file names used by VDS-devices

Usage	Application name VDS Device ID	Directory and name of uncompressed file Directory and name of compressed file Directory and name of TTSCOF Directory and name of TTTCOF
Traditional file based CDR transfer	GSMCHA VDS-0	CFxxxx.DATCFxxxxx.DAT CFxxxx.ZCFxxxxx.Z TTSCOF00.IMG TTTCOF00.IMG
Hot Billing test files	GSMBIL VDS-1	GSMBIL/BFxxxx.DATGSMBIL/BFxxxxx.DAT GSMBIL/BFxxxx.ZGSMBIL/BFxxxxx.Z GSMBIL/TTSCOF01.IMG GSMBIL/TTTCOF01.IMG
Circuit Group accounting reports	ACFCIG VDS-2	ACCDIR/AGxxxx.DATACCDIR/AGxxxxx.DAT ACCDIR/AGxxxxx.Z ACCDIR/TTSCOF02.IMG ACCDIR/TTTCOF02.IMG
Circuit Sum accounting reports	ACFCSU VDS-3	ACCDIR/ASxxxxx.DAT ACCDIR/ASxxxxx.Z ACCDIR/TTSCOF03.IMG ACCDIR/TTTCOF03.IMG
Circuit accounting reports	ACFCIR VDS-4	ACCDIR/ACxxxxx.DAT ACCDIR/ACxxxxx.Z ACCDIR/TTSCOF04.IMG ACCDIR/TTTCOF04.IMG
Charging Zone accounting reports	ACFZON VDS-5	ACCDIR/AZxxxxx.DAT ACCDIR/AZxxxxx.Z ACCDIR/TTSCOF05.IMG ACCDIR/TTTCOF05.IMG
Total accounting reports	ACFTOT VDS-6	ACCDIR/ATxxxxx.DAT ACCDIR/ATxxxxx.Z ACCDIR/TTSCOF06.IMG ACCDIR/TTTCOF06.IMG

Table 10 Directories and file names used by VDS-devices (Cont.)

Usage	Application name VDS Device ID	Directory and name of uncompressed file Directory and name of compressed file Directory and name of TTSCOF Directory and name of TTTCOF
MSS accounting reports	GENE07 VDS-7	VIDAST/ORIG07/GExxxxx.DAT VIDAST/COMP07/GExxxxx.Z VIDAST/TTSCOF07.IMG VIDAST/TTTCOF07.IMG
CDR fall back saving of immediate CDR transfer for all subscribers	GENE08 VDS-8	VIDAST/ORIG08/GExxxxx.DAT VIDAST/COMP08/GExxxxx.Z VIDAST/TTSCOF08.IMG VIDAST/TTTCOF08.IMG
Reserved for future use	GENE09 VDS-9	VIDAST/ORIG09/GExxxxx.DAT VIDAST/COMP09/GExxxxx.Z VIDAST/TTSCOF09.IMG VIDAST/TTTCOF09.IMG
Storing charging data on removable media (UMS-0)	GENE10 VDS-10	VIDAST/ORIG10/GExxxxx.DAT VIDAST/COMP10/GExxxxx.Z VIDAST/TTSCOF10.IMG VIDAST/TTTCOF10.IMG
Storing charging data on removable media (UMS-1)	GENE11 VDS-11	VIDAST/ORIG11/GExxxxx.DAT VIDAST/COMP11/GExxxxx.Z VIDAST/TTSCOF11.IMG VIDAST/TTTCOF11.IMG
Reserved for future use	GENE12 VDS-12	VIDAST/ORIG12/GExxxxx.DAT VIDAST/COMP12/GExxxxx.Z VIDAST/TTSCOF12.IMG VIDAST/TTTCOF12.IMG
Reserved for future use	GENE13 VDS-13	VIDAST/ORIG13/GExxxxx.DAT VIDAST/COMP13/GExxxxx.Z VIDAST/TTSCOF13.IMG VIDAST/TTTCOF13.IMG

Table 10 Directories and file names used by VDS-devices (Cont.)

Usage	Application name VDS Device ID	Directory and name of uncompressed file Directory and name of compressed file Directory and name of TTSCOF Directory and name of TTTCOF
Reserved for future use	GENE14 VDS-14	VIDAST/ORIG14/GExxxxx.DAT VIDAST/COMP14/GExxxxx.Z VIDAST/TTSCOF14.IMG VIDAST/TTTCOF14.IMG
Reserved for future use	GENE15 VDS-15	VIDAST/ORIG15/GExxxxx.DAT VIDAST/COMP15/GExxxxx.Z VIDAST/TTSCOF15.IMG VIDAST/TTTCOF15.IMG

For more detailed information, see *Virtual Data Storing Device Handling*, IF command group. For more detailed information on alarms, see *Alarm, Descriptions*, and for more detailed information on data transfer from a VDS device to a post-processing system, see *Data File Transfer from VDS Device to Post-processing System*, interface specification.

2.2.6.4 VDS storing capacity

The storing capacity depends on the size of each storing task. The larger the size of each storing entity is, the better performance is achived. On the other hand, larger the size of each storing entity is, the longer the data is buffered on RAM.

The size of the hard disks must be large enough to hold data of all VDS devices. If the size or the amount of the disk files is changed, the operator must make sure that the hard disks are large enough. If the file size is changed, additional hard disk space might be needed temporarily. The space can be obtained by deleting some of the old files manually. The hard disks in the CHU must only be used for storing charging data.

