Link Layer Interface Specification

Link Layer Interface Specification

Version 0.9.2 Edition 1
Updated 2008-10-31
Distributed with Package strx25-0.9.2.1

Abstract

This document is a Specification containing technical details concerning the implementation of the Link Layer Interface for OpenSS7. It contains recommendations on software architecture as well as platform and system applicability of the Link Layer Interface. It provides abstraction of the LAPB Protocol (ISO/IEC 7776) and LLC2 Protocol (ISO/IEC 8802) service interface to these components as well as providing a basis for link layer control for other protocols.

Brian Bidulock <bidulock@openss7.org> for
The OpenSS7 Project <http://www.openss7.org/>

Published by:

OpenSS7 Corporation
1469 Jefferys Crescent
Edmonton, Alberta T6L 6T1
Canada

Notice:

Short Contents

Table of Contents

List of Figures

List of Tables

1 Introduction

UNIX System V Release 3.2 included a Logical Link Interface (LLI) that was later standardized as the Data Link Provider Interface (DLPI) by the *Open Group*.[1]

The Link Layer Interface (LLI) was developed by Spider Systems, Ltd., (now a division of Emerson Power) and is widely available on many platforms. For example, *AIX AIXlink/X.25*, *HP-UX X.25/9000*, *IRIX IRIS SX.25*, *PT X.25*, *RadiSys WAN*, *SBE X.25*, *Solaris SunLink X.25* and *Solaris Solstice X.25*, implement the Link Layer Interface (LLI).[2]

The Link Layer Interface (LLI) as designed to be used directly with standard *STREAMS* system calls and does not require the use of a cooperating user space shared library. Applications programs directly use the `getmsg(2s)`, `getpmsg(2)`, `putmsg(2s)`, `putpmsg(2)` and `ioctl(2)` system calls.[3] Nevertheless, user shared object libraries can easily be constructed using this *STREAMS* service primitive interface.

The system header files that must be included when compiling user applications, or *STREAMS* drivers and modules that use the interface, are detailed in Chapter 6 [LLI Header Files], page 27.

A user library, '`libsx25`', is provided, not for interfacing to the message primitive service interface, but for providing various helper functions when using the *STREAMS* service interface. This library is detailed in ⟨undefined⟩ [⟨undefined⟩], page ⟨undefined⟩.

[1] Evidence of this can still be found in the '`dlpi.h`' header file distributed by *The Open Group* in the DLPI standard. Reference is made in the header file to backward compatibility to the LLI interface.

[2] But, for the most part, it is not called the *Link Lyaer Interface*, but is instead sunk inside the Network Layer Interface (NLI).

[3] See `getmsg(2s)`, `getpmsg(2)`, `putmsg(2s)`, `putpmsg(2)` and `ioctl(2)` manual pages.

2 Model of the Link Layer

3 LLI Services

3.1 LLI Commands

3.2 LLI Data Structures

4 LLI Message Primitives

Although it perhaps did not at first, the LLI uses the service primitives and message format of the *Data Link Provider Interface*, [DLPI], page 43.

5 LLI Input-Output Controls

5.1 Input-Output Control Data Structures

The '`/usr/include/strx25/sys/snet/ll_control.h`' header file ('`<sys/snet/ll_control.h>`' with proper compile flags) defines a number of structures, pointers to which are used as argumnets to input-output controls. These structures fall into four classes, identified by the value of the first byte of the structure, as follows:

`LI_PLAIN` A `ll_hdioc` structure that identifies the subnetwork (link)

`LI_SNID` A `ll_snioc` structure that identifies the subnetwork (link)

`LI_STATUS` A `ll_stnioc` structure that identifies the subnetwork (link)

`LI_STATS` A `ll_stioc` or `mlp_stioc` structure that identifies the subnetwork (link)

`LI_GSTATS` A `ll_gstioc` or `mlp_gstioc` structure that identifies the subnetwork (link)

`LI_LAPBTUNE` A `lapb_tnioc` structure that identifies the subnetwork (link)

`LI_LLC2TUNE` A `llc2_tnioc` structure that identifies the subnetwork (link)

`LI_MLPTUNE` A `mlp_tnioc` structure that identifies the subnetwork (link)

5.1.1 LI_PLAIN - Plain Data Structures

5.1.1.1 ll_hdioc Structure

The `ll_hdioc` structure is formatted as follows:

```
struct ll_hdioc {
    uint8_t lli_type;
    uint8_t lli_spare[3];
    uint32_t lli_snid;
};
```

The `ll_hdioc` structure contains the following members:

`lli_type` Always `LLI_PLAIN`.

`lli_spare`
Spare bytes for alignment: set to zero by issuer and ignored by responder.

`lli_snid` Specifies the subnetwork identifier (link identifier) for the operation.

5.1.2 LI_SNID - Subnetwork Identifier Data Strucures

5.1.2.1 ll_snioc Structure

The `ll_snioc` structure is used in the `L_SETSNID`, `L_GETSNID`, `M_SETSNID` and `M_GETSNID` input-output controls.

The `ll_snioc` structure is formatted as follows:

```
struct ll_snioc {
    uint8_t lli_type;
    uint8_t lli_class;
    uint8_t lli_spare[2];
    uint32_t lli_snid;
    uint32_t lli_index;
    uint32_t lli_slp_snid;
    uint16_t lli_slp_pri;
};
```

The `ll_snioc` structure contains the following members:

`lli_type` Always `LLI_SNID`.

`lli_class`
Specifies the class of the link being registered. This field can be one of the following values:

`LC_LLC1`	LLC Type 1 link.
`LC_LLC2`	LLC TYpe 2 link.
`LC_LAPBDTE`	LAPB DTE.
`LC_LAPBXDTE`	LAPB DTE with extended addressing.
`LC_LAPBDCE`	LAPB DCE.
`LC_LAPBXDCE`	LAPB DCE with extended addressing.
`LC_LAPDTE`	LAP DTE.
`LC_LAPDCE`	LAP DCE.
`LC_HDLC`	HDLC.
`LC_HDLCX`	HDLC with extended addressing.
`LC_MLAPBDTE`	LAPB DTE multilink.
`LC_MLAPBXDTE`	LAPB DTE multilink with extended addressing.
`LC_MLAPBDCE`	LAPB DCE multilink.
`LC_MLAPBXDCE`	LAPB DCE multilink with extended addressing.
`LC_MLP`	MLP link.

`lli_spare`
Spare bytes for alignment: set to zero by issuer and ignored by responder.

`lli_snid` Specifies the subnetwork identifier (link identifier) for the operation.

