Signalling Data Link Interface (SDLI) Specification

Signalling Data Link Interface (SDLI) Specification

Version 1.1 Edition 7.20141001
Updated October 25, 2014
Distributed with Package openss7-1.1.7.20141001

Abstract:

This document is a Specification containing technical details concerning the implementation of the Signalling Data Link Interface (SDLI) for OpenSS7. It contains recommendations on software architecture as well as platform and system applicability of the Signalling Data Link Interface (SDLI). It provides abstraction of the Signalling Data Link (SDL) interface to these components as well as providing a basis for Signalling Data Link control for other Signalling Data Link 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:

OpenSS7 Corporation disclaims all warranties with regard to this documentation including all implied warranties of merchantability, fitness for a particular purpose, non-infringement, or title; that the contents of the document are suitable for any purpose, or that the implementation of such contents will not infringe on any third party patents, copyrights, trademarks or other rights. In no event shall OpenSS7 Corporation be liable for any direct, indirect, special or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with any use of this document or the performance or implementation of the contents thereof.

Short Contents

Table of Contents

List of Figures

List of Tables

Preface

Notice

Abstract

This document is a Specification containing technical details concerning the implementation of the Signalling Data Link Interface (SDLI) for OpenSS7. It contains recommendations on software architecture as well as platform and system applicability of the Signalling Data Link Interface (SDLI).

This document specifies a Signalling Data Link Interface (SDLI) Specification in support of the OpenSS7 Signalling Data Link (SDL) protocol stacks. It provides abstraction of the Signalling Data Link interface to these components as well as providing a basis for Signalling Data Link control for other Signalling Data Link protocols.

Purpose

The purpose of this document is to provide technical documentation of the Signalling Data Link Interface (SDLI). This document is intended to be included with the OpenSS7 STREAMS software package released by *OpenSS7 Corporation*. It is intended to assist software developers, maintainers and users of the Signalling Data Link Interface (SDLI) with understanding the software architecture and technical interfaces that are made available in the software package.

Intent

It is the intent of this document that it act as the primary source of information concerning the Signalling Data Link Interface (SDLI). This document is intended to provide information for writers of OpenSS7 Signalling Data Link Interface (SDLI) applications as well as writers of OpenSS7 Signalling Data Link Interface (SDLI) Users.

Audience

The audience for this document is software developers, maintainers and users and integrators of the Signalling Data Link Interface (SDLI). The target audience is developers and users of the OpenSS7 SS7 stack.

Revision History

Take care that you are working with a current version of this documentation: you will not be notified of updates. To ensure that you are working with a current version, check the OpenSS7 Project website for a current version.

A current version of this specification is normally distributed with the *OpenSS7* package, `openss7-1.1.7.20141001`.[1]

[1] `http://www.openss7.org/repos/tarballs/openss7-1.1.7.20141001.tar.bz2`

Version Control

Although the author has attempted to ensure that the information in this document is complete and correct, neither the Author nor OpenSS7 Corporation will take any responsibility in it. *OpenSS7 Corporation* is making this documentation available as a reference point for the industry. While *OpenSS7 Corporation* believes that these interfaces are well defined in this release of the document, minor changes may be made prior to products conforming to the interfaces being made available. *OpenSS7 Corporation* reserves the right to revise this software and documentation for any reason, including but not limited to, conformity with standards promulgated by various agencies, utilization of advances in the state of the technical arts, or the reflection of changes in the design of any techniques, or procedures embodied, described, or referred to herein. *OpenSS7 Corporation* is under no obligation to provide any feature listed herein.

```
$Log: sdli.texi,v $
Revision 1.1.2.2  2011-02-07 02:21:43  brian
- updated manuals

Revision 1.1.2.1  2009-06-21 10:56:20  brian
- added files to new distro
```

ISO 9000 Compliance

Only the TEX, texinfo, or roff source for this maual is controlled. An opaque (printed, postscript or portable document format) version of this manual is a **UNCONTROLLED VERSION**.

Disclaimer

OpenSS7 Corporation **disclaims all warranties with regard to this documentation including all implied warranties of merchantability, fitness for a particular purpose, non-infrincement, or title; that the contents of the manual are suitable for any purpose, or that the implementation of such contents will not infringe on any third party patents, copyrights, trademarks or other rights. In no event shall** *OpenSS7 Corporation* **be liable for any direct, indirect, special or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action or contract, negligence or other tortious action, arising out of or in connection with any use of this documentation or the performance or implementation of the contents thereof.**

U.S. Government Restricted Rights

If you are licensing this Software on behalf of the U.S. Government ("Government"), the following provisions apply to you. If the Software is supplied by the Department of Defense ("DoD"), it is classified as "Commercial Computer Software" under paragraph 252.227-7014 of the DoD Supplement to the Federal Aquisition Regulations ("DFARS") (or any successor regulations) and the Government is acquiring only the license rights granded herein (the license rights customarily provided to non-Government users). If the Software is supplied to any unit or agency of the Government other than DoD, it is classified as "Restricted Computer Software" and the Government's rights in the Software are defined in paragraph 52.227-19 of the Federal Acquisition Regulations ("FAR") (or any successor regulations) or, in the cases of NASA, in paragraph 18.52.227-86 of the NASA Supplerment to the FAR (or any successor regulations).

Acknowledgements

The OpenSS7 Project was funded in part by:

- Monavacon Limited
- OpenSS7 Corporation

Thanks to the subscribers to and sponsors of The OpenSS7 Project. Without their support, open software like this would not be possible.

As with most open source projects, this project would not have been possible without the valiant efforts and productive software of the Free Software Foundation, the Linux Kernel Community, and the open source software movement at large.

1 Introduction

This document specifies a STREAMS-based kernel-level instantiation of the Signalling Data Link Interface (SDLI) definition. The Signalling Data Link Interface (SDLI) enables the user of a signalling data link service to access and use any of a variety of conforming signalling data link providers without specific knowledge of the provider's protocol. The service interface is designed to support any network signalling data link protocol and user signalling data link protocol. This interface only specifies access to signalling data link service providers, and does not address issues concerning signalling data link management, protocol performance, and performance analysis tools.

This specification assumes that the reader is familiar with ITU-T state machines and signalling data link interfaces (e.g. Q.703, Q.2210), and STREAMS.

1.1 Related Documentation

— **ITU-T Recommendation Q.703 (White Book)**

— **ITU-T Recommendation Q.2210 (White Book)**

— **ANSI T1.111.3/2002**

— **System V Interface Definition, Issue 2 - Volume 3**

1.1.1 Role

This document specifies an interface that supports the services provided by the *Signalling System No. 7 (SS7)* for ITU-T, ANSI and ETSI applications as described in ITU-T Recommendation Q.703, ITU-T Recommendation Q.2210, ANSI T1.111.3, ETSI ETS 300 008-1. These specifications are targeted for use by developers and testers of protocol modules that require signalling data link service.

1.2 Definitions, Acronyms, Abbreviations

LM Local Management.

LMS Local Management Service.

LMS User A user of Local Management Services.

LMS Provider
A provider of Local Management Services.

Originating SDL User
A SDL-User that initiates a Signalling Data Link.

Destination SDL User
A SDL-User with whom an originating SDL user wishes to establish a Signalling Data Link.

ISO International Organization for Standardization

SDL User Kernel level protocol or user level application that is accessing the services of the Signalling Data Link sub-layer.

SDL Provider
Signalling Data Link sub-layer entity/entities that provide/s the services of the Signalling Data Link interface.

SDLI	Signalling Data Link Interface
TIDU	Signalling Data Link Interface Data Unit
TSDU	Signalling Data Link Service Data Unit
OSI	Open Systems Interconnection
QOS	Quality of Service
STREAMS	A communication services development facility first available with UNIX System V Release 3.

2 The Signalling Data Link Layer

The Signalling Data Link Layer provides the means to manage the association of SDL-Users into connections. It is responsible for the routing and management of data to and from signalling data link connections between SDL-user entities.

2.1 Model of the SDLI

The SDLI defines the services provided by the signalling data link layer to the signalling data link user at the boundary between the signalling data link provider and the signalling data link user entity. The interface consists of a set of primitives defined as STREAMS messages that provide access to the signalling data link layer services, and are transferred between the SDLS user entity and the SDLS provider. These primitives are of two types; ones that originate from the SDLS user, and others that originate from the SDLS provider. The primitives that originate from the SDLS user make requests to the SDLS provider, or respond to an indication of an event of the SDLS provider. The primitives that originate from the SDLS provider are either confirmations of a request or are indications to the SDLS user that an event has occurred. Figure 2.1 shows the model of the SDLI.

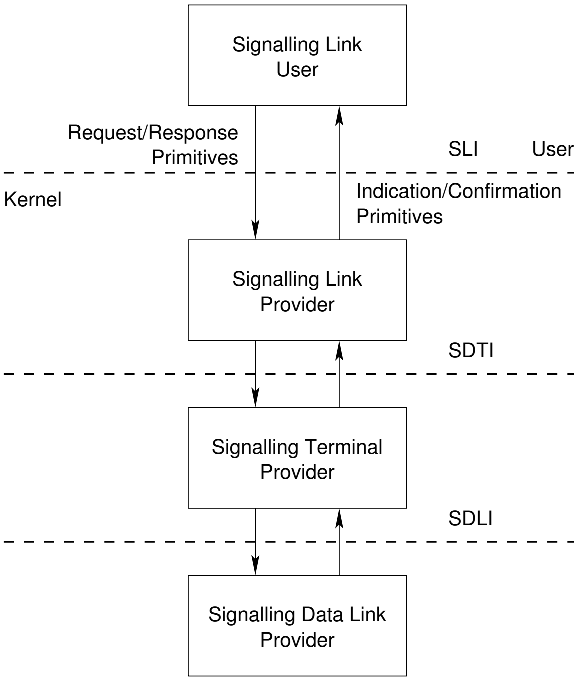
Signalling Link User
Request/Response Primitives
SLI
User
Kernel
Indication/Confirmation Primitives
Signalling Link Provider
SDTI
Signalling Terminal Provider
SDLI
Signalling Data Link Provider

Figure 2.1: *Model of the SDLI*

The SDLI allows the SDLS provider to be configured with any signalling data link layer user (such as a signalling data terminal application) that also conforms to the SDLI. A signalling data link layer user can also be a user program that conforms to the SDLI and accesses the SDLS provider via **putmsg**(2s) and **getmsg**(2s) system calls. The typical configuration, however, is to place a signalling data terminal module above the signalling data link layer.

