Technical Document - Confidential

GSM Fax & Data Services

Message Sequence Charts

L2R

Document Number:	8411.202.01.106
Version:	0.8
Status:	Draft
Approval Authority:	
Creation Date:	1998-April-14
Last changed:	2015-Mar-08 by XGUTTEFE
File Name:	l2r.doc

Important Notice

Change History

Date	Changed by	Approved by	Version	Status	Notes
1998-Apr-14	Manfred Gutheins		0.1		1
1998-Apr-15	Manfred Gutheins		0.2		2
1998-Apr-27	Manfred Gutheins		0.3		3
1998-Jul-31	Manfred Gutheins		0.4		4
1998-Aug-20	Manfred Gutheins		0.5		5
1998-Oct-26	Manfred Gutheins		0.6		6
2001-Nov-21	Jacek Kwasnik		0.7		7
2003-May-15	XGUTTEFE		0.8	Draft	

Note s:

1. Initial version
2. Reset and Error Indication added
3. Description of L2R/RIP interface changed
4. Description of L2R SAP changed, BREAK handling added
5. Collisions at connection establishment added, Collision of RLP_CONN_IND and parameter negotiation added, RLP_RESET_RES contains CAK/NAK
6. RLP_RESET_RES contains no(!) ACK/NAK, Handover of UART added, L2R_ACTIVE_CNF and L2R_DEACTIVE_CNF added
7. Relay entity removed, DTI Interface introduced, MSCs reviewed and reworked

Table of Contents

List of Figures and Tables

List of References

[ISO 9000:2000] International Organization for Standardization. Quality management systems - Fundamentals and vocabulary. December 2000

1.1 References

[1] Rec. T.4 Standardisation of group 3 facsimile apparatus for document transmission;
(CCITT-T.4, 1984)

[2] ITU-T Recommendation T.30; Series T: Terminal equipments and protocols for telematic services;
Procedures for document facsimile transmission in the general switched
telephone network;
(ITU-T.30, 1996)

[3] ITU-T Recommendation T.31; Terminals for telematic services;
Asynchronous facsimile DCE control - service class 1
(ITU-T.31, 1995)

[4] ITU-T Recommendation T.32; Terminals for telematic services;
Asynchronous facsimile DCE control - service class 2
(ITU-T.32, 1995)

[5] Rec. T.35; Terminal equipment and protocols for telematic services;
Procedures for the allocation of CCITT definde codes for non-standard facilities;
(CCITT-T.35, 1991)

[6] ITU-T Recommendation V.25 ter; Series V: data communication over the telephone network;
Interfaces and voiceband modems; Serial asynchronous automatic dialling and control
(ITU-T V.25 ter, 1997)

[7] Rec. V.42 bis Data compression procedures for data circuit terminating equipment (DCE) using error correction procedures;
(CCITT-V.42 bis, 1990)

[8] Rec. V.110 (Blue book, Vol. VIII, Fascicle VIII.1) Support of data terminal equipments (DTEs) with V-series type interfaces by an integrated services digital network (ISDN);
(CCITT-V.110, 1988)

[9] European digital cellular telecommunications system (Phase 2);
GSM Public Land Mobile Network (PLMN) connection types;
(GSM 3.10, September 1994, version 4.3.1)

[10] European digital cellular telecommunications system (Phase 2);
Technical realisation of facsimile group 3 transparent;
(GSM 3.45, September 1995, version 4.5.0)

[11] Digital cellular telecommunications system (Phase 2);
Mobile radio interface layer 3 specification;
(GSM 4.08, November 1996, version 4.17.0)

[12] European digital cellular telecommunications system (Phase 2);
Rate adaptation on the Mobile Station - Base Station System (MS - BSS) Interface;
(GSM 4.21, May 1995, version 4.6.0)

[13] European digital cellular telecommunications system (Phase 2);
Radio Link Protocol (RLP) for data and telematic services on the Mobile Station - Base Station System (MS - BSS) interface and the Base Station System - Mobile-service Switching Centre (BSS - MSC) interface
(GSM 4.22, September 1994, version 4.3.0)

[14] European digital cellular telecommunications system (Phase 2);
Radio Link Protocol (RLP) for data and telematic services on the Mobile Station - Base Station System (MS - BSS) interface and the Base Station System - Mobile-service Switching Centre (BSS - MSC) interface
(Amendment prA1 for GSM 4.22, version 4.3.0)
(GSM 4.22, March 1995, version 4.4.0)

[15] European digital cellular telecommunications system (Phase 2);
General on Terminal Adaptation Functions (TAF) for Mobile Stations (MS);
(GSM 7.01, December 1995, version 4.10.0)

[16] European digital cellular telecommunications system (Phase 2);
Terminal Adaptation Functions (TAF) for services using asynchronous bearer capabilities;
(GSM 7.02, September 1994, version 4.5.1)

[17] European digital cellular telecommunications system (Phase 2);
Terminal Adaptation Functions (TAF) for services using synchronous bearer capabilities;
(GSM 7.03, September 1994, version 4.5.1)

[18] Digital cellular telecommunications system (Phase 2);
Use of Data Terminal Equipment - Data Circuit terminating Equipment (DTE - DCE) interface for Short Message Service (SMS) and Cell Broadcast Services (CBS);
(GSM 7.05, November 1996, version 4.8.0)

[19] Digital cellular telecommunications system (Phase 2);
AT command set for GSM Mobile Equipment (ME)
(GSM 7.07, May 1996, version 4.1.0)

[20] Digital cellular telecommunication system (Phase 2);
Mobile Station (MS) conformance specification;
Part 1: Conformance specification
(GSM 11.10-1, November 1996, version 4.17.0)

[21] Digital cellular telecommunications system (Phase 2);
Mobile Station (MS) conformance specification;
Part 2: Protocol Implementation Conformance Statement (PICS) proforma specification
(GSM 11.10-2, May 1996, version 4.15.0)

[22] Digital cellular telecommunications system (Phase 2);
Mobile Station (MS) conformance specification;
Part 3: Layer 3 (L3) Abstract Test Suite (ATS)
(GSM 11.10-3, November 1996, version 4.17.0)

[23] Proposal for Rate Adaptation implemented on a DSP;
(C. Bianconi, Texas Instruments, January 1998, version 1.0)

[24] MCU-DSP Interfaces for Data Applications;
Specification S844
(C. Bianconi, Texas Instruments, March 1998, version 0.1)

[25] Users Guide
6147.300.96.100; Condat GmbH

[26] Service Access Point RA
8411.100.98.100; Condat GmbH

[27] Service Access Point RLP
8411.101.98.100; Condat GmbH

[28] Service Access Point L2R
8411.102.98.100; Condat GmbH

[29] Service Access Point FAD
8411.103.98.100; Condat GmbH

[30] Service Access Point T30
8411.104.98.100; Condat GmbH

[31] Service Access Point ACI
8411.105.98.100; Condat GmbH

[32] Message Sequence Charts RLP
8411.201.98.100; Condat GmbH

[33] Message Sequence Charts L2R
8411.202.98.100; Condat GmbH

[34] Message Sequence Charts FAD
8411.203.98.100; Condat GmbH

[35] Message Sequence Charts T30
8411.204.98.100; Condat GmbH

[36] Message Sequence Charts ACI
8411.205.98.100; Condat GmbH

[37] Proposal for Fax & Data Integration; March 1998
8411.300.98.100; Condat GmbH

[38] Test Specification RLP
8411.401.98.100; Condat GmbH

[39] Test Specification L2R
8411.402.98.100; Condat GmbH

[40] Test Specification FAD
8411.403.98.100; Condat GmbH
[41] Test Specification T30
8411.404.98.100; Condat GmbH
[42] Test Specification ACI
8411.405.98.100; Condat GmbH
[43] SDL Specification RLP
8411.501.98.100; Condat GmbH
[44] SDL Specification L2R
8411.502.98.100; Condat GmbH
[45] SDL Specification FAD
8411.503.98.100; Condat GmbH
[46] SDL Specification T30
8411.504.98.100; Condat GmbH
[47] SDL Specification ACI
8411.505.98.100; Condat GmbH
[48] Technical Documentation RLP
8411.701.98.100; Condat GmbH
[49] Technical Documentation L2R
8411.702.98.100; Condat GmbH
[50] Technical Documentation FAD
8411.703.98.100; Condat GmbH
[51] Technical Documentation T30
8411.704.98.100; Condat GmbH
[52] Technical Documentation ACI
8411.705.98.100; Condat GmbH