`lli_index`
Specifies the lower multiplex identifier of the Stream linked under a multiplexing driver as returned from `I_LINK(7)`.[1]

[1] See the `I_LINK(7)` manual page.

`lli_slp_snid`
: Specifies the subnetwork (link) identifier for an SLP link within an MLP link.

`lli_slp_pri`
: Specifies the subnetwork (link) priority within an MLP link.

5.1.3 LI_STATUS - Status Data Strucures

5.1.3.1 ll_stnioc Structure

The `ll_stnioc` structure is formatted as follows:

```
struct ll_stnioc {
    uint8_t lli_type;
    uint8_t lli_spare[3];
    uint32_t lli_snid;
};
```

The `ll_stnioc` structure contains the following members:

`lli_type` Always `LLI_STATUS`.

`lli_spare`
: Spare bytes for alignment: set to zero by issuer and ignored by responder.

`lli_snid` Specifies the subnetwork identifier (link identifier) for the operation.

5.1.4 LI_STATS - Statistics Data Structures

5.1.4.1 lapb_stioc Structure

The `lapb_stioc` structure is formatted as follows:

```
struct lapb_stioc {
    uint8_t lli_type;
    uint8_t state;
    uint16_t lli_spare;
    uint32_t lli_snid;
    lapbstats_t lli_stats;
};
```

The `lapb_stioc` structure contains the following members:

`lli_type` Always `LLI_STATS`.

`state`

`lli_spare`
: Spare bytes for alignment: set to zero by issuer and ignored by responder.

`lli_snid` Specifies the subnetwork identifier (link identifier) for the operation.

`lli_stats`
: Contains the `lapbstats_t` structure described in Section 5.1.4.2 [lapbstats't Structure], page 14.

5.1.4.2 lapbstats_t Structure

The `lapbstats_t` structure is formatted as follows:

```
typedef struct lapb_stats {
    uint32_t lapbmonarray[49];
} lapbstats_t;
```

The `lapbstats_t` structure contains the following members:

`lapbmonarray`

Provides an array of 32-bit unsigned integers containing statistics. The indexes of the elements of the array are as follows:

0	`tx_ign`	-
1	`rx_badlen`	-
2	`rx_unknown`	-
3	`t1_exp`	-
4	`t4_exp`	-
5	`t4_n2_exp`	-
6	`RR_rx_cmd`	-
7	`RR_rx_rsp`	-
8	`RR_tx_cmd`	-
9	`RR_tx_rsp`	-
10	`RR_tx_cmd_p`	-
11	`RNR_rx_cmd`	-
12	`RNR_rx_rsp`	-
13	`RNR_tx_cmd`	-
14	`RNR_tx_rsp`	-
15	`RNR_tx_cmd_p`	-
16	`REJ_rx_cmd`	-
17	`REJ_rx_rsp`	-
18	`REJ_tx_cmd`	-
19	`REJ_tx_rsp`	-
20	`REJ_tx_cmd_p`	-
21	`SABME_rx_cmd`	-
22	`SABME_tx_cmd`	-
23	`DISC_rx_cmd`	-
24	`DISC_tx_cmd`	-
25	`UA_rx_rsp`	-
26	`UA_tx_rsp`	-
27	`DM_rx_rsp`	-
28	`DM_tx_rsp`	-
29	`I_rx_cmd`	-
30	`I_tx_cmd`	-
31	`FRMR_rx_rsp`	-
32	`FRMR_tx_rsp`	-
33	`tx_rtr`	-
34	`rx_bad`	-

35	rx_dud	-
36	rx_ign	-
37	XID_rx_cmd	-
38	XID_rx_rsp	-
39	XID_tx_cmd	-
40	XID_tx_rsp	-
41	TEST_rx_cmd	-
42	TEST_rx_rsp	-
43	TEST_tx_cmd	-
44	TEST_tx_rsp	-
45	SABM_rx_cmd	-
46	SABM_tx_cmd	-
47	SARM_rx_cmd	-
48	SARM_tx_cmd	-

5.1.4.3 llc2_stioc Structure

The `llc2_stioc` structure is formatted as follows:

```
struct llc2_stioc {
    uint8_t lli_type;
    uint8_t lli_spare[3];
    uint32_t lli_snid;
    llc2stats_t lli_stats;
};
```

The `llc2_stioc` structure contains the following members:

`lli_type` Always `LLI_STATS`.

`lli_spare`
Spare bytes for alignment: set to zero by issuer and ignored by responder.

`lli_snid` Specifies the subnetwork identifier (link identifier) for the operation.

`lli_stats`
Contains the `llc2stats_t` structure containing the statistics described in Section 5.1.4.4 [llc2stats˙t Structure], page 15.

5.1.4.4 llc2stats_t Structure

The `llc2stats_t` structure is formatted as follows:

```
typedef struct llc2_stats {
    uint32_t llc2monarray[49];
} llc2stats_t;
```

The `llc2stats_t` structure contains the following members:

`llc2monarray`
Provides an array of 32-bit unsigned integers containing the statistics. The indices of the individual elements are as follows:

0	tx_ign	-

1	rx_badlen	-
2	rx_unknown	-
3	t1_exp	-
4	t4_exp	-
5	t4_n2_exp	-
6	RR_rx_cmd	-
7	RR_rx_rsp	-
8	RR_tx_cmd	-
9	RR_tx_rsp	-
10	RR_tx_cmd_p	-
11	RNR_rx_cmd	-
12	RNR_rx_rsp	-
13	RNR_tx_cmd	-
14	RNR_tx_rsp	-
15	RNR_tx_cmd_p	-
16	REJ_rx_cmd	-
17	REJ_rx_rsp	-
18	REJ_tx_cmd	-
19	REJ_tx_rsp	-
20	REJ_tx_cmd_p	-
21	SABME_rx_cmd	-
22	SABME_tx_cmd	-
23	DISC_rx_cmd	-
24	DISC_tx_cmd	-
25	UA_rx_rsp	-
26	UA_tx_rsp	-
27	DM_rx_rsp	-
28	DM_tx_rsp	-
29	I_rx_cmd	-
30	I_tx_cmd	-
31	FRMR_rx_rsp	-
32	FRMR_tx_rsp	-
33	tx_rtr	-
34	rx_bad	-
35	rx_dud	-
36	rx_ign	-
37	XID_rx_cmd	-
38	XID_rx_rsp	-
39	XID_tx_cmd	-
40	XID_tx_rsp	-
41	TEST_rx_cmd	-
42	TEST_rx_rsp	-
43	TEST_tx_cmd	-
44	TEST_tx_rsp	-
45	I_rx_rsp	-

46	I_tx_rsp	-
47	UI_rx_cmd	-
48	UI_tx_cmd	-

5.1.4.5 mlp_stioc Structure

The `mlp_stioc` structure is formatted as follows:

```
struct mlp_stioc {
    uint8_t lli_type;
    uint8_t state;
    uint16_t lli_spare;
    uint32_t lli_snid;
    mlpstats_t lli_stats;
};
```

The `mlp_stioc` structure contains the following members:

`lli_type` Always `LLI_STATS`.