If a new disk file cannot be created with the new size (for example, there is not enough free space on the hard disks), the file is marked 'UNUSEABLE'. After that the system changes the current file pointer to the next file that has already been created. All files with the state 'TRANSFERRED' which were detected as nonexistent, are marked 'UNUSEABLE'. If the size of the disk file is changed, the system recreates the file. Despite the recovery procedure mentioned above, it is the operator's responsibility to ensure that there is enough free space on the hard disks when the size and the amount of files are changed.

If the charging file closing interval or TTTCOF loading interval is set incorrectly, the payload decreases in the I/O System and in the file transfer. If TTSCOF and TTTCOF files are frequently handled in this situation, the amount of the real charging information that has been generated between the handlings of the control files decreases. In the worst case, the handling of the control files is performed so frequently that it affects the performance of the system. Therefore, it is important that the charging files become full during busy hours and are closed because of the storing time limits only during low traffic hours. This means that you should define the charging file closing interval and the TTTCOF reading interval values to be somewhat longer than the time period that is needed to get a full charging file during a busy hour. However, note that from the performance point of view, it is not sensible to decrease file size to get shorter timer intervals. The shorter the charging file is, the greater amount of files is needed to hold the same amount of data, which results in the increase of the size of control files, this in turn increases overhead and decreases the payload ratio.

In case there are several usages of the same I/O resources, the performance which can be offered to single service varies. It has to be ensured that in the CHU there is no extra users of I/O system, especially during busy hours. For this reason, it is not reasonable to store the HB CDRs to separate files. The copying of accounting counters to VDS devices should be timed so that the copying does not happen in busy hours. The copying of the charging files from the hard disks to removable media should be done only during the time of low traffic.

The performance of the VDS device varies for a number of reasons. For example, the model of the hard disk is a significant factor, not only for storing capacity in terms of bytes, but also in terms of speed. The location of files in the hard disk is also a significant factor, as the searches between the data files and the directory structure happen often.

In case small blocks, for which the search times are relatively slow, the VDS device might have to be optimised for that usage. In case all the recommendations are followed (hard disk models, charging block sizes, and so on), these optimizations should not be used. When these small block/slow search optimizations are used with the latest hard disk models and the recommended block sizes, the speed of the VDS device actually drops when compared to the default (not optimised) mode.

2.2.6.5 Alarms of VDS device

There are three types of alarms: general alarms which apply to all VDS applications, fill up alarms of general applications (for example, statistics), and fill up alarms of charging applications.

General alarms

- If the storing does not succeed on one disk, the alarm 'STORING OF THE DATA FAILED ON ONE DISK' (2650) is set.
- If the storing does not succeed on both disks, the alarm 'STORING OF THE DATA FAILED ON BOTH DISKS' (2651) is set.
- If the storing does not succeed because the hard disk(s) does not have enough free space or there is no free entries in directory structure, the alarm 'PHYSICAL DISK IS FULL' (2652) is set.

- If the handling of the control files (TTSCOF and TTTCOF) does not succeed, the alarm 'FAILURE IN THE HANDLING OF CONTROL FILES' (2653) is set.
- It is possible to define an alarm which is set when untransferred files have been overwritten. The VDS device driver sets the alarm 'OVERWRITING UNTRANSFERRED VDS-DEVICE DATA FILE' (2386) in case untransferred file (file with 'FULL' state) is opened for writing. (It is also possible to disable overwriting of untransferred files. It is recommended that overwriting is disabled in charging applications.)
- If the VDS device is used as a spare, it is possible to define the VDS device to generate an alarm whenever it has untransferred data in its disk buffer. In case the alarm is enabled, the VDS device alarms with alarm 'UNTRANSFERRED DATA IN VDS DEVICE' (3233) when data is stored to that device. The system automatically cancels the alarm when the post-processing system has transferred all files from the disk buffer.
- It is possible to define an age limit for the data. In case there is untransferred file in the disk whose age exceeds the threshold age, the VDS device sends the alarm 'TOO OLD FILES IN VDS DEVICE' (3234). The system automatically cancels the alarm when the post-processing system has transferred 'too old' files from the disk buffer.

Fill up alarms of general applications

They are used in statistical applications that do not require as high priority alarms as charging does.

- It is possible to define the first fill up ratio threshold of disk buffer usage. In case the fill up ratio exceeds the threshold limit, the VDS device driver sends the alarm 'FIRST ALARM LIMIT FOR UNAVAILABLE VDS-DEVICE DATA FILES REACHED' (2731). The system automatically cancels the alarm when the post-processing system has transferred files so that fill up ratio is lower than the first fill up ratio threshold cancel limit.
- It is possible to define second fill up ratio threshold of disk buffer usage. In case the fill up ratio exceeds the threshold limit, the VDS device driver sends the alarm 'SECOND ALARM LIMIT FOR UNAVAILABLE VDS-DEVICE DATA FILES REACHED' (2732). The system automatically cancels the alarm when the post-processing system has transferred files so that fill up ratio is lower than second fill up ratio threshold cancel limit.
- It is possible to define an alarm which is set when the last available file is opened. The VDS device driver sets the alarm 'OUT OF AVAILABLE VDS-DEVICE DATA FILES' (2733) in case the last available file is opened.

Fill up alarms of charging applications

These alarms are used in charging applications because the charging applications require alarms with a higher priority than, for example, statistical applications.