1.2 Abbreviations

ACI	AT Command Interpreter
AGCH	Access Grant Channel
AT	Attention sequence "AT" to indicate valid commands of the ACI
BCCH	Broadcast Control Channel
BCS	Binary Coded Signals
BS	Base Station
BSIC	Base Station Identification Code
C/R	Command/Response
C1	Path Loss Criterion
C2	Reselection Criterion
CBCH	Cell Broadcast Channel
CBQ	Cell Bar Qualify
CC	Call Control
CCCH	Common Control Channel
CCD	Condat Coder Decoder
CKSN	Ciphering Key Sequence Number
CRC	Cyclic Redundancy Check
DCCH	Dedicated Control Channel
DISC	Disconnect Frame
DL	Data Link Layer
DM	Disconnected Mode Frame
DTX	Discontinuous Transmission
EA	Extension Bit Address Field
EL	Extension Bit Length Field
EMMI	Electrical Man Machine Interface
EOL	End Of Line
F	Final Bit
F&D	Fax and Data Protocol Stack
FACCH	Fast Associated Control Channel
FHO	Forced Handover
GP	Guard Period
GSM	Global System for Mobile Communication
HDLC	High level Data Link Control
HISR	High level Interrupt Service Routine
HPLMN	Home Public Land Mobile Network
I	Information Frame
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
ITU	International Telecommunication Union
IWF	Interworking Function
Kc	Authentication Key
L	Length Indicator
LAI	Location Area Information
LISR	Low level Interrupt Service Routine
LPD	Link Protocol Discriminator
M	More Data Bit
MCC	Mobile Country Code
MM	Mobility Management
MMI	Man Machine Interface
MNC	Mobile Network Code

MS	Mobile Station
MSG	Message phase in the GSM 3.45 protocol
N(R)	Receive Number
N(S)	Send Number
NCC	National Colour Code
NECI	New Establishment Causes included
OTD	Observed Time Difference
P	Poll Bit
P/F	Poll/Final Bit
PCH	Paging Channel
PCO	Point of Control and Observation
PDU	Protocol Description Unit
PL	Physical Layer
PLMN	Public Land Mobile Network
RACH	Random Access Channel
REJ	Reject Frame
RNR	Receive Not Ready Frame
RR	Radio Resource Management
RR	Receive Ready Frame
RTD	Real Time Difference
RTOS	Real Time Operating System
SABM	Set Asynchronous Balanced Mode
SACCH	Slow Associated Control Channel
SAP	Service Access Point
SAPI	Service Access Point Identifier
SDCCH	Slow Dedicated Control Channel
SIM	Subscriber Identity Module
SMS	Short Message Service
SMSCB	Short Message Service Cell Broadcast
SS	Supplementary Services
T.4	CCITT Standardisation for Document coding of Group 3 Facsimile Apparatus
TAP	Test Application Program
TCH	Traffic Channel
TCH/F	Traffic Channel Full Rate
TCH/H	Traffic Channel Half Rate
TDMA	Time Division Multiple Access
TE	Terminal Equipment - e. g. a PC
TMSI	Temporary Mobile Subscriber Identity
UA	Unnumbered Acknowledgement Frame
UI	Unnumbered Information Frame
V(A)	Acknowledgement State Variable
V(R)	Receive State Variable
V(S)	Send State Variable
VPLMN	Visiting Public Land Mobile Network

1.3 Terms

Entity:	Program which executes the functions of a layer
Message:	A message is a data unit which is transferred between the entities of the same layer (peer-to-peer) of the mobile and infrastructure side. Message is used as a synonym to protocol data unit (PDU). A message may contain several information elements.
Primitive:	A primitive is a data unit which is transferred between layers on one component (mobile station or infrastructure). The primitive has an operation code which identifies the primitive and its parameters.
Service Access Point:	A Service Access Point is a data interface between two layers on one component (mobile station or infrastructure).

2 Overview

The Protocol Stacks are used to define the functionality of the GSM protocols for interfaces. The GSM specifications are normative when used to describe the functionality of interfaces, but the stacks and the subdivision of protocol layers does not imply or restrict any implementation.

The protocol stack for fax and data transmission consists of several entities. Each entity has one ore more service access points, over which the entity provides a service for the upper entity. The entity, which is described in this document, is coloured grey in the following figure :

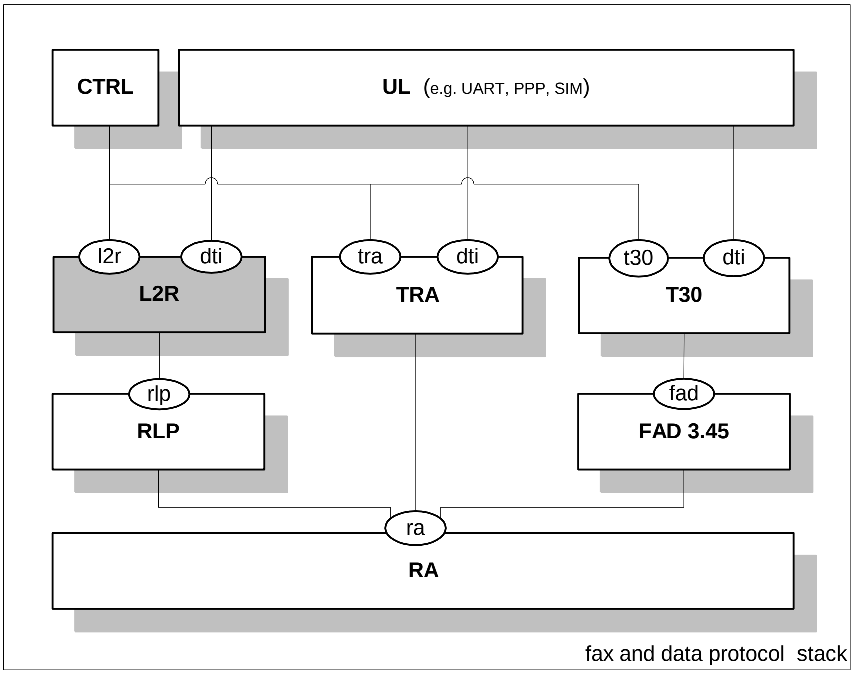

Figure 2-1: Architecture of the fax and data protocol stack

The information units passed via the SAPs are called primitives and consists of an operation code and several parameters. See the Users Guide for details.

The entities of the fax and data protocol stack are:

2.1 RA - Rate Adaptation

This entity performs an adaptation between an asynchronous or synchronous data stream with several bit rates on to the fixed bit rate used at the TCH. This is performed by the rate adaptation functions RA1' and RA0 described in GSM 04.21.

2.2 RLP - Radio Link Protocol

This entity provides a Layer 2 protocol for asynchronous reliable data transfer as specified in GSM 04.22. It includes error correction, sequence numbers and a mechanism for repeating corrupted and lost messages.

2.3 L2R - Layer 2 Relay Functionality

The L2R provides relay functions in order to adapt the character-oriented data received from the TE via UART to the bit-oriented RLP protocol.

2.4 FAD 03.45 - Fax Adaptation Protocol

The fax adaptation protocol, as specified in GSM 03.45, provides synchronisation with the BCS and MSG modems of the peer entity. It uses byte repetition in conjunction with a voting algorithm to handle corruption on the TCH data stream. The non-transparent fax protocol in accordance with GSM 03.46 is not part of this implementation.

The fax adapter enables T.30 to send BCS at 300 BPS and T.4 MSG in 2400, 4800, 7200 and 9600 BPS.

2.5 T.30 - Fax Protocol Entity

The protocol uses binary coded signals packed in HDLC frames to set up and release a connection in the message phase of the FAX transmission. This entity is specified in the ITU-T.30. The main tasks of this unit are:

- Building the HDLC frames with CRC.
- Performing bit stuffing/de-stuffing.
- Executing a sequence of 5 phases: 1.) set up, 2.) pre-message procedures, 3.) transmission/reception, 4.) post message procedures, 5.) waiting for call release.