`lli_state`
: Provides the state of the link.

`lli_spare`
: Spare bytes for alignment: set to zero by issuer and ignored by responder.

`lli_snid` Specifies the subnetwork identifier (link identifier) for the operation.

`lli_stats`
: Contains the `mlpstats_t` structure containing the statistics described in Section 5.1.4.6 [mlpstats˙t Structure], page 17.

5.1.4.6 mlpstats_t Structure

The `mlpstats_t` structure is formatted as follows:

```
typedef struct mlp_stats {
    uint32_t mlpmonarray[12];
} mlpstats_t;
```

The `mlpstats_t` structure contains the following members:

`mlpmonarray`
: Contains an array of 32-bit unsigned integers containing the statistics. The indices of the individual elements of the array are as follows:

0	MLP_frames_tx	-
1	MLP_frames_rx	-
2	MLP_reset_tx	-
3	MLP_reset_rx	-
4	MLP_confs_tx	-
5	MLP_confs_rx	-
6	MLP_slps	-
7	MLP_num_slps	-

8	MLP_mt1_exp	-
9	MLP_mt2_exp	-
10	MLP_mt3_exp	-
11	MLP_mn1_exp	-

5.1.5 LI_GSTATS - Global Statistics Data Structures

5.1.5.1 lapb_gstioc Structure

The `lapb_gstioc` structure is formatted as follows:

```
struct lapb_gstioc {
    uint8_t lli_type;
    uint8_t lli_spare[3];
    uint32_t lapbgstats[6];
};
```

The `lapb_gstioc` structure contains the following members:

`lli_type` Always `LLI_GSTATS`.

`lli_spare`
Spare bytes for alignment: set to zero by issuer and ignored by responder.

`lapbgstats`
Contains an array of 32-bit unsigned integers containing the statistics. The indices of the individual elements of the array are as follows:

0	frames_tx	-
1	frames_rx	-
2	sabm_tx	-
3	sabm_rx	-
4	bytes_tx	-
5	bytes_rx	-

5.1.5.2 llc2_gstioc Structure

The `llc2_gstioc` structure is formatted as follows:

```
struct llc2_gstioc {
    uint8_t lli_type;
    uint8_t lli_spare[3];
    uint32_t llc2gstats[6];
};
```

The `llc2_gstioc` structure contains the following members:

`lli_type` Always `LLI_GSTATS`.

`lli_spare`
Spare bytes for alignment: set to zero by issuer and ignored by responder.

`llc2gstats`
Contains an array of 32-bit unsigned integers containing the statistics. The indices of the individual elements of the array are as follows:

0	frames_tx	-
1	frames_rx	-
2	sabme_tx	-
3	sabme_rx	-
4	bytes_tx	-
5	bytes_rx	-

5.1.5.3 mlp_gstioc Structure

The `mlp_gstioc` structure is formatted as follows:

```
struct mlp_gstioc {
    uint8_t lli_type;
    uint8_t lli_spare[3];
    uint32_t mlpgstats[10];
};
```

The `mlp_gstioc` structure contains the following members:

`lli_type` Always `LLI_GSTATS`.

`lli_spare`
Spare bytes for alignment: set to zero by issuer and ignored by responder.

`mlpgstats`
Contains an array of 32-bit unsigned integers containing the statistics. The indices of the individual elements of the array are as follows:

0	MLP_frames_tx	-
1	MLP_frames_rx	-
2	MLP_reset_tx	-
3	MLP_reset_rx	-
4	MLP_confs_tx	-
5	MLP_confs_rx	-
6	MLP_slps	-
7	MLP_num_slps	-
8	MLP_bytes_tx	-
9	MLP_bytes_rx	-

5.1.6 LI_LAPBTUNE - LAPB Tuning Data Structures

5.1.6.1 lapb_tnioc Structure

The `lapb_tnioc` structure is formatted as follows:

```
struct lapb_tnioc {
    uint8_t lli_type;
    uint8_t lli_spare[3];
    uint32_t lli_snid;
    lapbtune_t lapb_tune;
};
```

The `lapb_tnioc` structure contains the following members:

`lli_type` Always `LLI_LAPBTUNE`.

`lli_spare`
Spare bytes for alignment: set to zero by issuer and ignored by responder.

`lli_snid` Specifies the subnetwork identifier (link identifier) for the operation.

`lapb_tune`
Contains the `lapbtune_t` tuning structure containing tuning parameters described in Section 5.1.6.2 [lapbtune˙t Structure], page 20.

5.1.6.2 lapbtune_t Structure

The `labtune_t` structure is formatted as follows:

```
typedef struct lapbtune {
    uint16_t N2;
    uint16_t T1;
    uint16_t Tpf;
    uint16_t Trej;
    uint16_t Tbusy;
    uint16_t Tidle;
    uint16_t ack_delay;
    uint16_t notack_max;
    uint16_t tx_window;
    uint16_t tx_probe;
    uint16_t max_I_len;
    uint16_t llconform;
    uint16_t sabm_in_x32;
} lapbtune_t;
```

The `lapbtune_t` structure contains the following members:

`N2`

`T1`

`Tpf`

`Trej`

`Tbusy`

`Tidle`

`ack_delay`
`notack_max`
`tx_window`
`tx_probe`

`max_I_len`
`llconform`
`sabm_in_x32`

5.1.7 LI_LLC2TUNE - LLC2 Tuning Data Structures

5.1.7.1 llc2_tnioc Structure

The `llc2_tnioc` structure is formatted as follows:

```
struct llc2_tnioc {
    uint8_t lli_type;
    uint8_t lli_spare[3];
    uint32_t lli_snid;
    llc2tune_t llc2_tune;
};
```

The `llc2_tnioc` structure contains the following members:

`lli_type` Always `LLI_LLC2TUNE`.

`lli_spare`
Spare bytes for alignment: set to zero by issuer and ignored by responder.

`lli_snid` Specifies the subnetwork identifier (link identifier) for the operation.

`llc2_tune`
Contains the `llc2tune_t` tuning structure containing tuning parameters described in Section 5.1.7.2 [llc2tune˙t Structure], page 21.