- It is possible to define the first fill up ratio threshold of disk buffer usage. In case the fill up ratio exceeds the threshold limit, the VDS device driver sends the alarm 'FIRST ALARM LIMIT FOR UNAVAILABLE VDS-DEVICE DATA FILES REACHED' (2549). The system automatically cancels the alarm when the post-processing system has transferred files so that the fill up ratio is lower than the first fill up ratio threshold cancel limit.

- It is possible to define a second fill up ratio threshold of disk buffer usage. In case the fill up ratio exceeds the threshold limit, the VDS device driver sends the alarm 'SECOND ALARM LIMIT FOR UNAVAILABLE VDS-DEVICE DATA FILES REACHED' (2550). The system automatically cancels the alarm when the post-processing system has transferred files so that the fill up ratio is lower than the second fill up ratio threshold cancel limit.
- It is possible to define an alarm which is set when the last available file is opened. The VDS device driver sets the alarm 'OUT OF AVAILABLE VDS-DEVICE DATA FILES' (2551) in case the last available file is opened.

Note: The calculation of the fill up ratio considers only the continuous area of the transferred files available and considers the other files unavailable. A fill up ratio is the proportion of the unavailable files out of the total amount of files. Thus, if the next file after the currently open file is in FULL state, the fill up ratio is 100% , and the status of all the rest of the files does not have any meaning to the calculation of the fill up ratio. It is possible to define whether the skipping of the untransferred files is enabled or disabled. However, the calculation of the fill up ratio does not change. Skipping of the untransferred files is a recovery action, which does not have any effect on the calculation of the fill up ratio.

2.2.7 Transfer of charging files from hard disk to a post-processing system

The transfer of the charging disk files and the charging control files from the hard disks of the CHU to a post-processing system (for example to the BC) is done with FTP over TCP/IP using LAN (Ethernet) interface of the CHU. Both IPv4 and IPv6 are supported.

The transfer procedure is described in the following section.

2.2.7.1 Charging file transfer using FTP

As described earlier, the charging control files are copied both from an Open MSS to the post-processing system and from the post-processing system to the Open MSS during the charging disk file transfer procedure. The concept *charging information transfer* is a TTSCOF and TTTCOF-based method for controlling the charging disk file transfer to the operator's post-processing system.

TTTCOF is a read-only file from the Open MSS point of view and the updating is performed by the post-processing system, whereas the TTSCOF is updated by the Open MSS and the post-processing system cannot update that file locally.

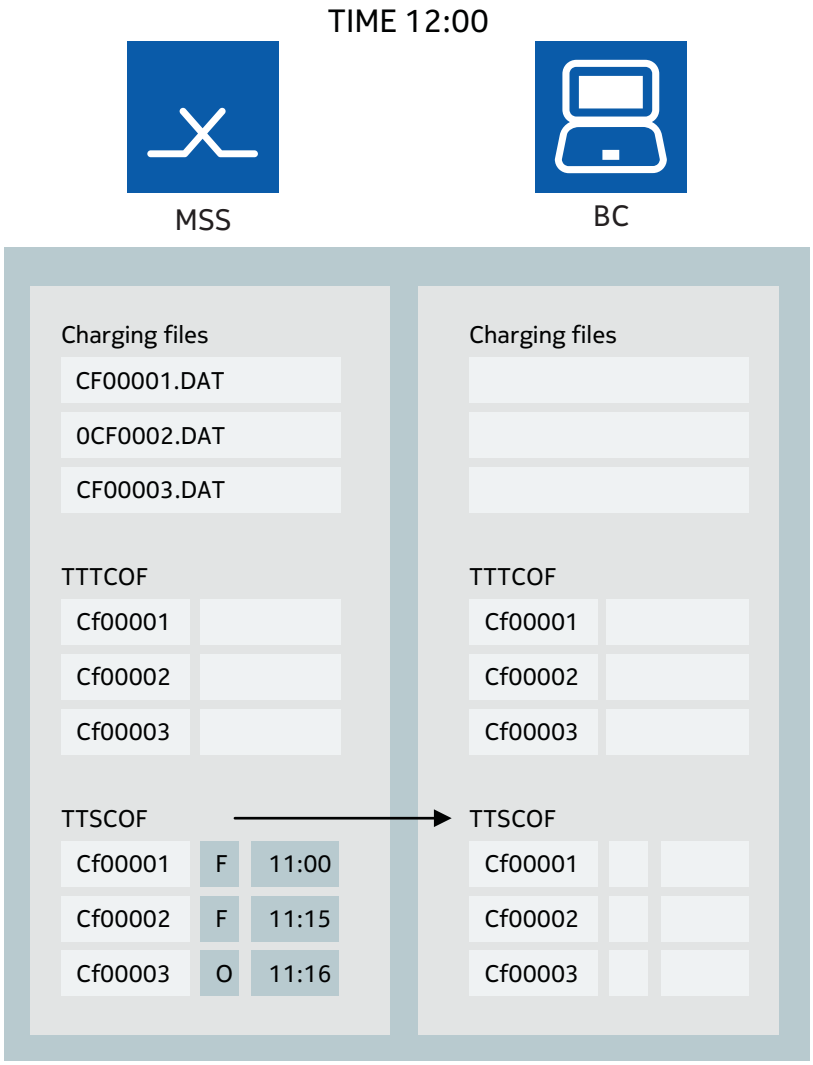
A charging information transfer to a post-processing system can be carried out with the FTP protocol.