2.2 Services

The features of the SDLI are defined in terms of the services provided by the SDLS provider, and the individual primitives that may flow between the SDLS user and the SDLS provider.

The SDLI Services are broken into two groups: local management services and protocol services. Local management services are responsible for the local management of Streams, assignment of Streams to physical points of attachment, enabling and disabling of Streams, management of options associated with a Stream, and general acknowledgement and event reporting for the Stream. Protocol services consist of connecting a Stream to a medium, exchanging bits with the medium, and disconnecting the Stream from the medium.

2.2.1 Local Management

Local management services are listed in Table 2.1.

Phase	Service	Primitives
Local Management	Acknowledgement	LMI_OK_ACK, LMI_ERROR_ACK
	Information Reporting	LMI_INFO_REQ, LMI_INFO_ACK
	PPA Attachment	LMI_ATTACH_REQ, LMI_DETACH_REQ, LMI_OK_ACK
	Initialization	LMI_ENABLE_REQ, LMI_ENABLE_CON, LMI_DISABLE_REQ, LMI_DISABLE_CON
	Options Management	LMI_OPTMGMT_REQ, LMI_OPTMGMT_ACK
	Event Reporting	LMI_ERROR_IND, LMI_STATS_IND, LMI_EVENT_IND

Table 2.1: *Local Management Services*

The local management services interface is described in Section 3.1 [Local Management Services], page 13, and the primitives are detailed in Section 4.1 [Local Management Service Primitives], page 23. The local management services interface is defined by the `ss7/lmi.h` header file (see Section A.1 [LMI Header File Listing], page 77).

2.2.2 Protocol

Protocol services are listed in Table 2.2.

Phase	Service	Primitives
Protocol	Connection	SDL_CONNECT_REQ
	Data Transfer	SDL_BITS_FOR_TRANSMISSION_REQ, SDL_RECEIVED_BITS_IND
	Disconnection	SDL_DISCONNECT_REQ, SDL_DISCONNECT_IND

Table 2.2: *Protocol Services*

The protocol services interface is described in Section 3.2 [Protocol Services], page 19, and the primitives are detailed in Section 4.2 [Protocol Service Primitives], page 60. The protocol services interface is defined by the `ss7/sdli.h` header file (see Section A.2 [SDLI Header File Listing], page 83).

2.3 Purpose of the SDLI

The SDLI is typically implemented as a device driver controlling a TDM (Time Division Multiplexing) device that provides access to channels. The purpose behind exposing this low level interface is that almost all communications channel devices can be placed into a *raw* mode, where a bit stream can be exchanged between the driver and the medium. The SDLI provides an interface that, once implemented as a driver for a new device, can provide complete and verified SS7 signalling link capabilities by pushing generic SDT (Signalling Data Terminal) and SL (Signalling Link) modules over an open device Stream.

This allows SDT and SL modules to be verified independently for correct operation and then simply used for all manner of new device drivers that can implement the SDLI interface.

3 Services Definition

3.1 Local Management Services

3.1.1 Acknowledgement Service

The acknowledgement service provides the LMS user with the ability to receive positive and negative acknowledgements regarding the successful or unsuccessful completion of services.

- `LMI_OK_ACK`: The `LMI_OK_ACK` message is used by the LMS provider to indicate successful receipt and completion of a service primitive request that requires positive acknowledgement.
- `LMI_ERROR_ACK`: The `LMI_ERROR_ACK` message is used by the LMS provider to indicate successful receipt and failure to complete a service primitive request that requires negative acknowledgement.

A successful invocation of the acknowledgement service is illustrated in Figure 3.1.

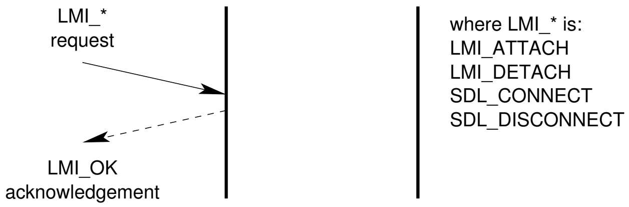

Figure 3.1: *Message Flow: Successful Acknowledgement Service*

As illustrated in Figure 3.1, the service primitives for which a positive acknowledgement may be returned are the `LMI_ATTACH_REQ` and `LMI_DETACH_REQ`.

An unsuccessful invocation of the acknowledgement service is illustrated in Figure 3.2.

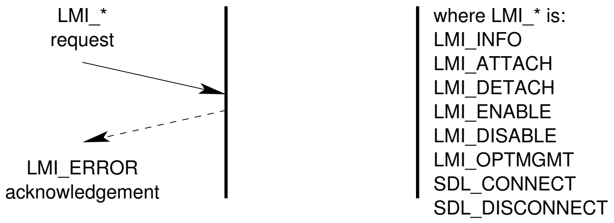

Figure 3.2: *Message Flow: Unsuccessful Acknowledgement Service*

As illustrated in Figure 3.2, the service primitives for which a negative acknowledgement may be returned are the `LMI_INFO_REQ`, `LMI_ATTACH_REQ`, `LMI_DETACH_REQ`, `LMI_ENABLE_REQ`, `LMI_DISABLE_REQ` and `LMI_OPTMGMT_REQ` messages.

3.1.2 Information Reporting Service

The information reporting service provides the LMS user with the ability to elicit information from the LMS provider.

- `LMI_INFO_REQ`: The `LMI_INFO_REQ` message is used by the LMS user to request information about the LMS provider.
- `LMI_INFO_ACK`: The `LMI_INFO_ACK` message is issued by the LMS provider to provide requested information about the LMS provider.

A successful invocation of the information reporting service is illustrated in Figure 3.3.

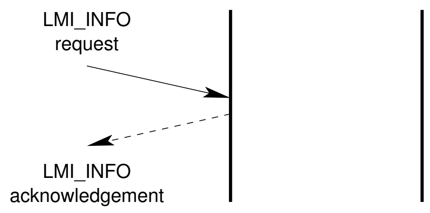

Figure 3.3: *Message Flow: Successful Information Reporting Service*

3.1.3 Physical Point of Attachment Service

The local management interface provides the LMS user with the ability to associate a Stream to a physical point of appearance (*PPA*) or to disassociate a Stream from a PPA. The local management interface provides for two styles of LMS provider:

Style 1 LMS Provider

A *Style 1* LMS provider is a provider that associates a Stream with a PPA at the time of the first `open(2s)` call for the device, and disassociates a Stream from a PPA at the time of the last `close(2s)` call for the device.

Physical points of attachment (PPA) are assigned to major and minor device number combinations. When the major and minor device number combination is opened, the opened Stream is automatically associated with the PPA for the major and minor device number combination. The last close of the device disassociates the PPA from the Stream.

Freshly opened *Style 1* LMS provider Streams start life in the `LMI_DISABLED` state.

This approach is suitable for LMS providers implemented as real or pseudo-device drivers and is applicable when the number of minor devices is small and static.

Style 2 LMS Provider

A *Style 2* LMS provider is a provider that associates a Stream with a PPA at the time that the LMS user issues the `LMI_ATTACH_REQ` message. Freshly opened Streams are not associated with any PPA. The *Style 2* LMS provider Stream is disassociated from a PPA when the Stream is closed or when the LMS user issues the `LMI_DETACH_REQ` message.

Freshly opened *Style 2* LMS provider Streams start life in the `LMI_UNATTACHED` state.

This approach is suitable for LMS providers implemented as clone real or pseudo-device drivers and is applicable when the number of minor devices is large or dynamic.

3.1.3.1 PPA Attachment Service

The PPA attachment service provides the LMS user with the ability to attach a *Style 2* LMS provider Stream to a physical point of appearance (PPA).

- `LMI_ATTACH_REQ`: The `LMI_ATTACH_REQ` message is issued by the LMS user to request that a *Style 2* LMS provider Stream be attached to a specified physical point of appearance (PPA).

- `LMI_OK_ACK`: Upon successful receipt and processing of the `LMI_ATTACH_REQ` message, the LMS provider acknowledges the success of the service completion with a `LMI_OK_ACK` message.

- `LMI_ERROR_ACK`: Upon successful receipt but failure to process the `LMI_ATTACH_REQ` message, the LMS provider acknowledges the failure of the service completion with a `LMI_ERROR_ACK` message.

A successful invocation of the attachment service is illustrated in Figure 3.4.

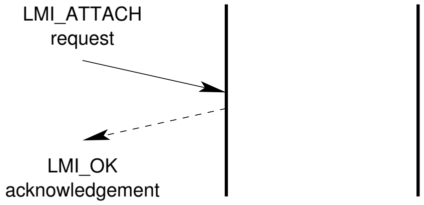

Figure 3.4: *Message Flow: Successful Attachment Service*

3.1.3.2 PPA Detachment Service

The PPA detachment service provides the LMS user with the ability to detach a *Style 2* LMS provider Stream from a physical point of attachment (PPA).

- `LMI_DETACH_REQ`: The `LMI_DETACH_REQ` message is issued by the LMS user to request that a *Style 2* LMS provider Stream be detached from the attached physical point of appearance (PPA).