5.1.7.2 llc2tune_t Structure

The `llc2tune_t` structure is formatted as follows:

```
typedef struct llc2tune {
    uint16_t N2;
    uint16_t T1;
    uint16_t Tpf;
    uint16_t Trej;
    uint16_t Tbusy;
    uint16_t Tidle;
    uint16_t ack_delay;
    uint16_t notack_max;
    uint16_t tx_window;
    uint16_t tx_probe;
    uint16_t max_I_len;
    uint16_t xid_window;
    uint16_t xid_Ndup;
    uint16_t xid_Tdup;
} llc2tune_t;
```

The `llc2tune_t` structure contains the following members:

`N2`

`T1`

`Tpf`

`Trej`

`Tbusy`

`Tidle`

`ack_delay`
`notack_max`
`tx_window`
`tx_probe`

`max_I_len`
`xid_window`
`xid_Ndup`

`xid_Tdup`

5.1.8 LI_MLPTUNE - MLP Tuning Data Structures

5.1.8.1 mlp_tnioc Structure

The `mlp_tnioc` structure is formatted as follows:

```
struct mlp_tnioc {
    uint8_t lli_type;
    uint8_t lli_spare[3];
    uint32_t lli_snid;
    mlptune_t mlp_tune;
};
```

The `mlp_tnioc` structure contains the following members:

`lli_type` Always `LLI_MLPTUNE`.

`lli_spare`
Spare bytes for alignment: set to zero by issuer and ignored by responder.

`lli_snid` Specifies the subnetwork identifier (link identifier) for the operation.

`mlp_tune` Contains the `mlptune_t` tuning structure containing tuning parameters described in Section 5.1.8.2 [mlptune't Structure], page 22.

5.1.8.2 mlptune_t Structure

The `mlptune_t` structure is formatted as follows:

```
typedef struct mlptune {
    uint16_t mw;
    uint16_t mx;
    uint16_t mt1;
    uint16_t mt2;
    uint16_t mt3;
    uint16_t mn1;
} mlptune_t;
```

The `mlptune_t` structure contains the following members:

`mw` The size of the MLP window.

`mx` The size of the MLP guard region.

`mt1` The time interval to wait for MN(S) == MV(R) in deciseconds.

`mt2` The time interval to wait for unblock in deciseconds.

`mt3` The time interval to wait for reset confirmation.

`mn1` The number of SLP transmission retries.

5.2 Input-Output Control Commands

`L_SETSNID`	Set subnetwork identifier.
`L_GETSNID`	Get subnetwork identifier.
`L_SETTUNE`	Set common tuning parameters.
`L_GETTUNE`	Get common tuning parameters.
`L_GETSTATS`	Get subnetwork statistics.
`L_ZEROSTATS`	Zero subnetwork statistics.
`L_TRACEON`	Turn message tracing on.
`L_TRACEOFF`	Turn message tracing off.
`L_GETGSTATS`	Get global statistics.
`L_ZEROGSTATS`	Zero global statistics.
`L_LINKDISABLE`	Disable link (subnetwork identifier).
`L_LINKENABLE`	Enable link (subnetwork identifier).
`L_PUTX32MAP`	Put X.32 table mapping.
`L_GETX32MAP`	Get X.32 table mapping.
`M_SETSNID`	Set subnetwork identifier.
`M_GETSNID`	Get subnetwork identifier.
`M_SETTUNE`	Set common tuning parameters.
`M_GETTUNE`	Get common tuning parameters.
`M_GETSTATS`	Get subnetwork statistics.
`M_ZEROSTATS`	Zero subnetwork statistics.
`M_TRACEON`	Turn message tracing on.
`M_TRACEOFF`	Turn message tracing off.
`M_GETGSTATS`	Get global statistics.
`M_ZEROGSTATS`	Zero global statistics.

5.2.1 SLP Input-Output Control Commands

5.2.1.1 L_SETSNID

The argument of this input-output control is a pointer to the `ll_snioc` structure described in Section 5.1.2.1 [ll˙snioc Structure], page 12.

5.2.1.2 L_GETSNID

The argument of this input-output control is a pointer to the `ll_snioc` structure described in Section 5.1.2.1 [ll˙snioc Structure], page 12.

5.2.1.3 L_SETTUNE

The argument of this input-output control is a pointer to the `lapb_tnioc`, `llc2_tnioc` or `mlp_tnioc` structure.[2]

5.2.1.4 L_GETTUNE

The argument of this input-output control is a pointer to the `lapb_tnioc`, `llc2_tnioc` or `mlp_tnioc` structure.[3]

5.2.1.5 L_GETSTATS

The argument of this input-output control is a pointer to the `lapb_stioc`, `llc2_stioc` or `mlp_stioc` structure.[4]

5.2.1.6 L_ZEROSTATS

The argument of this input-output control is a pointer to the `lapb_stioc`, `llc2_stioc` or `mlp_stioc` structure.[5]

5.2.1.7 L_TRACEON

The argument of this input-output control is a pointer to the `ll_hdioc` structure described in Section 5.1.1.1 [ll˙hdioc Structure], page 11.

5.2.1.8 L_TRACEOFF

The argument of this input-output control is a pointer to the `ll_hdioc` structure described in Section 5.1.1.1 [ll˙hdioc Structure], page 11.

5.2.1.9 L_GETGSTATS

The argument of this input-output control is a pointer to the `lapb_gstioc`, `llc2_gstioc` or `mlp_gstioc` structure.[6]

5.2.1.10 L_ZEROGSTATS

The argument of this input-output control is a pointer to the `lapb_gstioc`, `llc2_gstioc` or `mlp_gstioc` structure.[7]

---

[2] These structures are described in Section 5.1.6.1 [lapb˙tnioc Structure], page 19, Section 5.1.7.1 [llc2˙tnioc Structure], page 21, and Section 5.1.8.1 [mlp˙tnioc Structure], page 22.

[3] These structures are described in Section 5.1.6.1 [lapb˙tnioc Structure], page 19, Section 5.1.7.1 [llc2˙tnioc Structure], page 21, and Section 5.1.8.1 [mlp˙tnioc Structure], page 22.

[4] These structures are described in Section 5.1.4.1 [lapb˙stioc Structure], page 13, Section 5.1.4.3 [llc2˙stioc Structure], page 15, and Section 5.1.4.5 [mlp˙stioc Structure], page 17.

[5] These structures are described in Section 5.1.4.1 [lapb˙stioc Structure], page 13, Section 5.1.4.3 [llc2˙stioc Structure], page 15, and Section 5.1.4.5 [mlp˙stioc Structure], page 17.