2.6 ACI - AT Command Interpreter

The ACI is specified in GSM 07.07. It is responsible for call establishment via the GSM voice protocol stack and terminal adaptation for asynchronous transparent character-oriented data transmission. The ACI is able to receive AT commands and send the replies over the UART driver to a remote PC. This makes it possible to control the voice and data protocol stack from a remote application running on a PC. The ACI also provides a unique interface for an internal MMI in the MS.

2.7 USART - Universal Synchronous Asynchronous Receiver Transmitter Driver

The USART is a hardware component that facilitates a connection between the mobile station and terminal equipment (e.g. a PC). This interface uses some of the circuits described in V.24.

The data exchange provided by this unit is serial and asynchronous (synchronous communication is not in the scope of this document). A driver that uses interrupts to manage a circular buffer for the sending and receiving direction is necessary in order to use this component in the F&D. The driver has to be able to perform flow control.

2.8 CTRL - Controller entity

ACI entity fulfills the function of the controller. It activates the other entities (e.g. L2R, TRA, T30, SIM) and connects them to the upper layer. It also can disconnect them and take the control over them any time.

3 Introduction

The L2R (layer 2 relay) entity is included in the Fax & Data Services in order to support non transparent asynchronous transfer of data. It uses the services offered by the RLP entity. The protocol between the mobile and the Interworking Function (IWF) is the L2RCOP (L2R character oriented protocol). In the current implementation there is no protocol between the mobile and the TE. L2R can be extended however, to support a protocol (e.g. LAPB) on the TE interface. In any case it is the responsibility of the L2R entity to translate the L2RCOP to the protocol on the TE side or to the character stream, if no protocol is used. Synchronous transfer and the bit oriented protocol L2RBOP are not supported by the current implementation. The L2R entity is controlled by the ACI entity.

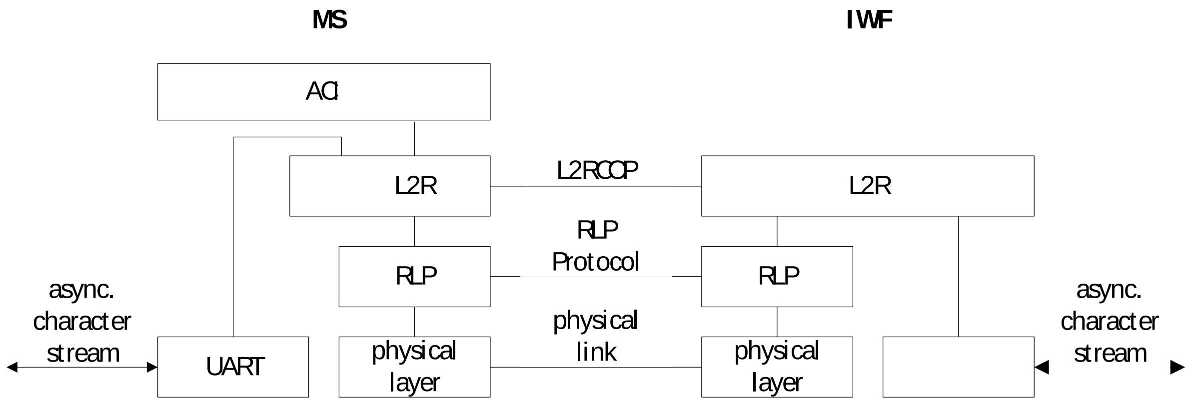

Figure 3-1: Function of the L2R entity

The main functions of the L2R entity are:

- Support of the L2RCOP.
- Buffering of data in both directions.
- Flow control.
- Transfer of status bits.
- Transfer of BREAK condition.

ETSI specifies a general structure of the L2R entity consisting of three sub-entities as shown in Figure 3-2.

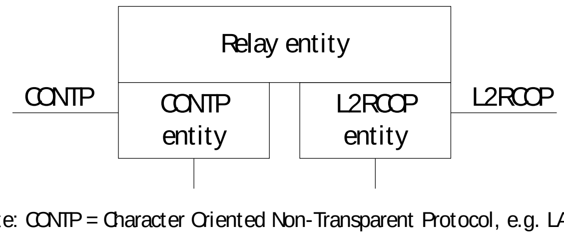

Figure 3-2: L2R sub-entities

In such a L2R structure the relay entity transfers data and status information between the CONTP entity and the L2RCOP entity. There is no direct communication between the CONTP entity and the L2RCOP entity.

The L2RCOP entity is responsible for handling the L2RCOP. It uses the services of the RLP entity.

The G23 implementation of the L2R functionality uses no relay entity and no CONTP entity. It is only an implementation of the L2RCOP entity. The communication with external and internal applications is realized over the DTI interface as shown in Figure 3-3 and Figure 3-4.

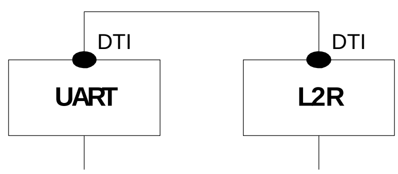

Figure 3-3: External Application: Relaying via DTI - SAP

Usually in the case of an external application there is no layer 2 protocol between the mobile and the TE. Therefore there is no CONTP entity and the L2R entity communicates directly with the UART driver over the DTI interface.

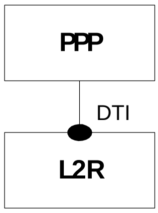

Figure 3-4: Internal Application: No relaying

In the case of an internal application there is no need for a relay functionality or a CONTP functionality. Actually for L2R there is no difference between internal and external applications, because L2R is always communicating via DTI in the same way.

3.1 The L2RCOP

Information is transferred between L2Rs in fixed length of **n** octet Protocol Data Units (PDUs). This corresponds to the RLP fixed length information field. The octets within the L2RCOP-PDU are numbered 0 to n-1, octet 0 is transmitted first. The bits within the octets are numbered from 1 to 8, bit 1 is transmitted first.

0	SA	SB	X	0	0	0	1	1	first status octet, 3 information octets
1	1	1	0	0	0	1	1	1	"G" (IA5, odd parity)
2	1	1	0	1	0	0	1	1	"S"
3	1	1	0	0	1	1	0	1	"M"
4	SA	SB	X	1	1	1	1	1	last status octet, rest of PDU empty
n									
Bit:	8	7	6	5	4	3	2	1	

Figure 3-5: Example of a L2RCOP PDU

Each octet contains a status octet, a character or fill

Octet 0 always contains a status octet

Status octets contain 3 status bits and 5 address bits

The 3 status bits correspond to SA, SB and X in CCITT Recommendation V.110 [8]. The SA, SB and X bits use bit positions 8, 7, 6 in the status octets. When one or more of the status bits changes the current state of all three bits is transmitted.

Character octets are coded in the following way:

- The first bit of the character received/transmitted corresponds to bit position 1 in the octet. The second bit to bit 2, ... and the seventh bit to bit 7.
- 7 bit characters are padded with a 0 in bit position 8. Received parity (if used) is inserted in bit position 8, if parity is not used bit 8 is set to 0.
- Any start/stop bits are removed by the L2R.

Characters are inserted into L2RCOP-PDUs in order of transmission in octets 1 to n-1.

The address field in the status octets indicates the position of the next status octet within the L2RCOP-PDU. This indicates the number of characters between status octets. Thus if two status octets are inserted into L2RCOP-PDU at offsets l and m the address value will be defined by m-l-1. Address bit 2^0 corresponds to bit 1 in the status octets. Address bit 2^1 to bit 2 etc.

Status octets are inserted in the character stream whenever a status change needs to be transmitted.

Only address values 1 to n-2 (currently expected to be 23) in the address field of status octets are used for addressing purposes. The codes and their meaning are shown in Table 3-1

Address field value	Meaning
1 – 23	Address (=number of characters between status octets)
24 – 27	Reserved
28	Destructive BREAK acknowledge, remainder of L2RCOP-PDU empty.
29	Destructive BREAK signal, remainder of L2RCOP-PDU empty.
30	Last status change, remainder of L2RCOP-PDU full of characters.
31	Last status change, remainder of L2RCOP-PDU empty.

Table 3-1: Address field values in status octets

When it is necessary to insert a status octet into the character stream when no status change has occurred, e.g. to indicate that the reminder of a L2RCOP-PDU is empty or to indicate a BREAK signal, the current status is repeated.

An example of a L2RCOP PDU is shown in Figure 3-5

3.2 Functions of the L2RCOP entity

The main function of the L2RCOP entity is to support the L2RCOP. This means providing and accepting L2R PDUs and the data contained in it including the status information.