- `LMI_OK_ACK`: Upon successful receipt and processing of the `LMI_DETACH_REQ` message, the LMS provider acknowledges the success of the service completion with a `LMI_OK_ACK` message.

- `LMI_ERROR_ACK`: Upon successful receipt but failure to process the `LMI_DETACH_REQ` message, the LMS provider acknowledges the failure of the service completion with a `LMI_ERROR_ACK` message.

A successful invocation of the detachment service is illustrated in Figure 3.5.

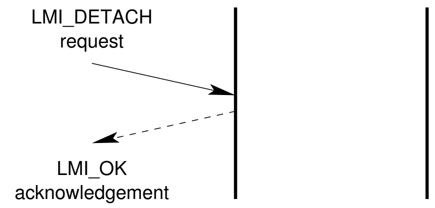

Figure 3.5: *Message Flow: Successful Detachment Service*

3.1.4 Initialization Service

The initialization service provides the LMS user with the abilty to enable and disable the Stream for the associated PPA.

3.1.4.1 Interface Enable Service

The interface enable service provides the LMS user with the ability to enable an LMS provider Stream that is associated with a PPA. Enabling the interface permits the LMS user to exchange protocol service interface messages with the LMS provider.

- `LMI_ENABLE_REQ`: The `LMI_ENABLE_REQ` message is issued by the LMS user to request that the protocol service interface be enabled.
- `LMI_ENABLE_CON`: Upon successful enabling of the protocol service interface, the LMS provider acknowledges successful completion of the service by issuing a `LMI_ENABLE_CON` message to the LMS user.
- `LMI_ERRORK_ACK`: Upon unsuccessful enabling of the protocol service interface, the LMS provider acknowledges the failure to complete the service by issuing an `LMI_ERROR_ACK` message to the LMS user.

A successful invocation of the enable service is illustrated in Figure 3.6.

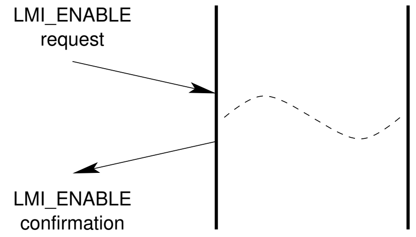

Figure 3.6: *Message Flow: Successful Enable Service*

3.1.4.2 Interface Disable Service

The interface disable service provides the LMS user with the ability to disable an LMS provider Stream that is associated with a PPA. Disabling the interface withdraws the LMS user's ability to exchange protocol service interface messages with the LMS provider.

- `LMI_DISABLE_REQ`: The `LMI_DISABLE_REQ` message is issued by the LMS user to request that the protocol service interface be disabled.

- `LMI_DISABLE_CON`: Upon successful disabling of the protocol service interface, the LMS provider acknowledges successful completion of the service by issuing a `LMI_DISABLE_CON` message to the LMS user.

- `LMI_ERRORK_ACK`: Upon unsuccessful disabling of the protocol service interface, the LMS provider acknowledges the failure to complete the service by issuing an `LMI_ERROR_ACK` message to the LMS user.

A successful invocation of the disable service is illustrated in Figure 3.7.

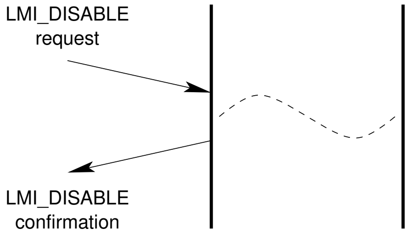

Figure 3.7: *Message Flow: Successful Disable Service*

3.1.5 Options Management Service

The options management service provides the LMS user with the ability to control and affect various generic and provider-specific options associated with the LMS provider.

- `LMI_OPTMGMT_REQ`: The LMS user issues a `LMI_OPTMGMT_REQ` message when it wishes to interrogate or affect the setting of various generic or provider-specific options associated with the LMS provider for the Stream upon which the message is issued.

- `LMI_OPTMGMT_ACK`: Upon successful receipt of the `LMI_OPTMGMT_REQ` message, and successful options processing, the LMS provider acknowledges the successful completion of the service with an `LMI_OPTMGMT_ACK` message.

- `LMI_ERROR_ACK`: Upon successful receipt of the `LMI_OPTMGMT_REQ` message, and unsuccessful options processing, the LMS provider acknowledges the failure to complete the service by issuing an `LMI_ERROR_ACK` message to the LMS user.

A successful invocation of the options management service is illustrated in Figure 3.8.

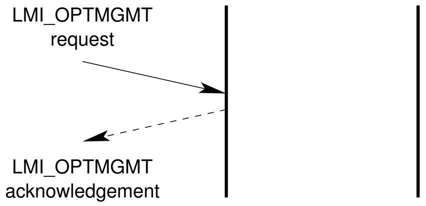

Figure 3.8: *Message Flow: Successful Options Management Service*

3.1.6 Error Reporting Service

The error reporting service provides the LMS provider with the ability to indicate asynchronous errors to the LMS user.

- `LMI_ERROR_IND`: The LMS provider issues the `LMI_ERROR_IND` message to the LMS user when it needs to indicate an asynchronous error (such as the unusability of the communications medium).

A successful invocation of the error reporting service is illustrated in Figure 3.9.

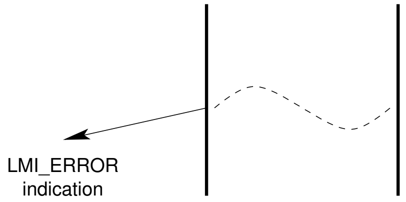

Figure 3.9: *Message Flow: Successful Error Reporting Service*

3.1.7 Statistics Reporting Service

- `LMI_STATS_IND`:

A successful invocation of the statistics reporting service is illustrated in Figure 3.10.

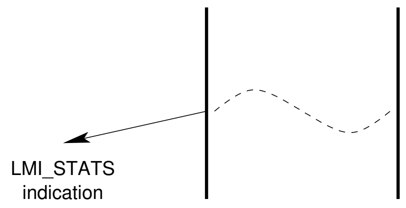

Figure 3.10: *Message Flow: Successful Statistics Reporting Service*

3.1.8 Event Reporting Service

The event reporting service provides the LMS provider with the ability to indicate specific asynchronous management events to the LMS user.

- `LMI_EVENT_IND`: The LMS provider issues the `LMI_EVENT_IND` message to the LMS user when it wishes to indicate an asynchronous (management) event to the LMS user.

A successful invocation of the event reporting service is illustrated in Figure 3.11.

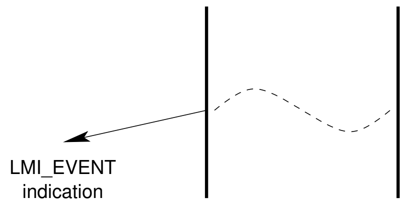

Figure 3.11: *Message Flow: Successful Event Reporting Service*

3.2 Protocol Services

Protocol services are specific to the Signalling Data Link interface. These services consist of connection services that permit the transmit and receive directions to be connected to or disconnected from the medium, and data transfer services that permit the exchange of bits between SDLS users. The service primitives that implement the protocol services are described in detail in Section 4.2 [Protocol Service Primitives], page 60.

3.2.1 Connection Service

The connection service provides the ability for the SDLS user to connect to the medium for the purpose of transmitting bits, receiving bits, or both. In SS7, this is a Level 1 function, possibly the responsibility of multiplex or digital cross-connect switch.

- `SDL_CONNECT_REQ`: The `SDL_CONNECT_REQ` message is used by the SDLS user to request that the Stream be connected to the medium. Connection to the medium might require some switching or other mechanism to prepare the Stream for data transmission and reception. Connections can be formed for the receive direction or the transmit direction independently.

A successful invocation of the connection service is illustrated in Figure 3.12.

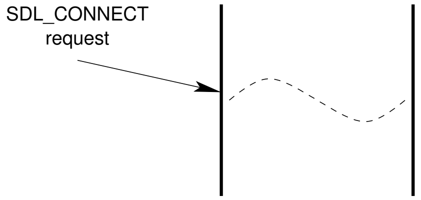

Figure 3.12: *Message Flow: Successful Connection Service*

3.2.2 Data Transfer Service

The data transfer service provides the SDLS user with the ability to request that bits be transmitted on the medium, and the SDLS provider with the ability to indicate bits that have been received from the medium.

- `SDL_BITS_FOR_TRANSMISSION_REQ`: The `SDL_BITS_FOR_TRANSMISSION_REQ` message is used by the SDLS user to place raw bits onto the medium. The Stream must have first been successfully activated in the transmit direction using the `SDL_CONNECT_REQ` message.
- `SDL_RECEIVED_BITS_IND`: The `SDL_RECEIVED_BITS_IND` message is issued by the SDLS provider when activated for the receive direction with the `SDL_CONNECT_REQ` message, to indicate bits received on the medium.

A successful invocation of the data transfer service is illustrated in Figure 3.13.

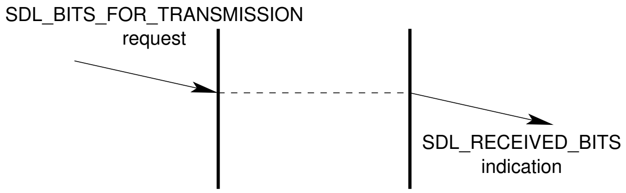

Figure 3.13: *Message Flow: Successful Data Transfer Service*

3.2.3 Disconnection Service

The disconnection service provides the ability for the SDLS user to disconnect from the medium, withdrawing from the purpose of transmitting bits, receiving bits, or both. It allows the SDLS provider to autonomously indicate that the medium has been disconnected from the Stream. In SS7, this is a Level 1 function, possibly the responsibility of a multiplex or digital cross-connect switch.