[6] These structures are described in Section 5.1.5.1 [lapb˙gstioc Structure], page 18, Section 5.1.5.2 [llc2˙gstioc Structure], page 18, and Section 5.1.5.3 [mlp˙gstioc Structure], page 19.

[7] These structures are described in Section 5.1.5.1 [lapb˙gstioc Structure], page 18, Section 5.1.5.2 [llc2˙gstioc Structure], page 18, and Section 5.1.5.3 [mlp˙gstioc Structure], page 19.

5.2.1.11 L_LINKDISABLE

The argument of this input-output control is a pointer to the `ll_hdioc` structure described in Section 5.1.1.1 [ll˙hdioc Structure], page 11.

5.2.1.12 L_LINKENABLE

The argument of this input-output control is a pointer to the `ll_hdioc` structure described in Section 5.1.1.1 [ll˙hdioc Structure], page 11.

5.2.1.13 L_PUTX32MAP

5.2.1.14 L_GETX32MAP

5.2.2 MLP Input-Output Control Commands

5.2.2.1 M_SETSNID

The argument of this input-output control is a pointer to the `ll_snioc` structure described in Section 5.1.2.1 [ll˙snioc Structure], page 12.

5.2.2.2 M_GETSNID

The argument of this input-output control is a pointer to the `ll_snioc` structure described in Section 5.1.2.1 [ll˙snioc Structure], page 12.

5.2.2.3 M_SETTUNE

The argument of this input-output control is a pointer to the `lapb_tnioc`, `llc2_tnioc` or `mlp_tnioc` structure.[8]

5.2.2.4 M_GETTUNE

The argument of this input-output control is a pointer to the `lapb_tnioc`, `llc2_tnioc` or `mlp_tnioc` structure.[9]

5.2.2.5 M_GETSTATS

The argument of this input-output control is a pointer to the `lapb_stioc`, `llc2_stioc` or `mlp_stioc` structure.[10]

5.2.2.6 M_ZEROSTATS

The argument of this input-output control is a pointer to the `lapb_stioc`, `llc2_stioc` or `mlp_stioc` structure.[11]

---

[8] These structures are described in Section 5.1.6.1 [lapb˙tnioc Structure], page 19, Section 5.1.7.1 [llc2˙tnioc Structure], page 21, and Section 5.1.8.1 [mlp˙tnioc Structure], page 22.

[9] These structures are described in Section 5.1.6.1 [lapb˙tnioc Structure], page 19, Section 5.1.7.1 [llc2˙tnioc Structure], page 21, and Section 5.1.8.1 [mlp˙tnioc Structure], page 22.

[10] These structures are described in Section 5.1.4.1 [lapb˙stioc Structure], page 13, Section 5.1.4.3 [llc2˙stioc Structure], page 15, and Section 5.1.4.5 [mlp˙stioc Structure], page 17.

[11] These structures are described in Section 5.1.4.1 [lapb˙stioc Structure], page 13, Section 5.1.4.3 [llc2˙stioc Structure], page 15, and Section 5.1.4.5 [mlp˙stioc Structure], page 17.

5.2.2.7 M_TRACEON

The argument of this input-output control is a pointer to the `ll_hdioc` structure described in Section 5.1.1.1 [ll˙hdioc Structure], page 11.

5.2.2.8 M_TRACEOFF

The argument of this input-output control is a pointer to the `ll_hdioc` structure described in Section 5.1.1.1 [ll˙hdioc Structure], page 11.

5.2.2.9 M_GETGSTATS

The argument of this input-output control is a pointer to the `lapb_gstioc`, `llc2_gstioc` or `mlp_gstioc` structure.[12]

5.2.2.10 M_ZEROGSTATS

The argument of this input-output control is a pointer to the `lapb_gstioc`, `llc2_gstioc` or `mlp_gstioc` structure.[13]

[12] These structures are described in Section 5.1.5.1 [lapb˙gstioc Structure], page 18, Section 5.1.5.2 [llc2˙gstioc Structure], page 18, and Section 5.1.5.3 [mlp˙gstioc Structure], page 19.

[13] These structures are described in Section 5.1.5.1 [lapb˙gstioc Structure], page 18, Section 5.1.5.2 [llc2˙gstioc Structure], page 18, and Section 5.1.5.3 [mlp˙gstioc Structure], page 19.

6 LLI Header Files

6.1 ll_control.h Header File

6.2 ll_proto.h Header File

6.3 mlp_control.h Header File

6.4 mlp_proto.h Header File

7 LLI Drivers and Modules

7.1 hdlc Driver

The '`hdlc`' driver is a pseudo-device *STREAMS* driver that provides raw HDLC framing as specified in *ISO/IEC 3309* described in reference [ISO3309], page 43.

The '`hdlc`' driver provides a *Data Link Provider Interface* as its upper multiplex service interface, as described in [DLPI], page 43.

The '`hdlc`' driver is documented in the `hdlc(4)` manual page.

7.2 lapb Driver

The '`lapb`' driver is a pseudo-device *STREAMS* driver that provides the X.25 compatible LAPB procedures as specified in *ISO/IEC 7776* described in reference [ISO7776], page 43. This driver, in combination with the '`x25`' multiplexing driver, provides X.25 over LAPB as described in reference [ISO8208], page 43.

The '`lapb`' driver provides a *Data Link Provider Interface* as its upper multiplex service interface, as described in [DLPI], page 43.

The '`lapb`' driver is documented in the `lapb(4)` manual page.

7.3 llc1 Driver

The '`llc1`' driver is a pseudo-device *STREAMS* driver that provides the IEEE 802.2 LLC Type 1 (LLC1) procedures as specified in *ISO/IEC 8802-2* described in reference [ISO8802-2], page 44. This driver, in combination with the '`x25`' multiplexing driver, provides X.25 over LLC1 as described in refeference [ISO8881], page 44.

The '`llc1`' driver provides a *Data Link Provider Interface* as its upper multiplex service interface, as described in [DLPI], page 43.

The '`llc1`' driver is documented in the `llc1(4)` manual page.

7.4 llc2 Driver

The '`llc1`' driver is a pseudo-device *STREAMS* driver that provides the IEEE 802.2 LLC Type 2 (LLC2) procedures as specified in *ISO/IEC 8802-2* described in reference [ISO8802-2], page 44. This driver, in combination with the '`x25`' multiplexing driver, provides X.25 over LLC2 as described in refeference [ISO8881], page 44.

The '`llc2`' driver provides a *Data Link Provider Interface* as its upper multiplex service interface, as described in [DLPI], page 43.