The file transfer initiator is located in the post-processing system and the responder in the CHU. The post-processing system decides when the charging disk files are transferred from the CHU to the post-processing system.

The different stages of the transferring sequence from the CHU to the post-processing system (BC) are shown in the figures *Transferring of charging disk file from CHU to BC (Step 1)*, *(Step 2)*, *(Step 3)*, *(Step 4)*, *(Step 5)*, and *(Step 6)*.

1. The Billing Center (BC) activates an FTP file transfer to read the TTSCOF from the hard disk of the CHU.

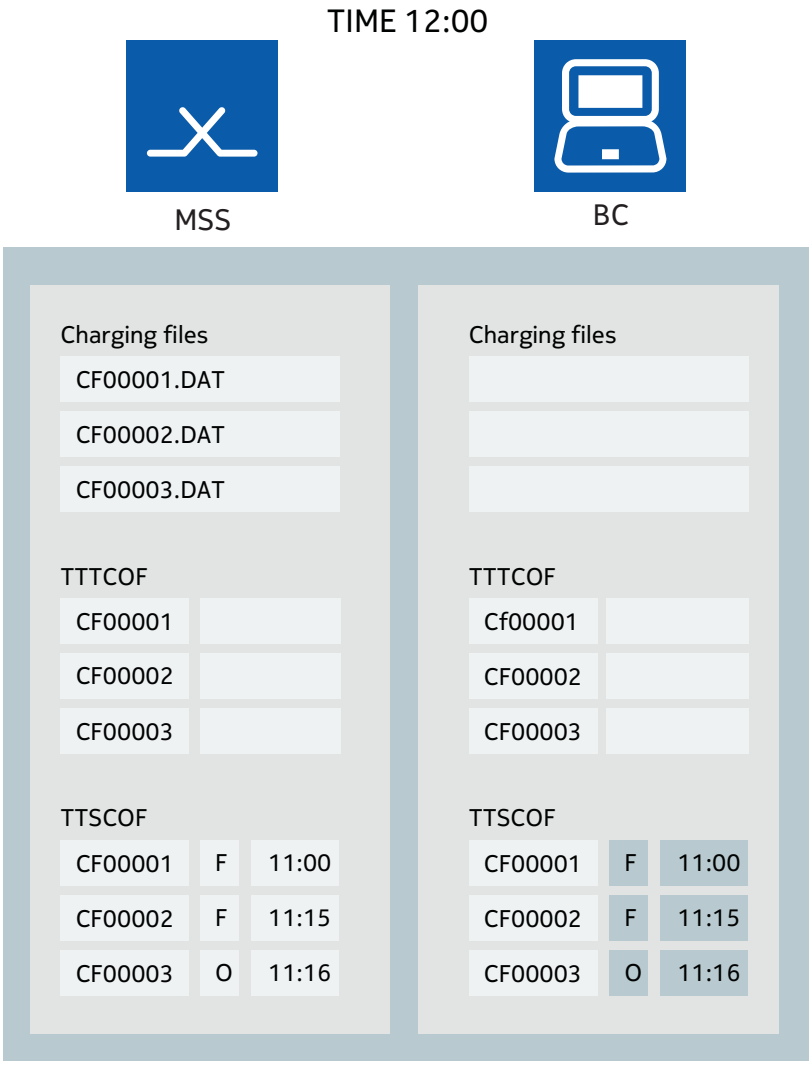
Figure 15 Transferring of charging disk file from CHU to BC (Step 1)



Note: The transfer mode must be binary.

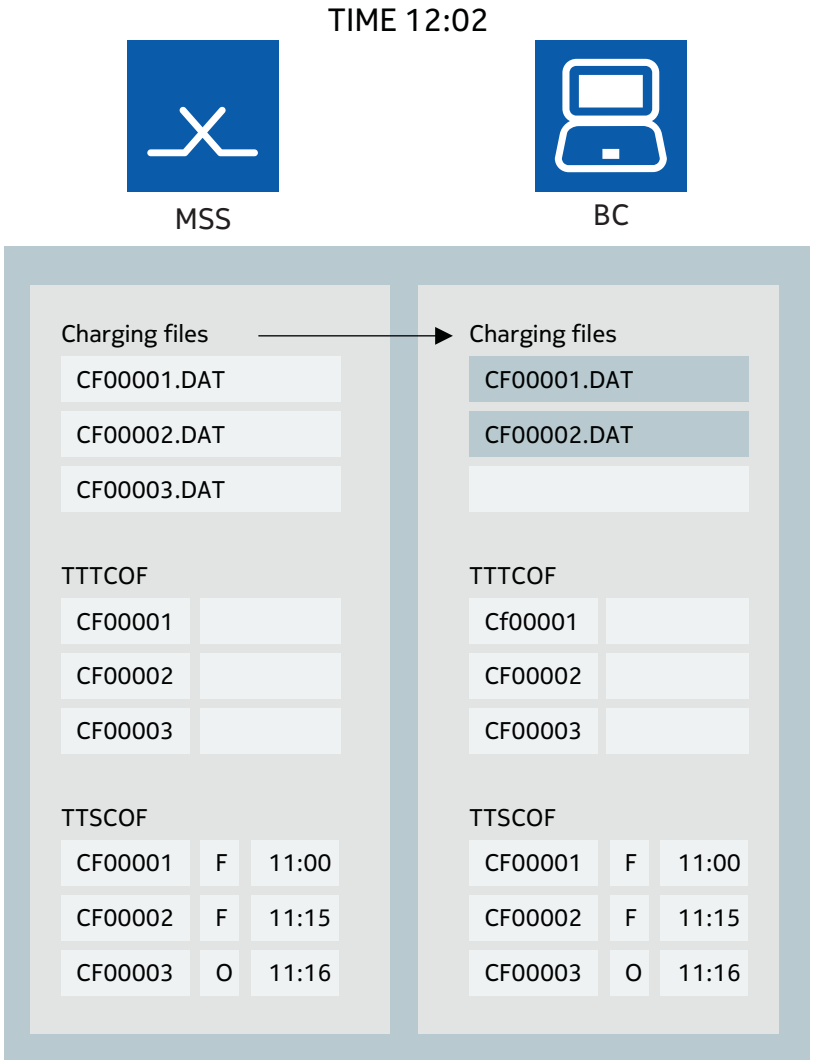
2. The BC examines the TTSCOF for the states and for the storing status information of the charging disk. The BC utilises the storing status flag to find out the location of the charging disk files with a 'FULL' state, and to find out whether the file is compressed or not.