- SDL_DISCONNECT_REQ: The SDL_DISCONNECT_REQ message is used by the SDLS user to request that the Stream be disconnected from the medium. Disconnection from the medium might require some switching or other mechanism. Disconnection can be performed for the receive direction or the transmit direction independently.
- SDL_DISCONNECT_IND: The SDL_DISCONNECT_IND message is used by the SDLS provider to indicate to the SDLS user that the Stream has been disconnected from the medium. Disconnection is indicated for both the receive and transmit directions.

A successful invocation of the disconnection service by the SDLS user is illustrated in Figure 3.14.

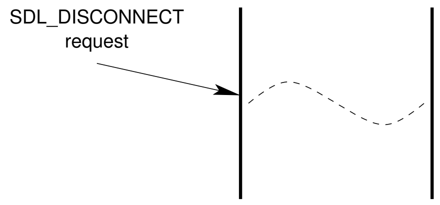

Figure 3.14: *Message Flow: Successful Disconnection Service by SDLS User*

A successful invocation of the disconnection service by the SDLS provider is illustrated in Figure 3.15.

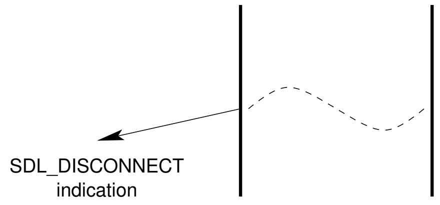

Figure 3.15: *Message Flow: Successful Disconnection Service by SDLS Provider*

4 Primitives

4.1 Local Management Service Primitives

These service primitives implement the local management services (see Section 3.1 [Local Management Services], page 13).

4.1.1 Acknowledgement Service Primitives

These service primitives implement the acknowledgement service (see Section 3.1.1 [Acknowledgement Service], page 13).

4.1.1.1 LMI_OK_ACK

Description

This primitive is used to acknowledge receipt and successful service completion for primitives requiring acknowledgement that have no confirmation primitive.

Format

This primitive consists of one `M_PCPROTO` message block, structured as follows:

```
typedef struct {
    lmi_long lmi_primitive;
    lmi_long lmi_correct_primitive;
    lmi_ulong lmi_state;
} lmi_ok_ack_t;
```

Parameters

The service primitive contains the following parameters:

lmi_primitive
: Indicates the service primitive type. Always `LMI_OK_ACK`.

lmi_correct_primitive
: Indicates the service primitive that was received and serviced correctly. This field can be one of the following values:

  `LMI_ATTACH_REQ`
  : Attach request.

  `LMI_DETACH_REQ`
  : Detach request.

lmi_state
: Indicates the current state of the LMS provider at the time that the primitive was issued. This field can be one of the following values:

  `LMI_UNATTACHED`
  : No PPA attached, awaiting `LMI_ATTACH_REQ`.

  `LMI_UNUSABLE`
  : Device cannot be used, Stream in hung state.

`LMI_DISABLED`
PPA attached, awaiting `LMI_ENABLE_REQ`.

`LMI_ENABLED`
Ready for use, awaiting primitive exchange.

State

This primitive is issued by the LMS provider in the `LMI_ATTACH_PENDING` or `LMI_DETACH_PENDING` state.

New State

The new state is `LMI_UNATTACHED` or `LMI_DISABLED`, depending on the primitive to which the message is responding.

4.1.1.2 LMI_ERROR_ACK

Description

The error acknowledgement primitive is used to acknowledge receipt and unsuccessful service completion for primitives requiring acknowledgement.

Format

The error acknowledgement primitive consists of one `M_PCPROTO` message block, structured as follows:

```
typedef struct {
    lmi_long lmi_primitive;
    lmi_ulong lmi_errno;
    lmi_ulong lmi_reason;
    lmi_long lmi_error_primitive;
    lmi_ulong lmi_state;
} lmi_error_ack_t;
```

Parameters

The error acknowledgement primitive contains the following parameters:

lmi_primitive

Indicates the primitive type. Always `LMI_ERROR_ACK`.

lmi_errno

Indicates the LM error number. This field can have one of the following values:

`[LMI_UNSPEC]`
Unknown or unspecified.

`[LMI_BADADDRESS]`
Address was invalid.

`[LMI_BADADDRTYPE]`
Invalid address type.

`[LMI_BADDIAL]`
(Not used.)

`[LMI_BADDIALTYPE]`
(Not used.)

`[LMI_BADDISPOSAL]`
Invalid disposal parameter.

`[LMI_BADFRAME]`
Defective SDU received.

`[LMI_BADPPA]`
Invalid PPA identifier.

`[LMI_BADPRIM]`
Unrecognized primitive.

`[LMI_DISC]`
Disconnected.

`[LMI_EVENT]`
Protocol-specific event occurred.

`[LMI_FATALERR]`
Device has become unusable.

`[LMI_INITFAILED]`
Link initialization failed.

`[LMI_NOTSUPP]`
Primitive not supported by this device.

`[LMI_OUTSTATE]`
Primitive was issued from invalid state.

`[LMI_PROTOSHORT]`
`M_PROTO` block too short.

`[LMI_SYSERR]`
UNIX system error.

`[LMI_WRITEFAIL]`
Unitdata request failed.

`[LMI_CRCERR]`
CRC or FCS error.

`[LMI_DLE_EOT]`
DLE EOT detected.

`[LMI_FORMAT]`
Format error detected.

`[LMI_HDLC_ABORT]`
Aborted frame detected.

`[LMI_OVERRUN]`
Input overrun.

`[LMI_TOOSHORT]`
Frame too short.

`[LMI_INCOMPLETE]`
Partial frame received.

`[LMI_BUSY]`
Telephone was busy.

`[LMI_NOANSWER]`
Connection went unanswered.

`[LMI_CALLREJECT]`
Connection rejected.

`[LMI_HDLC_IDLE]`
HDLC line went idle.

`[LMI_HDLC_NOTIDLE]`
HDLC link no longer idle.

`[LMI_QUIESCENT]`
: Line being reassigned.

`[LMI_RESUMED]`
: Line has been reassigned.

`[LMI_DSRTIMEOUT]`
: Did not see DSR in time.

`[LMI_LAN_COLLISIONS]`
: LAN excessive collisions.

`[LMI_LAN_REFUSED]`
: LAN message refused.

`[LMI_LAN_NOSTATION]`
: LAN no such station.

`[LMI_LOSTCTS]`
: Lost Clear to Send signal.

`[LMI_DEVERR]`
: Start of device-specific error codes.

lmi_reason

Indicates the reason for failure. This field is protocol-specific. When the *lmi_errno* field is `[LMI_SYSERR]`, the *lmi_reason* field is the UNIX error number as described in `errno(3)`.

lmi_error_primitive

Indicates the primitive that was in error. This field can have one of the following values:

`LMI_INFO_REQ`
: Information request.

`LMI_ATTACH_REQ`
: Attach request.

`LMI_DETACH_REQ`
: Detach request.

`LMI_ENABLE_REQ`
: Enable request.

`LMI_DISABLE_REQ`
: Disable request.

`LMI_OPTMGMT_REQ`
: Options management request.

`LMI_INFO_ACK`
: Information acknowledgement.

`LMI_OK_ACK`
: Successful receipt acknowledgement.

`LMI_ERROR_ACK`
: Error acknowledgement.

`LMI_ENABLE_CON`
Enable confirmation.

`LMI_DISABLE_CON`
Disable confirmation.

`LMI_OPTMGMT_ACK`
Options Management acknowledgement.

`LMI_ERROR_IND`
Error indication.

`LMI_STATS_IND`
Statistics indication.

`LMI_EVENT_IND`
Event indication.

lmi_state

Indicates the state of the LMS provider at the time that the primitive was issued. This field can have one of the following values:

`LMI_UNATTACHED`
No PPA attached, awaiting `LMI_ATTACH_REQ`.

`LMI_ATTACH_PENDING`
Waiting for attach.

`LMI_UNUSABLE`
Device cannot be used, STREAM in hung state.

`LMI_DISABLED`
PPA attached, awaiting `LMI_ENABLE_REQ`.

`LMI_ENABLE_PENDING`
Waiting to send `LMI_ENABLE_CON`.

`LMI_ENABLED`
Ready for use, awaiting primitive exchange.

`LMI_DISABLE_PENDING`
Waiting to send `LMI_DISABLE_CON`.

`LMI_DETACH_PENDING`
Waiting for detach.

State

This primitive can be issued in any state for which a local acknowledgement is not pending. The LMS provider state at the time that the primitive was issued is indicated in the primitive.

New State

The new state remains unchanged.

4.1.2 Information Reporting Service Primitives

These service primitives implement the information reporting service (see Section 3.1.2 [Information Reporting Service], page 14).

4.1.2.1 LMI_INFO_REQ

Description

This LMS user originated primitive is issued by the LMS user to request that the LMS provider return information concerning the capabilities and state of the LMS provider.

Format

The primitive consists of one `M_PROTO` or `M_PCPROTO` message block, structured as follows:

```
typedef struct {
    lmi_ulong lmi_primitive;
} lmi_info_req_t;
```

Parameters

This primitive contains the following parameters:

lmi_primitive
Specifies the primitive type. Always `LMI_INFO_REQ`.

State

This primitive may be issued in any state but only when a local acknowledgement is not pending.

New State

The new state remains unchanged.

Response

This primitive requires the LMS provider to acknowledge receipt of the primitive as follows:

- **Successful**: The LMS provider is required to acknowledge receipt of the primitive and provide the requested information using the `LMI_INFO_ACK` primitive.
- **Unsuccessful (non-fatal errors)**: The LMS provider is required to negatively acknowledge the primitive using the `LMI_ERROR_ACK` primitive, and include the reason for failure in the primitive.

Reasons for Failure

Non-Fatal Errors: applicable non-fatal errors are as follows:

`[LMI_UNSPEC]`
Unknown or unspecified.