The '`llc2`' driver is documented in the `llc2(4)` manual page.

7.5 s_llic Module

The '`s_llic`' module is a pushable *STREAMS* module that converts between the LLI input-output controls described in this manual and the DLPI input-output controls used by the OpenSS7 Project. This particular module does not convert primitives, as both the LLI and the DLPI use the primitives of the *Data Link Provider Interface, Revision 2.0.0* described in reference [DLPI], page 43.

The '`s_llic`' module provides a *Data Link Provider Interface* as its upper and lower service interface, as described in [DLPI], page 43.

The '`s_llic`' module is documented in the `s_llic(4)` manual page.

7.6 s_lli Module

The '`s_lli`' module is a pushable *STREAMS* module that converts between the LLI input-output controls described in this manual and the DLPI input-output controls used by the OpenSS7 Project. This particular module converts between the LLI primitives described in this manual and the primitives of the *Data Link Provider Interface*, described in reference [DLPI], page 43.

The '`s_llic`' module provides a *Data Link Provider Interface* as its upper and lower service interface, as described in [DLPI], page 43.

The '`s_lli`' module is documented in the `s_lli(4)` manual page.

8 LLI Utilities

8.1 lltune Utility

The `lltune` utility is documented in the `lltune(8)` manual page.

8.2 mlptune Utility

The `mlptune` utility is documented in the `mlptune(8)` manual page.

8.3 linkadd Utility

The `linkadd` utility is documented in the `linkadd(8)` manual page.

8.4 linkdel Utility

The `linkdel` utility is documented in the `linkdel(8)` manual page.

8.5 linklist Utility

The `linklist` utility is documented in the `linklist(8)` manual page.

8.6 linkreset Utility

The `linkreset` utility is documented in the `linkreset(8)` manual page.

8.7 linkstart Utility

The `linkstart` utility is documented in the `linkstart(8)` manual page.

8.8 linkstate Utility

The `linkstate` utility is documented in the `linkstate(8)` manual page.

8.9 linkstop Utility

The `linkstop` utility is documented in the `linkstop(8)` manual page.

9 LLI File Formats

9.1 lapbtemplate File Format

The '`lapbtemplate`' file format is documented in the `lapbtemplate(5)` manual page.

Name

'`lapbtemplate`' — Link Access Protocol (Balanced) File Format

Description

The '`lapbtemplate`' describes the file format for input to the `lltune(8)` command for LAPB class subnetworks. The file format consists of a number of parameter values, one per line, formatted as described below. Each parameter value is described using its line number in the file, a parameter name, and a description of the format of the value. Only the value appears in the file, each value on a line by itself, one value per line.

Each of the LAPB configuration parameters correspons to the member and values of the `lapb_tune` structure, that is carried in a `lapb_tnioc` structure by the `L_LAPBTUNE` input-output control.

These protocol parameters, and the default values that exist when tuning has not been applied to a newly created LAPB subnetwork, correspond directly to the protocol parameters and defaults in *ISO/IEC 7776*, *ITU-T Rec. X.25* and *X.75*.

Format

The LAPB template consists of 16 to 18 lines containing the following configuration information:

1. `N2_VAL` is the maximum number of times that a protocol data unit (PDU) is set following the expiry of the acknowlegement timer, the P-bit timer, or the reject timer. It also limits the number of times an RR with the P-bit set is sent when remote busy is true and the busy timer expires.
2. `T1_VAL` is the time during which the LAPB expects to receive an acknowledgement to an outstanding I-PDU or an expected response to a sent UI-PDU. The value is in units of 0.1 seconds (deciseconds).
3. `TPF_VAL` is the time during which the LAPB expectes to receive a PDU with the F-bit set to 1 in response to a command with the P-bit set to 1. The value should be less than the acknowledgement timer. The value is in units of 0.1 seconds (deciseconds).
4. `TREJ_VAL` is the time interval during which the LAPB expects to receive a reply to a sent REJ DPU. The value is in units of 0.1 seconds (deciseconds).
5. `TBUSY_VAL` is the time interval during which the LAPB waits for an indication of the clearance of a busy condition at the other LAPB. The value is in units of 0.1 seconds (deciseconds).
6. `IDLE_VAL` is the time interval during which the LAPB expects to receive a PDU from the other LAPB. If it expires then the P/F cycle is intiated which may result in link disconnection. The value is in units of 0.1 seconds (decicseconds).

7. `ACK_DELAY` is the maximum delay in 0.1 second units before transmitting a delayed RR. This must be considerably less that the acknowledgement timer value, `T1_VAL`.
8. `NOTACK_MAX` is the maximuum number of unacknowledged receive I PDUs before the RR acknowledging them all must be sent.
9. `LOC_WIND` is the number of unacknowledged I PDUs that may be sent.
10. `LOC_PROBE` is the position before the window is closed at which an I PDU is sent with the P-bit set to solicit an acknowledgement from the receiver.
11. `MAX_I_LEN` is the maximum isze of a LAPB I-frame. LAPB requires all incoming I-frames above a certain size to be rejected by a FRMR. This parameter specifies the maximum size. It is constructured as the sum of the maximum X.25 data size, the X.25 protocol length and the LAPB protocol length.
12. `IGN_UA_ERROR` define whether or not to ignore any UA frams received, when the connection is in ERROR state. The value is '`1`' for *true* and '`0`' for *false*. The defalt value is *false*.
13. `FRMR_FRMR_ERROR` defines whether or not to re-transmit a frame reject if a frame reject is received, when the connection is in `ERROR` state. The value is '`1`' for *true* and '`0`' for *false*. The default value is *false*.
14. `FRMR_INVRSP_ERROR` defines whether or not to transmit a frame reject if an invalid frame response is received, when the connection is in `ERROR` state. The value is '`1`' for *true* and '`0`' for *false*. The default value is *false*.
15. `SFRAME_PBIT` defines whether or not to send a frame reject if an S-frame is received without the P-bit set. The value is '`1`' for *true*a nd '`0`' for *false*. The default value is *false*.
16. `NO_DM_ADM` defines whether or not to send a `DM` on entry to `ADM` state after an N2 count expiry. The value is '`1`' for *true* and '`0`' for *false*. The default value is *false*.

The following two fields are optional extensions:

17. `IGN_DM_ERROR` defines whether or not to ignore DM frames received, when the connection is in `ERROR` state. The value is '`1`' for *true* and '`0`' for *false*. The default value is false.
18. `SABM_IN_X32` defines the action to take when a SABM is received in X.32 setup. The value is '`1`' for *true* and '`0`' for *false*. The default value is *false*.