3.2.1 Data transfer.

The L2R entity assembles and disassembles L2R PDUs. Data characters are assembled into L2R PDUs until either:

- The PDU is full
- A condition arises, where even incomplete L2R PDUs are to be transferred (see below).

L2R PDUs are transferred to the peer L2R entity using the data transfer services of the radio link.

The data transfer to and from the RLP entity uses handshaking by primitives. Transfer of data from one entity to the other is only possible after the receiving entity has indicated that it can accept data. The primitives containing data (RLP_DATA_REQ and RLP_DATA_IND) can hold a fixed number of L2R PDUs. This number is set at startup by the ACI. In the following figures it is shown, how primitives are exchanged between the L2R entity end the RLP entity.

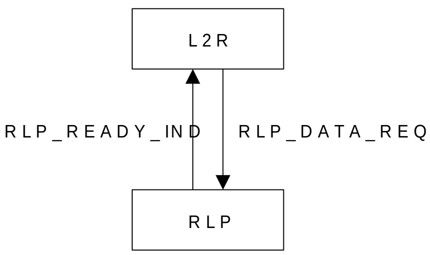

Figure 3-6: Handshaking for uplink data transfer

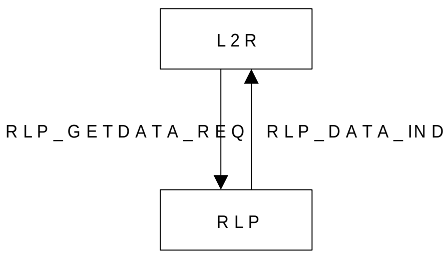

Figure 3-7: Handshaking for downlink data transfer

Figure 3-6 shows the **uplink** direction. First the RLP entity sends a RLP_READY_IND to the L2R entity. Hereby RLP indicates, that it is ready to accept more data from L2R. After receiving the RLP_READY_IND the L2R entity is allowed to send one RLP_DATA_REQ. RLP may sent the next RLP_READY_IND only after receiving the RLP_DATA_REQ.

Figure 3-7 shows the **downlink** direction. Here L2R signals with a RLP_GETDATA_REQ, that it is ready to accept data, and RLP transfers data with the RLP_DATA_IND.

The right to send a primitive is like a token, which is passed alternately from one entity to the other.

In case there is not enough data available, the sending entity does not generate a data primitive immediately. It starts a timer instead and waits for more data. If the sending entity can collect enough data, while the timer is still running, a data primitive is generated and the timer stopped. Otherwise the timer expires and all the data which is available at this time, is transferred. As a result of this each data primitive can contain one incomplete L2R PDU. In order not to degrade the performance, the timeout value must be set big enough, so that under normal operating circumstances, when the data source provides data at the GSM transmission speed, the timer does not expire.

3.2.2 Buffering of data in both directions

3.2.2.1 TX Buffers

Data received from the UART driver are buffered, so that if the mobile is unable to transfer the data over the radio path then it is not lost. The buffer is capable of holding 16-32 kbits of data. When the buffer is filled to a certain threshold, the TE is flow controlled as per 3.2.3.6, unless Character Orientated Protocol with No Flow Control is being used (see 3.2.3.7).

3.2.2.2 RX Buffers

Data for transfer to the TE are buffered, so that if the TE is unable to accept data then the data transferred from the mobile is not lost. The buffer size is 16-32 kbits of data. When the buffer is filled to a certain threshold, the L2R sends the "flow control active" indication, unless Character Orientated Protocol with No Flow Control is being used.

3.2.3 Flow control

A flow control active condition can take place under a number of circumstances:

- End to end flow control (TE to TE matter)
- Back pressure (buffer filling)
- Receive not ready (RLP condition)

3.2.3.1 End to end flow control

A TE may wish to send a flow control active condition to another TE.

Provisions exists in the L2R entity to transfer a flow control active condition (sent by its associated TE) to the other L2R entity as soon as possible. This mechanism in the L2R entities allows such a flow control condition to be put ahead of any queuing which exists in the L2R entities.

This mechanism prevents the possibly undesirable effect of congestion of the data in buffers.

The L2R entity, receiving a flow control active condition from its associated TE, stops sending data to that associated TE immediately.

3.2.3.2 Back pressure

The L2R and RLP entities have buffers which may become full to a predetermined threshold for a number of reasons, e.g. severe radio fading, failure or slowness of TE to react smoothly to the end to end flow control, certain RNR conditions. When this predetermined threshold is reached, a **flow control active** condition is sent to the associated TE. This TE is then prevented from sending any further data. The **flow control inactive** condition is being sent to the associated TE when the L2R or RLP entities have indicated the sufficient free capacity condition of their buffers.

3.2.3.3 Receive not ready (RNR)

No special handling for the RNR condition of RLP is implemented. Usually there is no need for the RLP of the base station to send a RNR command/response. RLP should always be able to pass the received data to the L2R entity, where the data are buffered. If the data can not be passed on further from the L2R buffer, either the remote end should activate an end to end flow control or L2R should send a flow control active signal, because its receive buffer is filling up. In this case the local L2R would stop transmitting so that RLP would never signal a RNR condition.

If in spite of this the base station RLP is sending a RNR command/response, the mobile RLP does not request any more PDUs from the L2R entity. Hence the L2R transmit buffer fills up until the threshold is reached („back pressure“). Following that L2R activates flow control to the TE locally.

3.2.3.4 Conditions requiring signalling of flow control towards the network

The L2R function will send immediately a "flow control active" indication under following conditions:

- If local flow control is initiated by the TE (see section 3.2.3.6). On receipt of this flow control indication forwarding of the data from receive buffer towards the TE is halted.
- If the receive buffer from the radio side reaches a preset threshold (Back pressure). If even more data are received after the flow control active indication has been issued and finally the L2R RX buffer becomes full, L2R does not collect any more data from RLP. If then in turn the RLP receive buffer fills up, RLP will signal RNR (receive not ready) to the peer RLP, so that no more data are transferred to the mobile.

Upon removal of both conditions the L2R will send a "flow control inactive" indication.

3.2.3.5 Conditions Requiring Flow Control towards TE

The L2R functions will immediately activate local flow control (see section 3.2.3.6) under the following circumstances:

- The L2R receives a "flow control active" indication.
- The transmit buffer reaches a pre-set threshold (Back pressure).

If both conditions are removed then local flow control is deactivated again.

3.2.3.6 Local Flow Control

For the data being sent towards the network the local flow control is accomplished in two ways depending on flow control (x-bit) support capability of the entity that is currently connected to L2R via the DTI interface:

- An **Entity supporting x-bit** stops sending data after receiving the x = ON condition
- An **Entity ignoring x-bit** continues sending data. **W**hen the L2R buffer fills up full, no DTI_READY_IND massage will be sent by L2R to the upper layer until there is enough space for data.

3.2.3.7 Character oriented protocol with no flow control

If ACI selects character oriented protocol with no flow control then no flow control is used i.e. the X-bit is not set and XON/XOFF are passed through as data.

3.2.4 Transfer of status bits

Interface status information is transferred between the L2Rs via the status octets in L2R PDUs. The meaning of the bits is:

X: Flow control,
SA: General ready signal,
SB: Data are valid.

Status changes are inserted in the L2R PDU at the position corresponding to the position in the character stream that the interface status change occurred. After a RLP-reset a L2R PDU with the current status octet must be sent.

The flow control condition however is not queued in the buffer, but immediately passed on, so that it can overtake all data in the buffers.

Filtering of channel control information as provided in the transparent mode is not required in the non transparent mode. Here the RLP protocol ensures the reliable transfer of the status bits.

NOTE:

For the status bits the ON condition means readiness. The ON condition is transferred in the status bits as a binary zero, whereas the OFF condition is transferred as a binary one. **In case of flow control:** if the **X** bit is **set to 1**, then **flow control is active** („not ready" - it means **NO DATA FLOW ALLOWED !**).

3.2.5 Establishment of the L2R connection

For the data connection of the mobile station with another end device to establish, following several consecutive steps are needed:

- RLP connection has to be established (as a result of SABM/UA commands exchange) first.
- L2R exchanges the empty frames with IWF containing the actual status of both sides.
- During delay period that comes next the IWF establishes the connection with another end device.
- IWF sends an empty frame to MS with the SB status bit set to "ON".
- L2R entity of the MS sends the L2R_CONNECT_CNF to ACI.
- ACI connects L2R with the upper layer over the DTI interface.
- The Data transfer can start now.