Figure 16 Transferring of charging disk file from CHU to BC (Step 2)



3. The BC activates the FTP file transfer, and reads all the charging disk files with the FULL state from the CHU.
   The BC should also analyze the storing status flag.
   If the file is successfully stored only on one hard disk, the BC adds a hard disk ID (W0-/ or W1-/) at the beginning of the path.

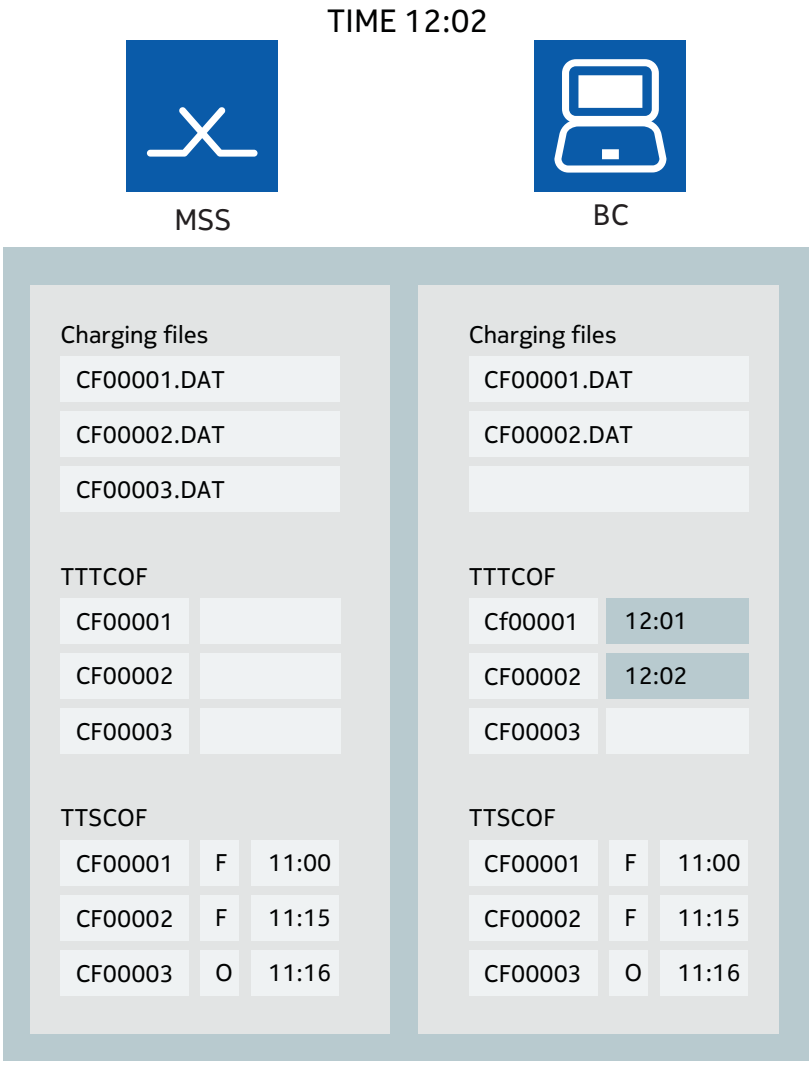
Figure 17 Transferring of charging disk file from CHU to BC (Step 3)



Note: The transfer mode must be binary.