`[LMI_BADADDRESS]`
Address was invalid.

`[LMI_BADADDRTYPE]`
Invalid address type.

`[LMI_BADDIAL]`
(Not used.)

[LMI_BADDIALTYPE]
(Not used.)

[LMI_BADDISPOSAL]
Invalid disposal parameter.

[LMI_BADFRAME]
Defective SDU received.

[LMI_BADPPA]
Invalid PPA identifier.

[LMI_BADPRIM]
Unrecognized primitive.

[LMI_DISC]
Disconnected.

[LMI_EVENT]
Protocol-specific event occurred.

[LMI_FATALERR]
Device has become unusable.

[LMI_INITFAILED]
Link initialization failed.

[LMI_NOTSUPP]
Primitive not supported by this device.

[LMI_OUTSTATE]
Primitive was issued from invalid state.

[LMI_PROTOSHORT]
`M_PROTO` block too short.

[LMI_SYSERR]
UNIX system error.

[LMI_WRITEFAIL]
Unitdata request failed.

[LMI_CRCERR]
CRC or FCS error.

[LMI_DLE_EOT]
DLE EOT detected.

[LMI_FORMAT]
Format error detected.

[LMI_HDLC_ABORT]
Aborted frame detected.

[LMI_OVERRUN]
Input overrun.

[LMI_TOOSHORT]
Frame too short.

`[LMI_INCOMPLETE]`
Partial frame received.

`[LMI_BUSY]`
Telephone was busy.

`[LMI_NOANSWER]`
Connection went unanswered.

`[LMI_CALLREJECT]`
Connection rejected.

`[LMI_HDLC_IDLE]`
HDLC line went idle.

`[LMI_HDLC_NOTIDLE]`
HDLC link no longer idle.

`[LMI_QUIESCENT]`
Line being reassigned.

`[LMI_RESUMED]`
Line has been reassigned.

`[LMI_DSRTIMEOUT]`
Did not see DSR in time.

`[LMI_LAN_COLLISIONS]`
LAN excessive collisions.

`[LMI_LAN_REFUSED]`
LAN message refused.

`[LMI_LAN_NOSTATION]`
LAN no such station.

`[LMI_LOSTCTS]`
Lost Clear to Send signal.

`[LMI_DEVERR]`
Start of device-specific error codes.

4.1.2.2 LMI_INFO_ACK

Description

This LMS provider originated primitive acknowledges receipt and successful processing of the `LMI_INFO_REQ` primitive and provides the requested information concerning the LMS provider.

Format

This message is formatted a one `M_PROTO` or `M_PCPROTO` message block, structured as follows:

```
typedef struct {
    lmi_long lmi_primitive;
    lmi_ulong lmi_version;
    lmi_ulong lmi_state;
    lmi_ulong lmi_max_sdu;
    lmi_ulong lmi_min_sdu;
    lmi_ulong lmi_header_len;
    lmi_ulong lmi_ppa_style;
    lmi_uchar lmi_ppa_addr[0];
} lmi_info_ack_t;
```

Parameters

The information acknowledgement service primitive has the following parameters:

lmi_primitive
: Indicates the service primitive type. Always `LMI_INFO_ACK`.

lmi_version
: Indicates the version of this specification that is being used by the LMS provider.

lmi_state
: Indicates the state of the LMS provider at the time that the information acknowledgement service primitive was issued. This field can be one of the following values:

`LMI_UNATTACHED`
: No PPA attached, awaiting `LMI_ATTACH_REQ`.

`LMI_ATTACH_PENDING`
: Waiting for attach.

`LMI_UNUSABLE`
: Device cannot be used, STREAM in hung state.

`LMI_DISABLED`
: PPA attached, awaiting `LMI_ENABLE_REQ`.

`LMI_ENABLE_PENDING`
: Waiting to send `LMI_ENABLE_CON`.

`LMI_ENABLED`
: Ready for use, awaiting primitive exchange.

`LMI_DISABLE_PENDING`
: Waiting to send `LMI_DISABLE_CON`.

`LMI_DETACH_PENDING`
: Waiting for detach.

lmi_max_sdu

Indicates the maximum size of a Service Data Unit.

lmi_min_sdu

Indicates the minimum size of a Service Data Unit.

lmi_header_len

Indicates the amount of header space that should be reserved for placing LMS provider headers.

lmi_ppa_style

Indicates the PPA style of the LMS provider. This value can be one of the following values:

`LMI_STYLE1`

PPA is implicitly attached by `open(2s)`.

`LMI_STYLE2`

PPA must be explicitly attached using `LMI_ATTACH_REQ`.

lmi_ppa_addr

This is a variable length field. The length of the field is determined by the length of the `M_PROTO` or `M_PCPROTO` message block.

For a *Style 2* driver, when *lmi_ppa_style* is `LMI_STYLE2`, and when in an attached state, this field provides the current PPA associated with the Stream; the length is typically 4 bytes.

For a *Style 1* driver, when *lmi_ppa_style* is `LMI_STYLE1`, the length it 0 bytes.

State

This primitive can be issued in any state where a local acknowledgement is not pending.

New State

The new state remains unchanged.

4.1.3 Physical Point of Attachment Service Primitives

These service primitives implement the physical point of attachment service (see Section 3.1.3 [Physical Point of Attachment Service], page 14).

4.1.3.1 LMI_ATTACH_REQ

Description

This LMS user originated primitive requests that the Stream upon which the primitive is issued be associated with the specified Physical Point of Attachment (PPA). This primitive is only applicable to *Style 2* LMS provider Streams, that is, Streams that return `LMI_STYLE2` in the *lmi_ppa_style* field of the `LMI_INFO_ACK`.

Format

This primitive consists of one `M_PROTO` message block, structured as follows:

```
typedef struct {
    lmi_long lmi_primitive;
    lmi_uchar lmi_ppa[0];
} lmi_attach_req_t;
```

Parameters

The attach request primitive contains the following parameters:

lmi_primitive
Specifies the service primitive type. Always `LMI_ATTACH_REQ`.

lmi_ppa Specifies the Physical Point of Attachment (PPA) to which to associate the *Style 2* Stream. This is a variable length identifier whose length is determined by the length of the `M_PROTO` message block.

State

This primitive is only valid in state `LMI_UNATTACHED` and when a local acknowledgement is not pending.

New State

Upon success, the new state is `LMI_ATTACH_PENDING`. Upon failure, the state remains unchanged.

Response

The attach request service primitive requires that the LMS provider respond as follows:

- **Successful**: The LMS provider acknowledges receipt of the primitive and successful outcome of the attach service with a `LMI_OK_ACK` primitive. The new state is `LMI_DISABLED`.
- **Unsuccessful (non-fatal errors)**: The LMS provider acknowledges receipt of the primitive and failure of the attach service with a `LMI_ERROR_ACK` primitive containing the reason for failure. The new state remains unchanged.

Reasons for Failure

Non-Fatal Errors: applicable non-fatal errors are as follows:

`[LMI_UNSPEC]`
Unknown or unspecified.

`[LMI_BADADDRESS]`
Address was invalid.

`[LMI_BADADDRTYPE]`
Invalid address type.

`[LMI_BADDIAL]`
(Not used.)

`[LMI_BADDIALTYPE]`
(Not used.)

`[LMI_BADDISPOSAL]`
Invalid disposal parameter.

`[LMI_BADFRAME]`
Defective SDU received.

`[LMI_BADPPA]`
Invalid PPA identifier.

`[LMI_BADPRIM]`
Unrecognized primitive.

`[LMI_DISC]`
Disconnected.

`[LMI_EVENT]`
Protocol-specific event occurred.

`[LMI_FATALERR]`
Device has become unusable.

`[LMI_INITFAILED]`
Link initialization failed.

`[LMI_NOTSUPP]`
Primitive not supported by this device.

`[LMI_OUTSTATE]`
Primitive was issued from invalid state.

`[LMI_PROTOSHORT]`
`M_PROTO` block too short.

`[LMI_SYSERR]`
UNIX system error.

`[LMI_WRITEFAIL]`
Unitdata request failed.

`[LMI_CRCERR]`
CRC or FCS error.

`[LMI_DLE_EOT]`
DLE EOT detected.

[LMI_FORMAT]
Format error detected.

[LMI_HDLC_ABORT]
Aborted frame detected.

[LMI_OVERRUN]
Input overrun.

[LMI_TOOSHORT]
Frame too short.

[LMI_INCOMPLETE]
Partial frame received.

[LMI_BUSY]
Telephone was busy.

[LMI_NOANSWER]
Connection went unanswered.

[LMI_CALLREJECT]
Connection rejected.

[LMI_HDLC_IDLE]
HDLC line went idle.

[LMI_HDLC_NOTIDLE]
HDLC link no longer idle.

[LMI_QUIESCENT]
Line being reassigned.

[LMI_RESUMED]
Line has been reassigned.

[LMI_DSRTIMEOUT]
Did not see DSR in time.

[LMI_LAN_COLLISIONS]
LAN excessive collisions.

[LMI_LAN_REFUSED]
LAN message refused.

[LMI_LAN_NOSTATION]
LAN no such station.

[LMI_LOSTCTS]
Lost Clear to Send signal.

[LMI_DEVERR]
Start of device-specific error codes.

4.1.3.2 LMI_DETACH_REQ

Description

This LMS user originated primitive requests that the Stream upon which the primitive is issued be disassociated from the Physical Point of Appearance (PPA) to which it is currently attached. This primitive is only applicable to *Style 2* LMS provider Streams, that is, Streams that return LMI_STYLE2 in the *lmi_ppa_style* field of the LMI_INFO_ACK.

Format

The detach request service primitive consists of one M_PROTO message block, structured as follows:

```
typedef struct {
    lmi_long lmi_primitive;
} lmi_detach_req_t;
```

Parameters

The detach request service primitive contains the following parameters:

lmi_primitive
: Specifies the service primitive type. Always LMI_DETACH_REQ.