3.2.6 Transfer of BREAK condition

The BREAK condition is recognised by the mobile and passed immediately to the IWF. The mobile generates a BREAK condition to the TE on receipt of a BREAK indication from the IWF. The BREAK condition is handled only logically in the L2R entity. The transfer of BREAK conditions between L2Rs takes place via the status octets with appropriate coding of the address field.

The L2R PDU contains the mandatory status octet coded as the destructive BREAK.

Upon the receipt of the BREAK signal, the L2R will destroy any existing data in front of the BREAK signal in the same direction, and all the buffered data in the other direction. The L2R will then pass the BREAK signal immediately on.

The termination of a BREAK condition is indicated by sending an L2RCOP-PDU containing characters.

4 Protocol

4.1 Activation and deactivation

4.1.1 Activation of L2R

```
 CTRL                     L2R                      RLP
   |                        |                        |
   |    L2R_ACTIVATE_REQ    |                        |
   *======================>*                         |
   |                     (L2R 1)                     |
   |                        |    RLP_ATTACH_REQ      |
   |                        *======================>*
   |                        |                    (RLP 1)
   |                        |                        |
```

(L2R 1)
Contoller initialises and activates L2R.

(RLP 1)
RLP is initialised and switched into the ADM and attached mode.

4.1.2 Activation of L2R, invalid parameters

```
 CTRL                     L2R                      RLP
   |                        |                        |
   |    L2R_ACTIVATE_REQ    |                        |
   *======================>*                         |
   |                     (L2R 1)                     |
   |                        |                        |
   |    L2R_ACTIVATE_CNF    |                        |
   |          (NAK)         |                        |
   *<======================*                         |
 (CTRL 1)                   |                        |
   |                        |                        |
```

(L2R 1)
Controller initialises and activates L2R.

(CTRL 1)
The parameter values are invalid. Therefore L2R sends a L2R_ACTIVATE_CNF with NAK (non acknowledgement) to Controller.

4.1.3 Deactivation of L2R

```
 CTRL                     L2R                      RLP
   |                       |                        |
   |    L2R_DEACTIVATE_REQ  |                        |
   *======================>*                        |
   |                    (L2R 1)                     |
   |                       |     RLP_DETACH_REQ     |
   |                       *======================>*
   |                       |                     (RLP 1)
   |                       |     RLP_DETACH_CNF     |
   |                       *<======================*
   |                    (L2R 2)                     |
   |    L2R_DEACTIVATE_CNF  |                        |
   *<======================*                        |
(CTRL 1)                   |                        |
   |                       |                        |
```

(L2R 1)
Controller deactivates L2R.

(RLP 1)
L2R in turn deactivates RLP with a RLP_DETACH_REQ. No acknowledgement is required for this primitive, since detaching is always possible without fail.

(L2R 2)
RLP acknowledges the RLP_DETACH_REQ by a RLP_DETACH_CNF.

(CTRL 1)
L2R in turn acknowledges the previous L2R_DACTIVATE_REQ by a L2R_DACTIVATE_CNF.

4.2 Parameter negotiation

4.2.1 Reception of negotiated Parameters

```
 CTRL                     L2R                      RLP
   |                       |                        |
   |                       |       RLP_XID_IND      |
   |                       *<======================*
   |                    (L2R 1)                     |
   |       L2R_XID_IND      |                        |
   *<======================*                        |
(CTRL 1)                   |                        |
   |                       |                        |
```

(L2R 1)
L2R receives the negotiated parameters.

(CTRL 1)
The negotiated Parameters are passed to Controller, where they may be made available for the user.

4.2.2 Reception of negotiated Parameters, pending RLP_ATTACH_REQ

```
 CTRL                      L2R                      RLP
   |                        |                        |
   |                        |      RLP_XID_IND       |
   |                        *<======================*
   |                     (L2R 1)                     |
   |       L2R_XID_IND      |                        |
   *<======================*                         |
(CTRL 1)                    |                        |
   |    L2R_ACTIVATE_CNF    |                        |
   |          (ACK)         |                        |
   *<======================*                         |
(CTRL 2)                    |                        |
   |                        |                        |
```

(L2R 1)
L2R receives the negotiated parameters.

(CTRL 1)
The negotiated Parameters are passed to Controller, where they may be made available for the user.

(CTRL 2)
The RLP_XID_IND is also an acknowledgement for a previous RLP_ATTACH_REQ. Therefore L2R in turn acknowledges the L2R_ACTIVATE_REQ with a L2R_ACTIVATE_CNF. Controller should wait for this reply before connecting with a L2R_CONNECT_REQ. A timeout is not required, since a timeout is implemented in RLP already, which generates an error indication, when no response of the base station is received.

4.3 Mobile originated connect request

4.3.1 Initiation

```
 CTRL                      L2R                      RLP
   |                        |                        |
   |    L2R_CONNECT_REQ     |                        |
   *======================>*                         |
   |                     (L2R 1)                     |
   |                        |    RLP_CONNECT_REQ     |
   |                        *======================>*
   |                        |                    (RLP 1)
   |                        |                        |
```

(L2R 1)
Controller requests L2R to connect to its peer entity.

(RLP 1)
RLP in turn is requested to connect to its peer.

4.3.2 Positive response from the base station

In the following MSC the flow control primitives RLP_GETDATA_REQ and RLP_READY_IND are not shown for the sake of clarity.

```
 CTRL                     L2R                     RLP
   |                       |                       |
   |                       |    RLP_CONNECT_CNF    |
   |                       |         (ACK)         |
   |                       *<======================*
   |                    (L2R 1)                    |
   |                       |      RLP_DATA_IND     |
   |                       |        (empty)        |
   |                       *<======================*
   |                    (L2R 2)                    |
   |                       |     RLP_DATA_REQ      |
   |                       |        (empty)        |
   |                       *======================>*
   |                       |                    (RLP 1)
DELAY                      |                       |
   |                       |      RLP_DATA_IND     |
   |                       |     (empty, SB=0)     |
   |                       *<======================*
   |                    (L2R 3)                    |
   |     L2R_CONNECT_CNF   |                       |
   |          (ACK)        |                       |
   *<======================*                       |
(CTRL 1)                   |                       |
   |                       |                       |
```

(L2R 1)
RLP confirms the previously received RLP_CONNECT_REQ to L2R.

(L2R 2)
RLP sends a primitive with an empty frame from the IWF with the actual status.

(RLP 1)
L2R sends a primitive with an empty frame and the actual status to the IWF in turn.

During **delay** time IWF is establishing the connection with the second TE (end device).

(L2R 3)
When this connection is established the IWF sends an empty frame with the status bit SB = 0 (“ON”).

(CTRL 1)
L2R in turn confirms the connection establishment to Controller. Now the connection is established and L2R signals its readiness to receive data (see 4.3.5 Connection established4.3.4).

4.3.3 Negative response from the base station

```
 CTRL                     L2R                     RLP
   |                       |                       |
   |                       |    RLP_CONNECT_CNF    |
   |                       |         (NAK)         |
   |                       *<======================*
   |                    (L2R 1)                    |
   |     L2R_CONNECT_CNF   |                       |
   |          (NAK)        |                       |
   *<======================*                       |
(CTRL 1)                   |                       |
   |                       |                       |
```

(L2R 1)
RLP rejects the RLP_CONNECT_REQ to L2R.

(CTRL 1)
L2R in turn rejects the connection establishment to Controller.

4.3.4 Collision of both sides connection establishment

In the following MSC the flow control primitives RLP_GETDATA_REQ and RLP_READY_IND are not shown for the sake of clarity.

```
CTRL                     L2R                      RLP
  |                        |                        |
  |                        |    RLP_CONNECT_IND     |
  |                        *<======================*
  |                      (L2R 1)                    |
  |                        |    RLP_CONNECT_RES     |
  |                        |         (ACK)          |
  |                        *======================>*
  |                        |                     (RLP 1)
  |                        |      RLP_DATA_IND      |
  |                        |        (empty)         |
  |                        *<======================*
  |                      (L2R 2)                    |
  |                        |      RLP_DATA_REQ      |
  |                        |       (empty)          |
  |                        *======================>*
  |                        |                     (RLP 2)
DELAY                      |                        |
  |                        |      RLP_DATA_IND      |
  |                        |     (empty, SB=0)      |
  |                        *<======================*
  |                      (L2R 3)                    |
  |     L2R_CONNECT_CNF    |                        |
  |          (ACK)         |                        |
  *<======================*                        |
(CTRL 1)                   |                        |
```

(L2R 1)
RLP indicates to L2R, that the base station wants to connect.