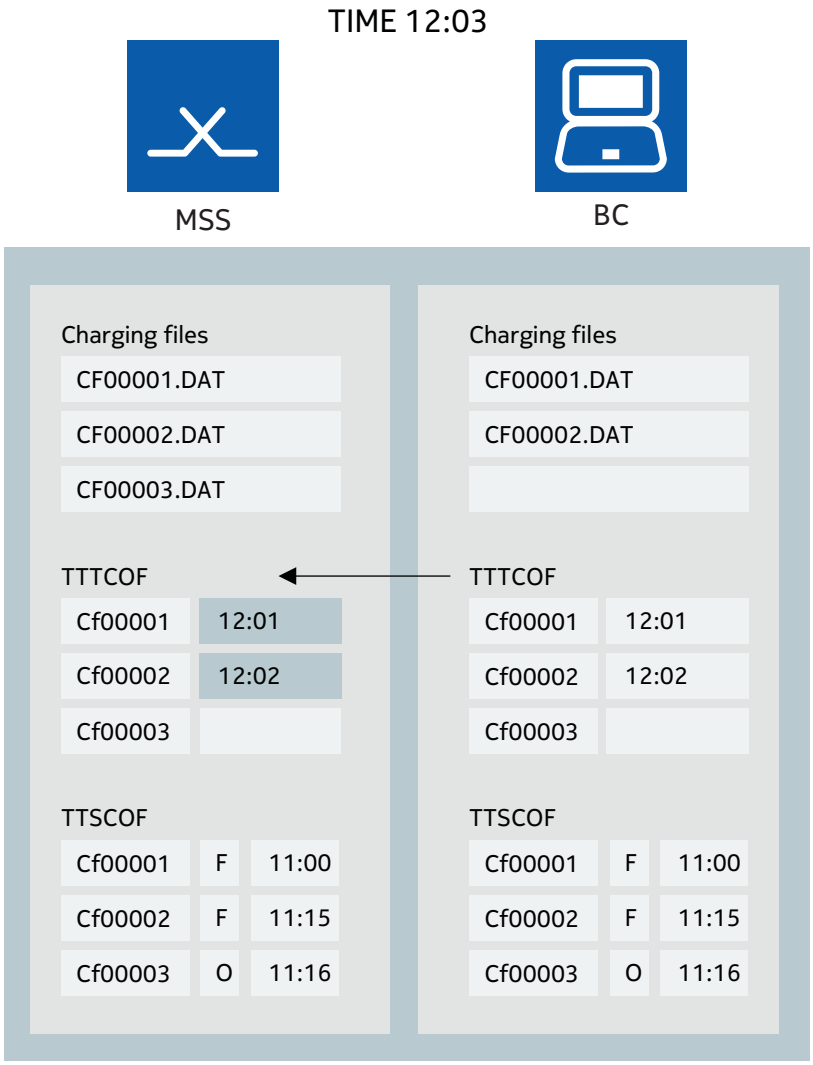
4. The BC updates the timestamps concerning the transferred disk files in the TTTCOF file. The timestamp must not be the same as in the TTSCOF file but at least one second more. This ensures that the file states will be changed from 'FULL' to 'TRANSFERRED' after the transfer.

Figure 18 Transferring of charging disk file from CHU to BC (Step 4)



5. The BC activates the FTP connection to write the TTTCOF file to the hard disk of CHU.

Figure 19 Transferring of charging disk file from CHU to BC (Step 5)



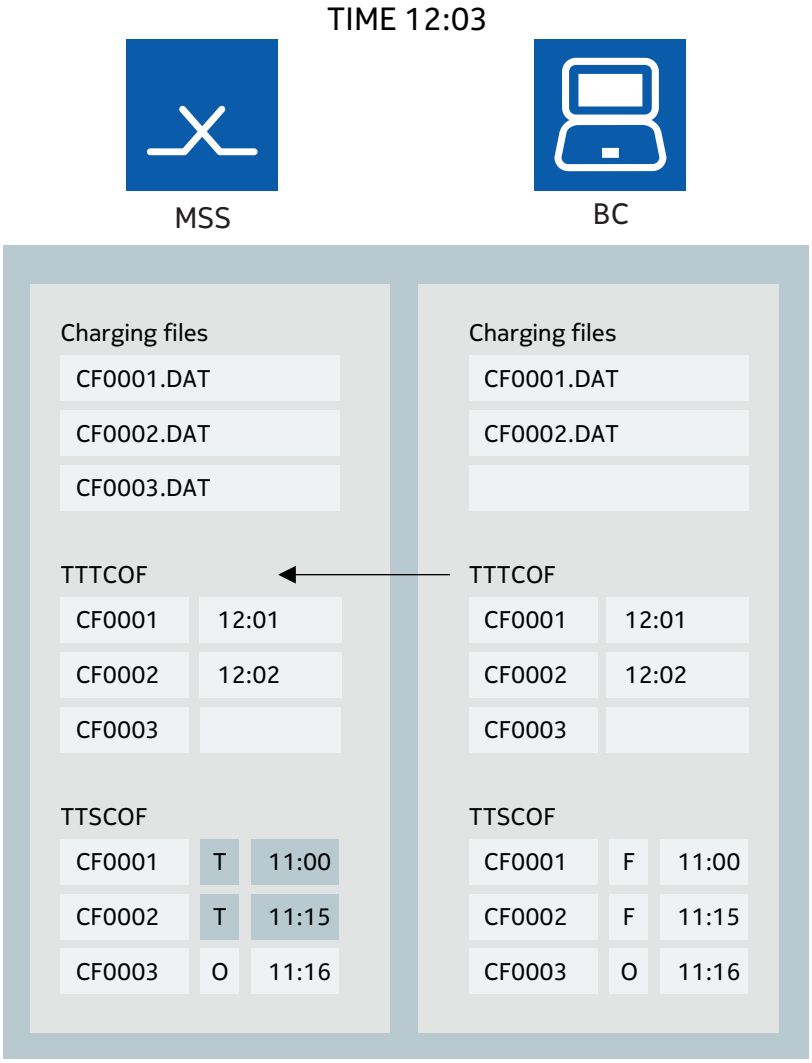
Note: The transfer mode must be binary.

Note: If the TTTCOF becomes corrupted during writing, it results in all charging files becoming 'FULL' which leads to a situation in which the charging data cannot be stored. This leads to a reduction of traffic.