State

This primitive is valid in the LMI_DISABLED state and when no local acknowledgement is pending.

New State

Upon success, the new state is LMI_DETACH_PENDING. Upon failure, the state remains unchanged.

Response

The detach request service primitive requires that the LMS provider respond as follows:

- **Successful**: The LMS provider acknowledges receipt of the primitive and successful outcome of the detach service with a LMI_OK_ACK primitive. The new state is LMI_UNATTACHED.
- **Unsuccessful (non-fatal errors)**: The LMS provider acknowledges receipt of the primitive and failure of the detach service with a LMI_ERROR_ACK primitive containing the reason for failure. The new state remains unchanged.

Reasons for Failure

Non-Fatal Errors: applicable non-fatal errors are as follows:

[LMI_UNSPEC]
: Unknown or unspecified.

[LMI_BADADDRESS]
: Address was invalid.

[LMI_BADADDRTYPE]
: Invalid address type.

[LMI_BADDIAL]
: (Not used.)

`[LMI_BADDIALTYPE]`
(Not used.)

`[LMI_BADDISPOSAL]`
Invalid disposal parameter.

`[LMI_BADFRAME]`
Defective SDU received.

`[LMI_BADPPA]`
Invalid PPA identifier.

`[LMI_BADPRIM]`
Unrecognized primitive.

`[LMI_DISC]`
Disconnected.

`[LMI_EVENT]`
Protocol-specific event occurred.

`[LMI_FATALERR]`
Device has become unusable.

`[LMI_INITFAILED]`
Link initialization failed.

`[LMI_NOTSUPP]`
Primitive not supported by this device.

`[LMI_OUTSTATE]`
Primitive was issued from invalid state.

`[LMI_PROTOSHORT]`
`M_PROTO` block too short.

`[LMI_SYSERR]`
UNIX system error.

`[LMI_WRITEFAIL]`
Unitdata request failed.

`[LMI_CRCERR]`
CRC or FCS error.

`[LMI_DLE_EOT]`
DLE EOT detected.

`[LMI_FORMAT]`
Format error detected.

`[LMI_HDLC_ABORT]`
Aborted frame detected.

`[LMI_OVERRUN]`
Input overrun.

`[LMI_TOOSHORT]`
Frame too short.

[LMI_INCOMPLETE]
Partial frame received.

[LMI_BUSY]
Telephone was busy.

[LMI_NOANSWER]
Connection went unanswered.

[LMI_CALLREJECT]
Connection rejected.

[LMI_HDLC_IDLE]
HDLC line went idle.

[LMI_HDLC_NOTIDLE]
HDLC link no longer idle.

[LMI_QUIESCENT]
Line being reassigned.

[LMI_RESUMED]
Line has been reassigned.

[LMI_DSRTIMEOUT]
Did not see DSR in time.

[LMI_LAN_COLLISIONS]
LAN excessive collisions.

[LMI_LAN_REFUSED]
LAN message refused.

[LMI_LAN_NOSTATION]
LAN no such station.

[LMI_LOSTCTS]
Lost Clear to Send signal.

[LMI_DEVERR]
Start of device-specific error codes.

4.1.4 Initialization Service Primitives

Initialization service primitives allow the LMS user to enable or disable the protocol service interface. Enabling the protocol service interface may require that some action be taken to prepare the protocol service interface for use or to remove it from use. For example, where the PPA corresponds to a signalling data link identifier as defined in Q.704, it may be necessary to perform switching to connect or disconnect the circuit identification code associated with the signalling data link identifier.

These service primitives implement the initialization service (see Section 3.1.4 [Initialization Service], page 16).

4.1.4.1 LMI_ENABLE_REQ

Description

This LMS user originated primitive requests that the LMS provider perform the actions necessary to enable the protocol service interface and confirm that it is enabled. This primitive is applicable to both styles of PPA.

Format

The enable request service primitive consists of one `M_PROTO` message block, structured as follows:

```
typedef struct {
    lmi_long lmi_primitive;
    lmi_uchar lmi_rem[0];
} lmi_enable_req_t;
```

Parameters

The enable request service primitive contains the following parameters:

lmi_primitive
Specifies the service primitive type. Always `LMI_ENABLE_REQ`.

lmi_rem Specifies a remote address to which to connect the PPA. The need for and form of this address is provider-specific. The length of the field is determined by the length of the `M_PROTO` message block. This remote address could be a circuit identification code, an IP address, or some other form of circuit or channel identifier.

State

This primitive is valid in the `LMI_DISABLED` state and when no local acknowledgement is pending.

New State

Upon success the new state is `LMI_ENABLE_PENDING`. Upon failure, the state remains unchanged.

Response

The enable request service primitive requires that the LMS provider acknowledge receipt of the primitive as follows:

- **Successful**: When successful, the LMS provider acknowledges successful completion of the enable service with an `LMI_ENABLE_CON` primitive. The new state is `LMI_ENABLED`.
- **Unsuccessful (non-fatal errors)**: When unsuccessful, the LMS provider acknowledges the failure of the enable service wtih an `LMI_ERROR_ACK` primitive containing the error. The new state remains unchanged.

Reasons for Failure

Non-Fatal Errors: applicable non-fatal errors are as follows:

`[LMI_UNSPEC]`
Unknown or unspecified.

`[LMI_BADADDRESS]`
Address was invalid.

`[LMI_BADADDRTYPE]`
Invalid address type.

`[LMI_BADDIAL]`
(Not used.)

`[LMI_BADDIALTYPE]`
(Not used.)

`[LMI_BADDISPOSAL]`
Invalid disposal parameter.

`[LMI_BADFRAME]`
Defective SDU received.

`[LMI_BADPPA]`
Invalid PPA identifier.

`[LMI_BADPRIM]`
Unrecognized primitive.

`[LMI_DISC]`
Disconnected.

`[LMI_EVENT]`
Protocol-specific event occurred.

`[LMI_FATALERR]`
Device has become unusable.

`[LMI_INITFAILED]`
Link initialization failed.

`[LMI_NOTSUPP]`
Primitive not supported by this device.

`[LMI_OUTSTATE]`
Primitive was issued from invalid state.

`[LMI_PROTOSHORT]`
`M_PROTO` block too short.

`[LMI_SYSERR]`
UNIX system error.

`[LMI_WRITEFAIL]`
Unitdata request failed.

`[LMI_CRCERR]`
CRC or FCS error.

`[LMI_DLE_EOT]`
DLE EOT detected.

`[LMI_FORMAT]`
Format error detected.

`[LMI_HDLC_ABORT]`
Aborted frame detected.

`[LMI_OVERRUN]`
Input overrun.

`[LMI_TOOSHORT]`
Frame too short.

`[LMI_INCOMPLETE]`
Partial frame received.

`[LMI_BUSY]`
Telephone was busy.

`[LMI_NOANSWER]`
Connection went unanswered.

`[LMI_CALLREJECT]`
Connection rejected.

`[LMI_HDLC_IDLE]`
HDLC line went idle.

`[LMI_HDLC_NOTIDLE]`
HDLC link no longer idle.

`[LMI_QUIESCENT]`
Line being reassigned.

`[LMI_RESUMED]`
Line has been reassigned.

`[LMI_DSRTIMEOUT]`
Did not see DSR in time.

`[LMI_LAN_COLLISIONS]`
LAN excessive collisions.

`[LMI_LAN_REFUSED]`
LAN message refused.

`[LMI_LAN_NOSTATION]`
LAN no such station.

`[LMI_LOSTCTS]`
Lost Clear to Send signal.

`[LMI_DEVERR]`
Start of device-specific error codes.

4.1.4.2 LMI_ENABLE_CON

Description

This LMS provider originated primitive is issued by the LMS provider to confirm the successful completion of the enable service.

Format

The enable confirmation service primitive consists of one `M_PROTO` message block, structured as follows:

```
typedef struct {
    lmi_long lmi_primitive;
    lmi_ulong lmi_state;
} lmi_enable_con_t;
```

Parameters

The enable confirmation service primitive contains the following parameters:

lmi_primitive
Indicates the service primitive type. Always `LMI_ENABLE_CON`.

lmi_state Indicates the state following issuing the enable confirmation primitive. This field can take on one of the following values:

`LMI_ENABLED`
Ready for use, awaiting primitive exchange.

State

This primitive is issued by the LMS provider in the `LMI_ENABLE_PENDING` state.

New State

The new state is `LMI_ENABLED`.

4.1.4.3 LMI_DISABLE_REQ

Description

This LMS user originated primitive requests that the LMS provider perform the actions necessary to disable the protocol service interface and confirm that it is disabled. The primitive is applicable to both styles of PPA.

Format

The disable request service primitive consists of one `M_PROTO` message block, structured as follows:

```
typedef struct {
    lmi_long lmi_primitive;
} lmi_disable_req_t;
```

Parameters

The disable request service primitive contains the following parameters:

lmi_primitive
Specifies the service primitive type. Always `LMI_DISABLE_REQ`.

State

The disable request service primitive is valid in the `LMI_ENABLED` state and when no local acknowledgement is pending.

New State

Upon success, the new state is `LMI_DISABLE_PENDING`. Upon failure, the state remains unchanged.

Response

The disable request service primitive requires the LMS provider to acknowledge receipt of the primitive as follows:

- **Successful**: When successful, the LMS provider acknowledges successful completion of the disable service with an `LMI_DISABLE_CON` primitive. The new state is `LMI_DISABLED`.
- **Unsuccessful (non-fatal errors)**: When unsuccessful, the LMS provider acknowledges the failure of the disable service with an `LMI_ERROR_ACK` primitive containing the error. The new state remains unchanged.

Reasons for Failure

Non-Fatal Errors: applicable non-fatal errors are as follows:

`[LMI_UNSPEC]`
Unknown or unspecified.