The last two fields ('`17`' and '`18`') are enhancements.

Files

Files following this format are normally kept in the '`/etc/sysconfig/strx25/template/`' directory.[1]

[1] Note that this directory varies depending on whether the build was on a `dpkg(1)`-based or `rpm(1)`-based system.

See Also

- `lltune(8)`
- `lapb(4)`
- `x25netd(8)`

Compatibility

The '`lapbtemplate`' file format is compatible with *Spider X.25*, and implementations based on *Spider X.25*, such as *AIXlink/X.25*, *HP-UX*, *IRIS SX.25*, *Solstice X.25*, *PT X.25*, *SBE X.25*, with the following compatibility considerations:

— Most implementations only define the first 16 lines. This implementation defines 18 lines, where the first 16 lines are compatible with other implementations and the last additional two lines are optional.

— *PT X.25* documents the `SABM_IN_X32` LAPB template field but not the `IGN_DM_ERROR` LAPB template field. *Solstice X.25* and *IRIS SX.25* do not document either the `IGN_DM_ERROR` nor `SABM_IN_X32` LAPB template fields.

For additional compatiblity infromation see, `lapb(4)`, and `STREAMS(9)`.

Conformance

AIXlink/X.25, *HP-UX*, *IRIS SX.25*, *Solstice X.25*, *PT X.25*, *SBE X.25*, documentation.

History

The '`lapbtemplate`' file format first appeared in *Spider X.25*.

9.2 llc2template File Format

The 'llc2template' file format is documented in the llc2template(5) manual page.

Name

'llc2template' — Logical Link Control Type 2 File Format

Description

The 'llc2template' describes the file format for input to the lltune(8) command for LLC2 class subnetworks. The file format consists of a number of parameter values, one per line, formatted as described below. Each parameter value is described using its line number in the file, a parameter name, and a description of the format of the value. Only the value appears in the file, each value on a line by itself, one value per line.

Each of the LLC2 configuration parameters corresponds to the member and values of the llc2_tune structure, that is carried in a llc2_tnioc structure by the L_LLC2TUNE input-output control.

These protocol parameters, and the default values that exist when tuning has not been applied to a newly created LLC2 subnetwork, correspond directly to the protocol parameters and defaults in *ISO/IEC 8802-2:1998*.

Format

The LLC2 template consists of 14 lines containing the following configuration information.

1. N2_VAL is the maximum number of times that a Protocol Data Unit (PDU) is sent following the expiry of the acknowledgement timer, the P-bit timer, or the reject timer. This parameter also limits the number of times an RR is sent with the P-bit set when remote busy is true and the busy timer expires.
2. T1_VAL is the time interval during which the LLC2 expects to receive an acknowledgement to an outstanding I-PDU or an expected response to a sent UI-PDU. The value is in units of 0.1 seconds.
3. TPF_VAL is the time during which the LLC2 expects to receive a PDU with the F-bit set to 1 in response to a command with the P-bit set to 1. The value should be less than that specified for the akcnowledgement timer. The value is in units of 0.1 seconds.
4. TREJ_VAL is the time interval during which the LLC2 expects to receive a reply to a sent REJ PDU. The value is in units of 0.1 seconds.
5. TBUSY_VAL is the timer interval during which the LLC2 waits for an indication of the clearance of busy condition at the other LLC2. The value is in units of 0.1 seconds.
6. TIDLE_VAL is the time interval during which the LLC2 expects to receive a PDU from the other LLC2. The value is in units of 0.1 seconds.
7. ACK_DELAY is the RR delay time. This is the time interval for which the LLC2 will withhold acknowledgements of unacknowledged received I-PDUs. The value is in units of 0.1 seconds.
8. NOTACK_MAX is the maximum number of unacknowledged received I-frames.
9. TX_WINDOW is the transmit window (if no XID received).

10. `TX_PROBE` is the position before the window is closed at which an I-PDU is sent with the P-bit set to solicit an acknowledgement from the receiver.
11. `MAX_I_LEN` is the maximum size of an LLC2 I-frame. LLC2 requires all incoming I-frames above a certain size to be rejected by a FRMR. This parameter specifies the maximum size of data that may bereceived starint from the LLC2 protocol header.

    In an X.25 network, it is constructed as (maximum X.25 data length + X.25 protool header length + LLC2 protocol header length). In an SNA network, it is constructed as (maximum SNA data length + SNA request header length + SNA transmission header length + LLC2 protocol header length).
12. `XID_WINDOW` is the XID window size (receive window), when the remote window size is unknown or zero.
13. `XID_NDUP` is the duplicate MAC XID count (0 means no test).
14. `XID_TDUP` is the duplicate MAC XID time. The value is in units of 0.1 seconds.

Files

Files following this format are normally kept in the '`/etc/sysconfig/strx25/template/`' directory.[1]

See Also

- `lltune(8)`
- `llc2(4)`
- `x25netd(8)`

Compatibility

The '`llc2template`' file format is compatible with *Spider X.25*, and implementations based on *Spider X.25*, such as *AIXlink/X.25*, *HP-UX*, *IRIS SX.25*, *Solstice X.25*, *PT X.25*, *SBE X.25*, with the following compatibility considerations:

— *PT X.25* does not support LLC2. *OpenSS7 X.25 Networking* supports LLC2 in support of XOL and porting applications from *AIXlink/X.25*, *Solstice X.25*, *HP-UX*, *IRIS SX.25*, *VxWorks*, *pSOS*, *SpiderX*, and many other implementations based on *SpiderX.25* support LLC2. Portable X.25 and XOL applications will use *OpenSS7 X.25 Networking* instead of *PT X.25*.

For additional compatibiltiy information see, `llc2(4)`, and `STREAMS(9)`.

Conformance

AIXlink/X.25, *HP-UX*, *IRIS SX.25*, *Solstice X.25*, *PT X.25*, *SBE X.25*, documentation.

History

The '`llc2template`' file format first appeared in *Spider X.25*.

[1] Note that this directory varies depending on whether the build was on a `dpkg(1)`-based or `rpm(1)`-based system.

9.3 mlptemplate File Format

The '`mlptemplate`' file format is documented in the `mlptemplate(5)` manual page.