6. The Open MSS compares the TTSCOF and the TTTCOF and updates the state information (TRANSFERRED) of all the files with a more recent timestamp in the TTTCOF.
   The TTSCOF is not immediately updated when the TTTCOF is received from the BC. The updating occurs when the file state is changed from a 'TRANSFERRED' to an 'OPEN' state. The alarms for the VDS ring file system fill-up are set and reset at the same time when the TTSCOF is updated. For more information on the different situations when the TTSCOF file is updated, see section .
   This is the end of one transfer session (transfer of TTSCOF and charging files from post-processing system to the Open MSS, and transfer of TTTCOF from post-processing system to the Open MSS). Before the system continues from step 1

(transfers the TTSCOF again), the system should be idle for a bit more than 5 minutes. This is the time it takes to be sure that the Open MSS has read the TTTCOF and updated the TTSCOF to the disk.

Figure 20 Transferring of charging disk file from CHU to BC (Step 6)



Note: The charging file marked 'OPEN' and files which become OPEN while the transfer is active, must not be transferred to the BC because it may prevent the storing of charging data on hard disks.

Transfer capacity

The charging records are transferred from the Open MSS to the post-processing system via LAN. The maximum number of concurrent file transfers by using the FTP are limited to 16.

Note: If several copies of charging records are needed, it is recommended to make the copies in the post-processing system. It is not reasonable to transfer the same charging records several times from the Open MSS. The design capacity figures of network elements assume that the charging data is transferred only once.

For more information, see *TCP/IP, Functional Description*.

2.2.7.2 FTP parameters

The FTP offers the data transfer services of a TCP/IP network. The MSS FTP contains only the server feature, which means that the post-processing system always acts as a client. Data transfer is always prompted by the post-processing system.

FTP parameters which must be used in connection with a Nokia FTP are presented in *TCP/IP Guide*.

2.2.7.3 CPU load of CHU while file transfer is active

The system is designed in a way that the file transfer (FTP) is performed as fast as possible, though with very low priority. This may cause high CPU load peaks in the CHU when the charging data is transferred through relatively fast connections. Though as the priority of transfer processes is set very low, all critical functions get enough run time. That is, the speed how fast the FTP transfer can be performed depends on the load caused by critical functions.

The system is not designed nor tested to work in conditions where all the idle processing time of the CHU is used for continuous TTSCOF polling. It is recommended that there is at least one minute of idle time between transfer sessions. (From post-processing system point of view, read TTSCOF, read 'CFxxxx.DAT' files, write TTTCOF, at least one minute delay, and again, read TTSCOF...)

2.2.8 Using removable media as a storing device

The storing of charging data to mass storage devices does not add any additional information to the charging data. The disk alternatives are MSDOS/FAT16 and MSDOS/FAT32. For more configuration, see *Charging Handling.*

2.2.8.1 Storing CDRs on Blu-ray disk

Charging data can be stored directly to the Blu-ray disk. For more information, see *Charging Handling.*

The following configurations can be used when the Blu-ray disks are used as storing devices of charging data:

- using it as a primary storing device
- using it parallel with hard disks
- using it as a back-up device if storing to hard disks fails

Transfer of charging records

The charging records on a Blu-ray disk cannot be transferred electronically to the post-processing system. The records are brought to the post-processing system as Blu-ray disks.

2.2.8.2 Storing CDRs on a USB mass storage device

Charging data can be stored directly on a USB mass storage device. For more information, see *Charging Handling.*

The following configurations can be used when a USB mass storage device is used as storing devices of charging data:

- using it as a primary storing device
- using it parallel with hard disks
- using it as a back-up device if storing to hard disks fails

Transfer of charging records

The charging records on a USB mass storage device cannot be transferred electronically to the post-processing system. The records are brought to the post-processing system as USB mass storage devices.

2.2.8.3 Copying CDRs from hard disk to Blu-ray disks

CDRs can be copied from a hard disk to a Blu-ray disk.

For more information on the use of optical disks, see *Charging Handling*, *Virtual Data Storing Device Handling*, IF command group, *Batch Copy Handling*, IP command group, *I/O File Backup*, IB command group, *I/O Device Working States*, IS command group and *I/O System Administration*.

2.2.8.4 Copying CDRs from hard disk to a USB mass storage device

CDRs can be copied from a hard disk to a USB mass storage device.

For more information on the use of USB storage devices, see *Charging Handling*, *Virtual Data Storing Device Handling*, IF command group, *Batch Copy Handling*, IP command group, *I/O File Backup*, IB command group, *I/O Device Working States*, IS command group and *I/O System Administration*.

2.2.9 Common storing and transfer problems

This section provides information about the most common problems with storing and transfer of charging data and about how those problems are solved.

2.2.9.1 CDR file gets corrupted during transfer

If a CDR file gets corrupted during transfer, the most common reason is that the file transfer has not been set as a binary transfer. For example, in an FTP transfer the transfer is done using the ASCII mode by default. In most systems the ASCII transfer causes that a byte 0x0D without 0x0A causes a line feed translation which appends 0x0A after every 0x0D (which does not already have preceding 0x0A). There can also be maximum line length defined in system which transfers these files, causing that 0x0A, 0x0D or any combination of them may be inserted to the file. The ASCII transfer may cause also other kinds of translations. It is important that the transfer mode is set to binary before the file transfer.