`[LMI_BADADDRESS]`
Address was invalid.

`[LMI_BADADDRTYPE]`
Invalid address type.

`[LMI_BADDIAL]`
(Not used.)

[LMI_BADDIALTYPE]
(Not used.)

[LMI_BADDISPOSAL]
Invalid disposal parameter.

[LMI_BADFRAME]
Defective SDU received.

[LMI_BADPPA]
Invalid PPA identifier.

[LMI_BADPRIM]
Unrecognized primitive.

[LMI_DISC]
Disconnected.

[LMI_EVENT]
Protocol-specific event occurred.

[LMI_FATALERR]
Device has become unusable.

[LMI_INITFAILED]
Link initialization failed.

[LMI_NOTSUPP]
Primitive not supported by this device.

[LMI_OUTSTATE]
Primitive was issued from invalid state.

[LMI_PROTOSHORT]
M_PROTO block too short.

[LMI_SYSERR]
UNIX system error.

[LMI_WRITEFAIL]
Unitdata request failed.

[LMI_CRCERR]
CRC or FCS error.

[LMI_DLE_EOT]
DLE EOT detected.

[LMI_FORMAT]
Format error detected.

[LMI_HDLC_ABORT]
Aborted frame detected.

[LMI_OVERRUN]
Input overrun.

[LMI_TOOSHORT]
Frame too short.

[LMI_INCOMPLETE]
Partial frame received.

[LMI_BUSY]
Telephone was busy.

[LMI_NOANSWER]
Connection went unanswered.

[LMI_CALLREJECT]
Connection rejected.

[LMI_HDLC_IDLE]
HDLC line went idle.

[LMI_HDLC_NOTIDLE]
HDLC link no longer idle.

[LMI_QUIESCENT]
Line being reassigned.

[LMI_RESUMED]
Line has been reassigned.

[LMI_DSRTIMEOUT]
Did not see DSR in time.

[LMI_LAN_COLLISIONS]
LAN excessive collisions.

[LMI_LAN_REFUSED]
LAN message refused.

[LMI_LAN_NOSTATION]
LAN no such station.

[LMI_LOSTCTS]
Lost Clear to Send signal.

[LMI_DEVERR]
Start of device-specific error codes.

4.1.4.4 LMI_DISABLE_CON

Description

This LMS provider originated primitive is issued by the LMS provider to confirm the successful completion of the disable service.

Format

The disable confirmation service primitive consists of one `M_PROTO` message block, structured as follows:

```
typedef struct {
    lmi_long lmi_primitive;
    lmi_ulong lmi_state;
} lmi_disable_con_t;
```

Parameters

The disable confirmation service primitive contains the following parameters:

lmi_primitive
: Indicates the service primitive type. Always `LMI_DISABLE_CON`.

lmi_state
: Indicates the state following issuing the disable confirmation primitive. This field can take on one of the following values:

  `LMI_DISABLED`
  : PPA attached, awaiting `LMI_ENABLE_REQ`.

State

This primitive is issued by the LMS provider in the `LMI_DISABLE_PENDING` state.

New State

The new state is `LMI_DISABLED`.

4.1.5 Options Management Service Primitives

The options management service primitives allow the LMS user to negotiate options with the LMS provider, retrieve the current and default values of options, and check that values specified for options are correct.

The options management service primitive implement the options management service (see Section 3.1.5 [Options Management Service], page 17).

4.1.5.1 LMI_OPTMGMT_REQ

Description

This LMS user originated primitive requests that LMS provider options be managed.

Format

The option management request service primitive consists of one M_PROTO or M_PCPROTO message block, structured as follows:

```
typedef struct {
    lmi_long lmi_primitive;
    lmi_ulong lmi_opt_length;
    lmi_ulong lmi_opt_offset;
    lmi_ulong lmi_mgmt_flags;
} lmi_optmgmt_req_t;
```

Parameters

The option management request service primitive contains the following parameters:

lmi_primitive

Specifies the service primitive type. Always LMI_OPTMGMT_REQ.

lmi_opt_length

Specifies the length of the options.

lmi_opt_offset

Specifies the offset, from the beginning of the M_PROTO message block, of the start of the options.

lmi_mgmt_flags

Specifies the management flags that determine what operation the LMS provider is expected to perform on the specified options. This field can assume one of the following values:

LMI_NEGOTIATE

Negotiate the specified value of each specified option and return the negotiated value.

LMI_CHECK

Check the validity of the specified value of each specified option and return the result. Do not alter the current value assumed by the LMS provider.

LMI_DEFAULT

Return the default value for the specified options (or all options). Do not alter the current value assumed by the LMS provider.

`LMI_CURRENT`
Return the current value for the specified options (or all options). Do not alter the current value assumed by the LMS provider.

State

This primitive is valid in any state where a local acknowledgement is not pending.

New State

The new state remains unchanged.

Response

The option management request service primitive requires the LMS provider to acknowledge receipt of the primitive as follows:

- **Successful**: Upon success, the LMS provider acknowledges receipt of the service primitive and successful completion of the options management service with an `LMI_OPTMGMT_ACK` primitive containing the options management result. The state remains unchanged.
- **Unsuccessful (non-fatal errors)**: Upon failure, the LMS provider acknowledges receipt of the service primitive and failure to complete the options management service with an `LMI_ERROR_ACK` primitive containing the error. The state remains unchanged.

Reasons for Failure

Non-Fatal Errors: applicable non-fatal errors are as follows:

`[LMI_UNSPEC]`
Unknown or unspecified.

`[LMI_BADADDRESS]`
Address was invalid.

`[LMI_BADADDRTYPE]`
Invalid address type.

`[LMI_BADDIAL]`
(Not used.)

`[LMI_BADDIALTYPE]`
(Not used.)

`[LMI_BADDISPOSAL]`
Invalid disposal parameter.

`[LMI_BADFRAME]`
Defective SDU received.

`[LMI_BADPPA]`
Invalid PPA identifier.

`[LMI_BADPRIM]`
Unrecognized primitive.

`[LMI_DISC]`
Disconnected.

[LMI_EVENT]
Protocol-specific event occurred.

[LMI_FATALERR]
Device has become unusable.

[LMI_INITFAILED]
Link initialization failed.

[LMI_NOTSUPP]
Primitive not supported by this device.

[LMI_OUTSTATE]
Primitive was issued from invalid state.

[LMI_PROTOSHORT]
M_PROTO block too short.

[LMI_SYSERR]
UNIX system error.

[LMI_WRITEFAIL]
Unitdata request failed.

[LMI_CRCERR]
CRC or FCS error.

[LMI_DLE_EOT]
DLE EOT detected.

[LMI_FORMAT]
Format error detected.

[LMI_HDLC_ABORT]
Aborted frame detected.

[LMI_OVERRUN]
Input overrun.

[LMI_TOOSHORT]
Frame too short.

[LMI_INCOMPLETE]
Partial frame received.

[LMI_BUSY]
Telephone was busy.

[LMI_NOANSWER]
Connection went unanswered.

[LMI_CALLREJECT]
Connection rejected.

[LMI_HDLC_IDLE]
HDLC line went idle.

[LMI_HDLC_NOTIDLE]
HDLC link no longer idle.

`[LMI_QUIESCENT]`
Line being reassigned.

`[LMI_RESUMED]`
Line has been reassigned.

`[LMI_DSRTIMEOUT]`
Did not see DSR in time.

`[LMI_LAN_COLLISIONS]`
LAN excessive collisions.

`[LMI_LAN_REFUSED]`
LAN message refused.

`[LMI_LAN_NOSTATION]`
LAN no such station.

`[LMI_LOSTCTS]`
Lost Clear to Send signal.

`[LMI_DEVERR]`
Start of device-specific error codes.

4.1.5.2 LMI_OPTMGMT_ACK

Description

This LMS provider originated primitive is issued by the LMS provider upon successful completion of the options management service. It indicates the outcome of the options management operation requested by the LMS user in a `LMI_OPTMGMT_REQ` primitive.

Format

The option management acknowledgement service primitive consists of one `M_PCPROTO` message block, structured as follows:

```
typedef struct {
    lmi_long lmi_primitive;
    lmi_ulong lmi_opt_length;
    lmi_ulong lmi_opt_offset;
    lmi_ulong lmi_mgmt_flags;
} lmi_optmgmt_ack_t;
```

Parameters

The option management acknowledgement service primitive contains the following parameters:

lmi_primitive
Indicates the service primitive type. Always `LMI_OPTMGMT_ACK`.

lmi_opt_length
Indicates the length of the returned options.

lmi_opt_offset
Indicates the offset of the returned options from the start of the `M_PCPROTO` message block.

lmi_mgmt_flags
Indicates the returned management flags. These flags indicate the overall success of the options management service. This field can assume one of the following values:

`LMI_SUCCESS`
The LMS provider succeeded in negotiating or returning all of the options specified by the LMS user in the `LMI_OPTMGMT_REQ` primitive.

`LMI_FAILURE`
The LMS provider failed to negotiate one or more of the options specified by the LMS user.

`LMI_PARTSUCCESS`
The LMS provider negotiated a value of lower quality for one or more of the options specified by the LMS user.

`LMI_READONLY`
The LMS provider failed to negotiate one or more of the options specified by the LMS user because the option is treated as read-only by the LMS provider.

`LMI_NOTSUPPORT`
The LMS provider failed to recognize one or more of the options specified by the LMS user.

State

This primitive is issued by the LMS provider in direct response to an `LMI_OPTMGMT_REQ` primitive.

New State

The new state remains unchanged.

Rules

The LMS provider observes the following rules when processing option management service requests:

— When the *lmi_mgmt_flags* field in the `LMI_OPTMGMT_REQ` primitive is set to `LMI_NEGOTIATE`, the LMS provider will attempt to negotiate a value for each of the options specified in the request.