Appendix A LLI Compatibility and Porting

A.1 Compatibility with AIXlink/X.25

A.2 Compatibility with HP X.25/9000

A.3 Compatibility with IRIS SX.25

A.4 Compatibility with PT X.25

A.5 Compatibility with RadiSys WAN

A.6 Compatibility with SBE X.25

A.7 Compatibility with Solstice X.25

Glossary of LLI Terms and Acronyms

ANSI	American National Standards Institute
CCITT	Old name for ITU-T
CONS	Connection-Oriented Network Service
ENSDU	Expedited Network Service Data Unit
ETSI	European Telecommunications Standards Institute
IEEE	Institute of Electrical and Electronics Engineers
ITU	International Telecommunications Union
ITU-T	ITU Telecom Sector
LCI	Logical Channel Identifier
LLC1	Logical Link Control Type 1
LLC2	Logical Link Control Type 2
LLC	Logical Link Control
MAC	Media Access Control
NLI	Network Layer Interface
NPDU	Network Protocol Data Unit
NSAP	Network Service Access Point
NSDU	Network Service Data Unit
NSP	Network Service Provider
NS	Network Service
NSU	Network Service User
PDU	Protocol Data Unit
PVC	Permanent Virtual Circuit
SAP	Service Access Point
SDU	Service Data Unit
VC	Virtual Circuit
X.121	ITU-T Recommendation X.121
X.25	ITU-T Recommendation X.25
X.29	ITU-T Recommendation X.29

References

[AIXlink/X.25] *AIXlink/X.25 Version 2.1 for AIX: Guide and Reference*, No: SC23-2520-07, Eighth Edition, September 2006, (Bolder, CO), International Business Machine Corp., IBM. IBM Documentation Library.

[ARTIC WAN] *ARTIC STREAMS Support WAN Driver Interface Reference*, Release 1.7, June 2004, (Hillsboro, OR), RadiSys Corporation, RadiSys. [Doc No: 007-01232-0003], RadiSys Support Documentation.

[CDI] *OpenSS7 CAE Specification: Communications Device Interface (CDI) Specification*, Revision 0.9.2, Draft 2, July 15, 2007, (Edmonton, Canada), B. Bidulock, OpenSS7 Corporation. Distributed with package '`strxns-0.9.2`' and '`openss7-0.9.2`'. OpenSS7 Documents.

[DLPI] *Open Group CAE Specification: Data Link Provider Interface (DLPI) Specification*, Revision 2.0.0, Draft 2, August 20, 1992, (Parsippany, New Jersey), UNIX International, Inc., UNIX International Press. The Open Group, The OpenSS7 Project.

[IRIS SX.25] *IRIS SX.25 NLI Programmer's Guide*, 1995, (Mountainview, CA), Silicon Graphics, Inc., SGI Technical Publications. [No: 007-2268-002]. SGI Technical Publications.

[ISO3309] **ISO/IEC 3309:1988**, *Information Technology – Data Communications – High-Level Data Link Control procedures – Frame Structure*, 1988, ISO/IEC, International Orgianization for Standardization. International Organization for Standardization.

[ISO8208] **ISO/IEC 8208:2000**, *Information Technology – Data Communications – X.25 Packet Layer Protocol for Data Terminal Equipment*, Fourth Edition, November 1, 2000, ISO/IEC, International Organization for Standardization. International Organization for Standardization.

[ISO7776] **ISO/IEC 7776:1995**, *Information technology – Telecommunications information exchange between systems – High-level data link control procedures – Description of the X.25 LAPB-compatible DTE data link procedures*, Second Edition, July 1, 1995, International Organization for Standardization. International Organization for Standardization.

[ISO8802-2] **ANSI/IEEE Standard 802.2-1998 [ISO/IEC 8802-2:1998]**, *IEEE Standard for Information Technology – Telecommunications and Information Exchange Between Systems – Local and Metropolitan Area Networks – Specific Requirements – Part 2: Logical Link Control*, May 7, 1998, (New York), ANSI/IEEE, IEEE Computer Society. [ISBN 1-55937-959-6]. Institute of Electrical and Electronics Engineers.

[ISO8881] **ISO/IEC 8881:1989**, *Information Processing Systems – Data Communications – User of the X.25 Packet Level Protocol in Local Area Networks*, 1989, ISO/IEC, International Organization for Standardization. International Organization for Standardization.

[NLI] *OpenSS7 CAE Specification: Network Layer Interface (NLI) Specification*, Revision 0.9.2, Draft 1, June 2008, (Edmonton, AB), B. Bidulock, OpenSS7 Corporation. Distributed with package '`strx25-0.9.2`' and '`openss7-0.9.2`'. The OpenSS7 Project.

[NPI] *Open Group CAE Specification: Network Provider Interface (NPI) Specification*, Revision 2.0.0, Draft 2, August 17, 1992, (Parisppany, New Jersey), UNIX International, Inc., UNIX International Press. The OpenSS7 Project.

[Solstice X.25] *Solstice X.25 9.2 Administration Guide*, October 1999, (Palo Alto, CA), Sun Microsystems, Inc., Sun. [Part No: 806-1234-10], Solaris Documentation.

[TPI] *Open Group CAE Specification: Transport Provider Interface (TPI) Specification*, Revision 2.0.0, Draft 2, 1999, (Berkshire, UK), Open Group, Open Group Publication. The Open Group, The OpenSS7 Project.

[V.25 bis] **ITU-T Recommendation V.25 bis (10/96)**, *Synchronous and asynchronous automatic dialing procedrues on switched networks*, October 1996, (Geneva), ITU, ITU-T Telecommunication Standardization Sector of ITU, (Previously "CCITT Recommendation"), `http://www.itu.int/rec/T-REC-V.25bis/en/ T-REC-V.25bis`.

[X.21] **ITU-T Recommendation X.21 (09/92)**, *Interface between Data Terminal Equipment (DTE) and Data Circuit-terminating Equiment (DCE) for synchronous operation on Public Data Networks*, September 1992, (Geneva), ITU, ITU-T Telecommunication Standardization Sector of ITU. (Previously "CCITT Recommendation"), T-REC-X.21.

[X.21 bis] **ITU-T Recommendation X.21 bis (03/88)**, *Use on Public Data Networks of Data Terminal Equipment (DTE) which is designed for interfacing to synchronous V-series modems*, March 1988, (Geneva), ITU, ITU-T Telecommunication Standardization Sector of ITU. (Previously "CCITT Recommendation"), T-REC-X.21bis.

[X.25] **ITU-T Recommendation X.25**. T-REC-X.25.

[X.29] **ITU-T Recommendation X.29**. T-REC-X.29.

[XX25] *X/Open CAE Specification: X.25 Programming Interface using XTI (XX25)*, No. c411, November 1995, (Berkshire, UK), X/Open, Open Group Publication. [ISBN: 1-85912-136-5]. The Open Group.

Index

M

N

P

R

S

T

U

X