2.2.9.2 TTTCOF file gets corrupted during transfer

The most common reason for a TTTCOF file getting corrupted during transfer is that the previous transfer of TTTCOF has been interrupted for some reason and the FTP client in system which tries to write the TTTCOF to the MSS tries to continue the interrupted transfer. The MSS does not support continuing an interrupted transfer. An attempt to continue an interrupted transfer may result that only the end of the TTTCOF is written to the disks of the MSS (the beginning of the file is lost). The post-processing system must not try to continue interrupted transfers, instead, it should transfer the TTTCOF again from the beginning.

2.2.9.3 There is no open charging file

The system closes an open charging file in the following circumstances:

- when an active charging unit is restarted,
- when the active role of a unit is switched to spare unit (switch-over is performed)
- when the previous file was closed (filled up exactly to the defined size or by timer).

A new charging file is opened when the first charging block is stored on disk. In case of very low traffic, it may take quite long time before a charging file is opened. If *Feature 1491: Immediate CDR Transfer for All Subscribers* is active, all charging data should be transferred immediately and there should not be an open charging file when the system is operating normally.

It is normal that there could be situations when an open file is not present. The MSS sends an alarm if any charging data storing problem arises. If new full charging files do not appear regularly when the traditional CDR file transfer is used (or GTP' messages are not received regularly when *Feature 161: Hot Billing* or *Feature 1491: Immediate CDR Transfer for All Subscribers* is used), it increases the total system reliability if an alarm is set at the post-processing system.

For more information, see *Charging Handling*.

2.2.9.4 Corrupted FBS files

When *Feature 1491: Immediate CDR Transfer for All Subscribers* is activated, the charging data is transferred immediately to the post-processing system using the GTP' protocol. If immediate transfer does not work to any destination or the transfer speed is too slow, the Fall Back Saving (FBS) will be activated.

The FBS stores the GTP' packages via the CDRFBS logical file. The CDRFBS is connected to the VDS-8 which is configured in a way that charging data is stored on hard disks. Alternatively, the CDRFBS may be connected to the VDS-10 which has a CHG type spare connection to the VDS-11 (or vice versa). VDS-10 is configured to store data on a Blu-ray disk using the UMS-0 USB mass storage drive, and the VDS-11 is configured to store data on a Blu-ray disk using the UMS-1 USB mass storage drive.

If the logical file CDRFBS is incorrectly connected to the VDS-0 device of the charging unit (GSMCHA application), the VDS-0 device will corrupt the FBS files. The VDS-0 device of the charging unit (GSMCHA application) writes four byte long batch_sequence_numer to offset 22. This batch_sequence_number is written to every block stored to the VDS-0 device. If the FBS files are written using the VDS-0 device, some CDRs will be corrupted. Note that only the VDS-0 device of the charging unit (GSMCHA application) writes the batch_sequence_number inside the charging block.

2.2.9.5 Writing of TTTCOF fails on one disk (FTP error code 550)

If there is a single hard disk failure in the CHU, the TTTCOF transfer is actually successful though error code 550 is given. If the FTP error code 550 is given and an acknowledgement sending failure (transferring the TTTCOF to the MSS fails) occurs, it is not practical for the post-processing system to ignore the charging data files due to the acknowledgement sending failure (for example, for preventing multiple processing of the same charging data) but on the other hand continue pulling the same charging data files over and over again. If the charging files are ignored due to an acknowledgement sending failure, the files have to be transferred again after both disks are up and running. For more information, see section *Multiple transfers of the same charging data*.

2.2.9.6 Writing of TTTCOF to disk fails if TTTCOF file is locked on one disk

If the *TTTC0F* file is locked on one disk of the Charging Unit (CHU), uploading of this file is not successful for any of the disks. There is no automatic disk mirroring. This means that applications must handle problems related to disk failures. If disk writing does not succeed to both disks, it is considered unsuccessful. In the case of an unsuccessful writing, the files must be transferred again after both of the *TTTC0F* files are unlocked on both disks.

2.2.9.7 Multiple transfers of the same charging data

Multiple transfers of the same charging data can be prevented in the following way: When the TTSCOF is read, and the FULL files are found, the BC checks if the filling timestamp in the TTSCOF is older than the transferring timestamp in the own copy of the TTTCOF. If the filling timestamp is older than the transferring timestamp, the BC has

already transferred the file and the problem is that the MSS has not yet received or processed the TTTCOF file. The names of these charging files should be marked to an error log of the BC, so that these file numbers can be informed to the MSS operating personnel, so that it can set them manually as 'TRANSFERRED' if the problem cannot be solved, for example, within one day by retransmission of the TTTCOF.

2.3 Support protocol description

FTP	File Transfer Protocol FTP is a protocol in the TCP/IP protocol family that enables connection establishment and file transfer over the network between hosts.
GTP'	GPRS Tunnel Protocol Prime GTP' protocol has been designed to deliver the GPRS CDRs to the CGF(s) from the network elements or the functional entities generating charging records.
IPv4	Version 4 of the Internet protocol
IPv6	Version 6 of the Internet protocol
LAN	Local Area Network A local area network is a fast data transmission network covering a small area. A LAN can cover, for example, a building or a group of buildings.
TCP/IP	Transmission Control Protocol/Internet Protocol The TCP/IP is a protocol suite in the combination of different protocols at various layers. It is named after the names of its two main standard. TCP/IP defines the protocols with which computers can exchange data and use each other's services.

For more detailed information, see *OSI Guide* and *TCP/IP Guide*.