(RLP 1)
L2R acknowledges the RLP_CONNECT_IND by a RLP_CONNECT_RES. This is always a positive acknowledgement. Now the connection is established and L2R signals its readiness to receive data (see 4.3.5 Connection established).

(L2R 2)
RLP sends a primitive with an empty frame and the actual status.

(RLP 2)
L2R sends a primitive with an empty frame and the actual status in turn.

During **delay** time IWF is establishing the connection with the second TE (end device).

(L2R 3)
When this connection is established the IWF sends an empty frame with the status bit SB = 0 (“ON”).

(CTRL 1)
L2R confirms the connection establishment to Controller.

4.3.5 Connection established

```
  UL                      L2R                     RLP
   |                       |                       |
   |                       |    RLP_GETDATA_REQ    |
   |                       *======================>*
   |                       |                    (RLP 1)
   |      DTI_READY_IND    |                       |
   *<======================*                       |
 (UL 1)                    |                       |
   |                       |                       |
```

(RLP 1)
L2R signals to RLP, that it is ready to receive data from RLP.

(UL 1)
L2R signals to the upper layer, that it is ready to receive data from the upper layer.

4.4 Mobile terminated connect request

4.4.1 Reception of connect indication after parameters have been negotiated

In the following MSC the flow control primitives RLP_GETDATA_REQ and RLP_READY_IND are not shown for the sake of clarity.

```
 CTRL                     L2R                     RLP
   |                       |                       |
   |                       |    RLP_CONNECT_IND    |
   |                       *<======================*
   |                    (L2R 1)                    |
   |                       |    RLP_CONNECT_RES    |
   |                       |         (ACK)         |
   |                       *======================>*
   |                       |                    (RLP 1)
   |                       |      RLP_DATA_IND     |
   |                       |        (empty)        |
   |                       *<======================*
   |                    (L2R 2)                    |
   |                       |     RLP_DATA_REQ      |
   |                       |        (empty)        |
   |                       *======================>*
   |                       |                    (RLP 2)
DELAY                      |                       |
   |                       |      RLP_DATA_IND     |
   |                       |     (empty, SB=0)     |
   |                       *<======================*
   |                    (L2R 3)                    |
   |    L2R_CONNECT_IND    |                       |
   *<======================*                       |
(CTRL 1)                   |                       |
```

(L2R 1)
RLP indicates to L2R, that the base station wants to connect.

(RLP 1)
L2R acknowledges the RLP_CONNECT_IND by a RLP_CONNECT_RES. This is always a positive acknowledgement. Now the connection is established and L2R signals its readiness to receive data (see 4.3.5 Connection established).

(L2R 2)
RLP sends a primitive with an empty frame and the actual status.

(RLP 2)
L2R sends a primitive with an empty frame and the actual status in turn.

During **delay** time IWF is establishing the connection with the second TE (end device).

(L2R 3)
When this connection is established the IWF sends an empty frame with the status bit SB = 0 (“ON”).

(CTRL 1)
L2R in turn indicates to Controller that the base station wants to connect.

4.4.2 Reception of connect indication before parameter negotiation has been finished

In the following MSC the flow control primitives RLP_GETDATA_REQ and RLP_READY_IND are not shown for the sake of clarity.

```
 CTRL                        L2R                        RLP
   |                          |                          |
   |                          |     RLP_CONNECT_IND      |
   |                          *<=======================*
   |                       (L2R 1)                       |
   |                          |       RLP_XID_IND        |
   |                          *<=======================*
   |                       (L2R 2)                       |
   |       L2R_XID_IND        |                          |
   *<=======================*                          |
(CTRL 1)                      |                          |
   |    L2R_ACTIVATE_CNF      |                          |
   *<=======================*                          |
(CTRL 2)                      |                          |
   |                          |     RLP_CONNECT_RES      |
   |                          |          (ACK)           |
   |                          *=======================>*
   |                          |                       (RLP 1)
   |                          |       RLP_DATA_IND       |
   |                          |         (empty)          |
   |                          *<=======================*
   |                       (L2R 3)                       |
   |                          |      RLP_DATA_REQ        |
   |                          |         (empty)          |
   |                          *=======================>*
   |                          |                       (RLP 2)
DELAY                         |                          |
   |                          |       RLP_DATA_IND       |
   |                          |      (empty, SB=0)       |
   |                          *<=======================*
   |                       (L2R 4)                       |
   |     L2R_CONNECT_IND      |                          |
   *<=======================*                          |
(CTRL 3)                      |                          |
```

(L2R 1)
RLP indicates to L2R, that the base station wants to connect. L2R has been activated by a L2R_ACTIVATE_REQ, but the XID parameter negotiation has not been completed yet. Therefore L2R delays the RLP_CONNECT_IND, until it receives a RLP_XID_IND.

(L2R 2)
L2R receives the negotiated parameters.

(CTRL 1)
The negotiated Parameters are passed to Controller, where they may be made available for the user.

(CTRL 2)
L2R acknowledges the previously received L2R_ACTIVATE_REQ with a L2R_ACTIVATE_CNF.

(RLP 1)
L2R acknowledges the RLP_CONNECT_IND by a RLP_CONNECT_RES. This is always a positive acknowledgement. Now the connection is established and L2R signals its readiness to receive data (see 4.3.5 Connection established).

(L2R 3)
RLP sends a primitive with an empty frame and the actual status.

(RLP 2)
L2R sends a primitive with an empty frame and the actual status in turn.

During **delay** time IWF is establishing the connection with the second TE (end device).

(L2R 4)
When this connection is established the IWF sends an empty frame with the status bit SB = 0 (“ON”).

(CTRL 3)
L2R in turn indicates to Controller that the base station wants to connect.

4.5 Mobile originated disconnection

4.5.1 Initiation from connected state

```
 CTRL                        L2R                        RLP
   |                          |                          |
   |                          |                          |
   |      L2R_DISC_REQ        |                          |
   *=======================>*                          |
   |                       (L2R 1)                       |
   |                          |       RLP_DISC_REQ       |
   |                          *=======================>*
   |                          |                       (RLP 1)
   |                          |                          |
```

(L2R 1)
CTRL requests L2R to disconnect.

(RLP 1)
L2R in turn requests RLP to disconnect from its peer.

4.5.2 Confirmation from RLP received

```
 CTRL                        L2R                        RLP
   |                          |                          |
   |                          |       RLP_DISC_CNF       |
   |                          *<=======================*
   |                       (L2R 1)                       |
   |       L2R_DISC_CNF       |                          |
   *<=======================*                          |
(CTRL 1)                      |                          |
   |                          |                          |
```

(L2R 1)
RLP confirms to L2R the release of the connection.

(CTRL 1)
L2R in turn confirms to Controller the release of the connection.

4.6 Mobile Terminated Disconnect

4.6.1 Reception of disconnect indication

```
 CTRL                       L2R                        RLP
   |                         |                          |
   |                         |       RLP_DISC_IND       |
   |                         *<======================*
   |                      (L2R 1)                       |
   |       L2R_DISC_IND      |                          |
   *<======================*                          |
 (CTRL 1)                    |                          |
   |                         |                          |
```

(L2R 1)
RLP indicates to L2R, that the connection will be released.

(CTRL 1)
L2R in turn indicates to Controller that the connection will be released. The control of the DTI is also passed to Controller, if L2R has the control.

4.7 Control of DTI is passed to L2R

4.7.1 Reception of L2R_DTI_REQ

```
 CTRL                       L2R                        RLP
   |                         |                          |
   |       L2R_DTI_REQ       |                          |
   |          (CONN)         |                          |
   *======================>*                          |
   |                      (L2R 1)                       |
   |        L2R_DTI_CNF      |                          |
   |          (CONN)         |                          |
   *<======================*                          |
 (CTRL 1)                    |                          |
   |                         |                          |
```

(L2R 1)
Controller passes the control of the DTI to L2R. The dti_conn parameter is set to L2R_CONNECT_DTI.

(CTRL 1)
The handover of the DTI is acknowledged. The dti_conn parameter of the L2R_DTI_REQ is repeated.

4.8 Control of DTI is passed to Controller

4.8.1 Controller requests DTI back

```
 CTRL                       L2R                        RLP
   |                         |                          |
   |        L2R_DTI_REQ      |                          |
   |          (DISC)         |                          |
   *======================>*                          |
   |                      (L2R 1)                       |
   |        L2R_DTI_CNF      |                          |
   |          (DISC)         |                          |
   *<======================*                          |
 (CTRL 1)                    |                          |
   |                         |                          |
```

(L2R 1)
Controller requests the control of the DTI back from L2R. The dti_conn parameter is set to L2R_DISC ONNECT_DTI.

(CTRL 1)
The handover of the DTI is acknowledged.

4.8.2 L2R gives up the DTI connection

In case of receiving the escape in the data stream, the DTI connection is released and this fact is signalled to Controller.

```
 CTRL                      L2R                      RLP
   |                        |                        |
   |        L2R_DTI_IND     |                        |
   *<======================*                         |
(CTRL 1)                    |                        |
   |                        |                        |
```

(CTRL 1)
L2R indicates the interruption of the DTI connection to Controller by a L2R_DTI_IND.

4.9 Mobile Terminated Reset

4.9.1 Reception of reset indication

```
 CTRL                      L2R                      RLP
   |                        |                        |
   |                        |      RLP_RESET_IND     |
   |                        *<======================*
   |                     (L2R 1)                     |
   |      L2R_RESET_IND     |                        |
   *<======================*                         |
(CTRL 1)                    |                        |
   |                        |      RLP_RESET_RES     |
   |                        *======================>*
   |                        |                     (RLP 1)
   |                        |                        |
```

(L2R 1)
RLP indicates to L2R, that the RLP connection is reset.

(CTRL 1)
L2R in turn indicates to Controller that the RLP connection is reset.

(RLP 1)
L2R acknowledges the reset.

4.10 Error indication from lower layer

4.10.1 Reception of error indication

```
 CTRL                      L2R                      RLP
   |                        |                        |
   |                        |      RLP_ERROR_IND     |
   |                        *<======================*
   |                     (L2R 1)                     |
   |                        |                        |
   |      L2R_ERROR_IND     |                        |
   *<======================*                         |
(CTRL 1)                    |                        |
   |                        |                        |
```

(L2R 1)
RLP indicates an error to L2R.

(CTRL 1)
L2R in turn indicates the error to Controller.

4.11 Uplink data transfer

4.11.1 Reception of data from the upper layer. TX buffer has space for more data

```
  UL                        L2R                         RLP
  |                          |                           |
  |      DTI_DATA_REQ        |                           |
  *=======================>*                           |
  |                       (L2R 1)                        |
  |                          |                           |
  |      DTI_READY_IND       |                           |
  *<=======================*                           |
 (UL 1)                      |                           |
  |                          |                           |
```

(L2R 1)
The upper layer sends data to L2R. These data are stored in the TX buffer.

(UL 1)
L2R signals to the upper layer, that it is ready to accept more data.

4.11.2 Reception of data from the upper layer. TX buffer has no space for more data

```
  UL                        L2R                         RLP
  |                          |                           |
  |      DTI_DATA_REQ        |                           |
  *=======================>*                           |
  |                       (L2R 1)                        |
  |                          |                           |
```

(L2R 1)
The upper layer sends data to L2R. These data are stored in the TX buffer. There is no space for more data in the L2R TX buffer. Therefore L2R does not request more data from the upper layer. Data will be requested, when space is available again.

4.11.3 Ready indication from RLP, data are available

```
  UL                        L2R                         RLP
  |                          |                           |
  |                          |      RLP_READY_IND        |
  |                          *<=======================*
  |                       (L2R 1)                        |
  |                          |                           |
  |                          |      RLP_DATA_REQ         |
  |                          *=======================>*
  |                          |                        (RLP 1)
  |      DTI_READY_IND       |                           |
  *<=======================*                           |
 (UL 1)                      |                           |
  |                          |                           |
```

(L2R 1)
RLP indicates to L2R, that it is ready to accept data from L2R.

(RLP 1)
There is enough data available in the TX buffer to fill a complete primitive. L2R sends these data to RLP.

(UL 1)
L2R signals to the upper layer, that it is ready to accept more data.

4.11.4 Ready indication from RLP, data are available with delay

```
  UL                       L2R                      RLP
   |                        |                        |
   |                        |     RLP_READY_IND      |
   |                        *<=======================*
   |                     (L2R 1)                     |
   |                        |                        |
   |      DTI_READY_IND     |                        |
   *<=======================*                        |
 (UL 1)                     |                        |
   |                        |                        |
   |      DTI_DATA_REQ      |                        |
   *=======================>*                        |
   |                     (L2R 2)                     |
   |                        |                        |
   |      DTI_READY_IND     |                        |
   *<=======================*                        |
 (UL 2)                     |                        |
   |                        |     RLP_DATA_REQ       |
   |                        *=======================>*
   |                        |                     (RLP 1)
   |                        |                        |
```

(L2R 1)
RLP indicates to L2R, that it is ready to accept data from L2R. There is not enough data available in the TX buffer, therefore L2R starts a timer and waits until either the data are ready or the timer expires.

(UL 1)
L2R signals to the upper layer, that it is ready to accept more data. In case L2R has already sent a DTI_READY_IND to the upper layer, without receiving a DTI_GETDATA_REQ, this primitive is omitted.

(L2R 1)
The upper layer sends data to L2R. These data are stored in the TX buffer.

(UL 2)
L2R signals to the upper layer, that it is ready to accept more data.

(RLP 1)
Now there is enough data available in the TX buffer to fill a complete primitive. L2R sends these data to RLP.

4.11.5 Timeout of timer UP_T_SND

```
  UL                      L2R                     RLP
   |                       |                       |
   |                       |     RLP_READY_IND     |
   |                       *<=====================*
   |                    (L2R 1)                    |
   |                       |                       |
   |     DTI_READY_IND     |                       |
   *<=====================*                        |
 (UL 1)                    |                       |
   |                       |                       |
   |                       *<Timeout UP_T_SND>     |
   |                    (L2R 2)                    |
   |                       |                       |
   |                       |     RLP_DATA_REQ      |
   |                       *=====================>*
   |                       |                    (RLP 1)
   |                       |                       |
```

(L2R 1)
RLP indicates to L2R, that it is ready to accept data from L2R. There is not enough data available in the TX buffer, therefore L2R starts a timer and waits until either the data are ready or the timer expires.

(UL 1)
L2R signals to the upper layer, that it is ready to accept more data. In case L2R has already sent a DTI_READY_IND to the upper layer, without receiving a DTI_GETDATA_REQ, this primitive is omitted.

(L2R 2)
Timer UP_T_SND expires.

(RLP 1)
L2R sends all available data to RLP. If there is no data available, an empty primitive is sent.

4.12 Downlink data transfer

It is assumed, that no BREAK signal is included in the data, unless specifically noted.

4.12.1 Reception of data from RLP, RX buffer has space for more data

```
  UL                      L2R                     RLP
   |                       |                       |
   |                       |     RLP_DATA_IND      |
   |                       *<=====================*
   |                    (L2R 1)                    |
   |                       |                       |
   |                       |    RLP_GETDATA_REQ    |
   |                       *=====================>*
   |                       |                    (RLP 1)
   |                       |                       |
```

(L2R 1)
RLP sends data to L2R.

(RLP 1)
There is space for more data in the L2R RX buffer. Therefore L2R sends a RLP_GETDATA_REQ to RLP to request more data from RLP.

4.12.2 Reception of data from RLP, RX buffer has no space for more data

```
  UL                      L2R                       RLP
   |                       |                         |
   |                       |      RLP_DATA_IND       |
   |                       *<=======================*
   |                    (L2R 1)                      |
   |                       |                         |
```

(L2R 1)
RLP sends data to L2R. There is no space for more data in the L2R RX buffer. Therefore L2R does not request more data from RLP. Data will be requested, when space is available again.

4.12.3 Upper layer is ready for data. Data are available

```
  UL                      L2R                       RLP
   |                       |                         |
   |    DTI_GETDATA_REQ    |                         |
   *=======================>*                        |
   |                    (L2R 1)                      |
   |                       |                         |
   |      DTI_DATA_IND     |                         |
   *<======================*                         |
 (UL 1)                    |                         |
   |                       |     RLP_GETDATA_REQ     |
   |                       *=======================>*
   |                       |                      (RLP 1)
   |                       |                         |
```

(L2R 1)
The upper layer requests L2R to pass data to the upper layer.

(UL 1)
Data are available. L2R sends these data to the upper layer.

(RLP 1)
L2R requests data from RLP.

4.12.4 Upper layer is ready for data. Data are available with delay

```
  UL                        L2R                         RLP
  |                          |                           |
  |    DTI_GETDATA_REQ       |                           |
  *=======================>*                           |
  |                       (L2R 1)                        |
  |                          |                           |
  |                          |     RLP_GETDATA_REQ       |
  |                          *=======================>*
  |                          |                       (RLP 1)
  |                          |                           |
  |                          |      RLP_DATA_IND         |
  |                          *<=======================*
  |                       (L2R 2)                        |
  |                          |                           |
  |      DTI_DATA_IND        |                           |
  *<=======================*                           |
 (UL 1)                      |                           |
  |                          |                           |
  |                          |     RLP_GETDATA_REQ       |
  |                          *=======================>*
  |                          |                       (RLP 2)
  |                          |                           |
```

(L2R 1)
The upper layer requests L2R to pass data to the upper layer. There is no data available.

(RLP 1)
L2R requests data from RLP.

(L2R 2)
RLP sends data to L2R.

(UL 1)
L2R sends these data to the upper layer.

(RLP 2)
There is space for more data in the L2R RX buffer. Therefore L2R sends a RLP_GETDATA_REQ to RLP to request more data from RLP.

4.12.5 Reception of BREAK from RLP. Data compression is not used

```
  UL                        L2R                         RLP
  |                          |                           |
  |                          |      RLP_DATA_IND         |
  |                          *<=======================*
  |                       (L2R 1)                        |
  |                          |                           |
  |      L2R_BREAK_IND       |                           |
  *<=======================*                           |
 (UL 1)                      |                           |
  |                          |                           |
  |                          |     RLP_GETDATA_REQ       |
  |                          *=======================>*
  |                          |                       (RLP 1)
  |                          |                           |
```

(L2R 1)
RLP sends data to L2R. The data contain a BREAK. The data are discarded by L2R. All buffers of L2R are cleared. In case there is data following the BREAK in the primitive, these data are not discarded but stored in the RX buffer and will be sent to the upper layer.

(UL 1)
L2R indicates the BREAK with a L2R_BREAK_IND to the upper layer.

(RLP 1)
There is space for more data in the L2R RX buffer. Therefore L2R sends a RLP_GETDATA_REQ to RLP to request more data from RLP.

4.12.6 Reception of BREAK from RLP. Data compression is used

```
  UL                     L2R                      RLP
  |                       |                        |
  |                       |     RLP_DATA_IND       |
  |                       *<=======================*
  |                    (L2R 1)                     |
  |                       |                        |
  |     L2R_BREAK_IND     |                        |
  *<======================*                        |
 (UL 1)                   |                        |
  |                       |                        |
  |                       |    RLP_GETDATA_REQ     |
  |                       *=======================>*
  |                       |                     (RLP 1)
  |                       |                        |
  |                       |    RLP_DATA_REQ        |
  |                       *=======================>*
  |                       |                     (RLP 2)
  |                       |                        |
```

(L2R 1)
RLP sends data to L2R. The data contain a BREAK. The data are discarded by L2R. All buffers of L2R and the DTI are cleared.

(UL 1)
L2R indicates the BREAK with a L2R_BREAK_IND to the upper layer. The upper layer resets the data compression function.

(RLP 1)
There is space for more data in the L2R RX buffer. Therefore L2R sends a RLP_GETDATA_REQ to RLP to request more data from RLP.

(RLP 2)

L2R sends a primitive with a BREAK acknowledge L2R PDU to RLP. If RLP has not sent a RLP_READY_IND previously, L2R has to wait for the next RLP_READY_IND to send this BREAK acknowledge.

4.13 Pending BREAK request

4.13.1 Reception of BREAK from upper layer

```
UL                     L2R                     RLP
|                       |                       |
|     L2R_BREAK_REQ     |                       |
*======================>*                       |
|                    (L2R 1)                    |
|                       |                       |
|                       |     RLP_DATA_REQ      |
|                       *======================>*
|                       |                    (RLP 1)
|                       |                       |
```

(L2R 1)
The upper layer sends a L2R_BREAK_REQ. All L2R buffers are cleared. If data compression is used the 'pending BREAK request' state is entered, in which no data transfer is possible until a corresponding BREAK acknowledgement has been received.

(RLP 1)
L2R sends a primitive with a BREAK request L2R PDU to RLP. If RLP has not sent a RLP_READY_IND previously, L2R has to wait for the next RLP_READY_IND to send this BREAK request.

4.13.2 Data are received from RLP

```
UL                     L2R                     RLP
|                       |                       |
|                       |      RLP_DATA_IND     |
|                       *<======================*
|                    (L2R 1)                    |
|                       |                       |
|                       |     RLP_GETDATA_REQ   |
|                       *======================>*
|                       |                    (RLP 1)
|                       |                       |
```

(L2R 1)
RLP sends data to L2R. The data contain no BREAK acknowledge. The data are discarded by L2R.

(RLP 1)
L2R sends a RLP_GETDATA_REQ to RLP to request data from RLP.

4.13.3 A BREAK acknowledge is received from RLP

```
UL                     L2R                     RLP
|                       |                       |
|                       |      RLP_DATA_IND     |
|                       *<======================*
|                    (L2R 1)                    |
|                       |                       |
|      L2R_BREAK_CNF    |                       |
*<======================*                       |
(UL 1)                  |                       |
|                       |                       |
|                       |     RLP_GETDATA_REQ   |
|                       *======================>*
|                       |                    (RLP 1)
|                       |                       |
```

(L2R 1)
RLP sends data to L2R, which contain a BREAK acknowledge. There is no data following the BREAK acknowledge in the primitive. Data, which are possibly in the primitive before the BREAK acknowledge are discarded by L2R.

(UL 1)
L2R confirms the previous L2R_BREAK_REQ and quits the 'pending BREAK request' state.

(RLP 1)
L2R sends a RLP_GETDATA_REQ to RLP to request data from RLP.

4.13.4 A BREAK acknowledge with additional data is received from RLP

```
  UL                      L2R                      RLP
  |                        |                        |
  |                        |      RLP_DATA_IND      |
  |                        *<======================*
  |                     (L2R 1)                     |
  |                        |                        |
  |      L2R_BREAK_CNF     |                        |
  *<======================*                         |
 (UL 1)                    |                        |
  |                        |                        |
  |      DTI_DATA_IND      |                        |
  *<======================*                         |
 (UL 2)                    |                        |
  |                        |                        |
  |                        |     RLP_GETDATA_REQ    |
  |                        *======================>*
  |                        |                     (RLP 1)
  |                        |                        |
```

(L2R 1)
RLP sends data to L2R, which contain a BREAK acknowledge. Data, which are possibly in the primitive before the BREAK acknowledge are discarded by L2R. There is data following the BREAK acknowledge in the primitive. These data are stored in the RX buffer.

(UL 1)
L2R confirms the previous L2R_BREAK_REQ and quits the 'pending BREAK request' state.

(UL 2)
L2R sends the data, which have been in the RLP_DATA_IND to the upper layer. If the upper layer has not send a DTI_GETDATA_REQ previously, no primitive is sent and the data remain in the RX buffer. In this case the following primitive is only sent, if there is more space available in the RX buffer.

(RLP 1)
L2R sends a RLP_GETDATA_REQ to RLP to request data from RLP.

Appendices

A. Acronyms

DS-WCDMA	Direct Sequence/Spread Wideband Code Division Multiple Access

B. Glossary

International Mobile Telecommunication 2000 (IMT-2000/ITU-2000)	Formerly referred to as FPLMTS (Future Public Land-Mobile Telephone System), this is the ITU's specification/family of standards for 3G. This initiative provides a global infrastructure through both satellite and terrestrial systems, for fixed and mobile phone users. The family of standards is a framework comprising a mix/blend of systems providing global roaming. <URL: http://www.imt-2000.org/>