— When the flags are `LMI_DEFAULT`, the LMS provider will return the default values of the specified options, or the default values of all options known to the LMS provider if no options were specified.

— When the flags are `LMI_CURRENT`, the LMS provider will return the current values of the specified options, or all options.

— When the flags are `LMI_CHECK`, the LMS provider will attempt to negotiate a value for each of the options specified in the request and return the resulg of the negotiation, but will not affect the current value of the option.

4.1.6 Event Reporting Service Primitives

The event reporting service primitives allow the LMS provider to indicate asynchronous errors, events and statistics collection to the LMS user.

These service primitives implement the event reporting service (see Section 3.1.8 [Event Reporting Service], page 19).

4.1.6.1 LMI_ERROR_IND

Description

This LMS provider originated service primitive is issued by the LMS provider when it detects and asynchronous error event. The service primitive is applicable to all styles of PPA.

Format

The error indication service primitive consists of one `M_PROTO` message block, structured as follows:

```
typedef struct {
    lmi_long lmi_primitive;
    lmi_ulong lmi_errno;
    lmi_ulong lmi_reason;
    lmi_ulong lmi_state;
} lmi_error_ind_t;
```

Parameters

The error indication service primitive contains the following parameters:

lmi_primitive
: Indicates the service primitive type. Always `LMI_ERROR_IND`.

lmi_errno
: Indicates the LMI error number describing the error. This field can have one of the following values:

`[LMI_UNSPEC]`
: Unknown or unspecified.

`[LMI_BADADDRESS]`
: Address was invalid.

`[LMI_BADADDRTYPE]`
: Invalid address type.

`[LMI_BADDIAL]`
: (Not used.)

`[LMI_BADDIALTYPE]`
: (Not used.)

`[LMI_BADDISPOSAL]`
: Invalid disposal parameter.

`[LMI_BADFRAME]`
: Defective SDU received.

`[LMI_BADPPA]`
: Invalid PPA identifier.

`[LMI_BADPRIM]`
Unrecognized primitive.

`[LMI_DISC]`
Disconnected.

`[LMI_EVENT]`
Protocol-specific event occurred.

`[LMI_FATALERR]`
Device has become unusable.

`[LMI_INITFAILED]`
Link initialization failed.

`[LMI_NOTSUPP]`
Primitive not supported by this device.

`[LMI_OUTSTATE]`
Primitive was issued from invalid state.

`[LMI_PROTOSHORT]`
`M_PROTO` block too short.

`[LMI_SYSERR]`
UNIX system error.

`[LMI_WRITEFAIL]`
Unitdata request failed.

`[LMI_CRCERR]`
CRC or FCS error.

`[LMI_DLE_EOT]`
DLE EOT detected.

`[LMI_FORMAT]`
Format error detected.

`[LMI_HDLC_ABORT]`
Aborted frame detected.

`[LMI_OVERRUN]`
Input overrun.

`[LMI_TOOSHORT]`
Frame too short.

`[LMI_INCOMPLETE]`
Partial frame received.

`[LMI_BUSY]`
Telephone was busy.

`[LMI_NOANSWER]`
Connection went unanswered.

`[LMI_CALLREJECT]`
Connection rejected.

`[LMI_HDLC_IDLE]`
: HDLC line went idle.

`[LMI_HDLC_NOTIDLE]`
: HDLC link no longer idle.

`[LMI_QUIESCENT]`
: Line being reassigned.

`[LMI_RESUMED]`
: Line has been reassigned.

`[LMI_DSRTIMEOUT]`
: Did not see DSR in time.

`[LMI_LAN_COLLISIONS]`
: LAN excessive collisions.

`[LMI_LAN_REFUSED]`
: LAN message refused.

`[LMI_LAN_NOSTATION]`
: LAN no such station.

`[LMI_LOSTCTS]`
: Lost Clear to Send signal.

`[LMI_DEVERR]`
: Start of device-specific error codes.

lmi_reason

Indicates the reason for failure. This field is protocol-specific. When the *lmi_errno* field is `[LMI_SYSERR]`, the *lmi_reason* field is the UNIX error number as described in `errno(3)`.

lmi_state

Indicates the state of the LMS provider at the time that the primitive was issued. This field can have one of the following values:

`LMI_UNATTACHED`
: No PPA attached, awaiting `LMI_ATTACH_REQ`.

`LMI_ATTACH_PENDING`
: Waiting for attach.

`LMI_UNUSABLE`
: Device cannot be used, STREAM in hung state.

`LMI_DISABLED`
: PPA attached, awaiting `LMI_ENABLE_REQ`.

`LMI_ENABLE_PENDING`
: Waiting to send `LMI_ENABLE_CON`.

`LMI_ENABLED`
: Ready for use, awaiting primitive exchange.

`LMI_DISABLE_PENDING`
: Waiting to send `LMI_DISABLE_CON`.

`LMI_DETACH_PENDING`
: Waiting for detach.

State

This primitive can be issued in any state for which a local acknowledgement is not pending. The LMS provider state at the time that the primitive was issued is indicated in the primitive.

New State

The new state remains unchanged.

4.1.6.2 LMI_STATS_IND

Description

This LMS provider originated primitive is issued by the LMS provider to indicate a periodic statistics collection event. The service primitive is applicable to all styles of PPA.

Format

The statistics indication service primitive consists of one `M_PROTO` message block, structured as follows:

```
typedef struct {
    lmi_long lmi_primitive;
    lmi_ulong lmi_interval;
    lmi_ulong lmi_timestamp;
} lmi_stats_ind_t;
```

Following this structure within the `M_PROTO` message block is the provider-specific statistics.

Parameters

The statistics indication service primitive contains the following parameters:

lmi_primitive
: Indicates the service primitive type. Always `LMI_STATS_IND`.

lmi_interval
: Indicates the statistics collection interval to which the statistics apply. This interval is specified in milliseconds.

lmi_timestamp
: Indicates the UNIX time (from epoch) at which statistics were collected. The timestamp is given in milliseconds from epoch.

State

This service primitive may be issued by the LMS provider in any state in which a local acknowledgement is not pending.

New State

The new state remains unchanged.

4.1.6.3 LMI_EVENT_IND

Description

This LMS provider originated primitive is issued by the LMS provider to indicate an asynchronous event. The service primitive is applicable to all styles of PPA.

Format

The event indication service primitive consists of one `M_PROTO` message block, structured as follows:

```
typedef struct {
    lmi_long lmi_primitive;
    lmi_ulong lmi_objectid;
    lmi_ulong lmi_timestamp;
    lmi_ulong lmi_severity;
} lmi_event_ind_t;
```

Following this structure within the `M_PROTO` message block is the provider-specific event information.

Parameters

THe event indication service primitive contains the following parameters:

lmi_primitive
Indicates the service primitive type. Always `LMI_EVENT_IND`.

lmi_objectid
Indicates the provider-specific object identifier that identifies the managed object to which the event is associated.

lmi_timestamp
Indicates the UNIX time from epoch (in milliseconds).

lmi_severity
Indicates the provider-specific severity of the event.

State

This service primitive can be issued by the LMS provider in any state where a local acknowledgement is not pending. Normally the LMS provider must be in the `LMI_ENABLED` state for event reporting to occur.

New State

The new state remains unchanged.

4.2 Protocol Service Primitives

Protocol service primitives implement the Signalling Data Link Interface protocol. Protocol service primitives provide the SDLS user with the ability to connect transmission or reception directions of the bit stream, pass bits for transmission and accept received bits.

These service primitives implement the protocol services (see Section 3.2 [Protocol Services], page 19).

4.2.1 Connection Service Primitives

The connection service primitives permit the SDLS user to establish a connection between the line (circuit or channel) and the SDLS user in the transmit, receive, or both, directions.

These service primitives implement the connection service (see Section 3.2.1 [Connection Service], page 19).

4.2.1.1 SDL_CONNECT_REQ

Description

This SDLS user originated service primitive allows the SDLS user to connect the user Stream to the medium in the transmit, receive, or both, directions.

Format

The connect request primitive consists of one `M_PROTO` message block, structured as follows:

```
typedef struct {
    sdl_long sdl_primitive;
    sdl_ulong sdl_flags;
} sdl_connect_req_t;
```

Parameters

The connect request service primitive contains the following parameters:

sdl_primitive
: Specifies the service primitive type. Always `SDL_CONNECT_REQ`.

sdl_flags
: Specifies the direction in which to connect. This field can contain a bitwise OR of one or more of the following flags:

  `SDL_RX_DIRECTION`
  : Specifies that the SDLS user Stream is to be connected to the medium in the receive direction.

  `SDL_TX_DIRECTION`
  : Specifies that the SDLS user Stream is to be connected to the medium in the transmit direction.

State

This service primitive is only valid in the `LMI_ENABLED` state.

New State

The state remains unchanged.

Response

The connection request service primitive is not acknowledged. However, the primitive may result in a non-fatal error as follows:

- **Successful:** Upon success, the connection request service primitive is not acknowledged.
- **Unsuccessful (non-fatal errors):** Upon failure, the SDLS provider indicates a non-fatal error with a `LMI_ERROR_ACK` message containing the error.

Reasons for Failure

4.2.2 Data Transfer Service Primitives

The data transfer service primitives permit the SDLS user to pass bits for transmission to the SDLS provider and accept received bits from the SDLS provider.

These service primitives implement the data transfer service (see Section 3.2.2 [Data Transfer Service], page 20).

4.2.2.1 SDL_BITS_FOR_TRANSMISSION_REQ

Description

This SDLS user originated primitive allows the SDLS user to specify bits for transmission on the medium.

Format

The transmission request service primitive consists of one optional `M_PROTO` message block followed by one or more `M_DATA` message blocks containing the bits for transmission. The `M_PROTO` message block is structured as follows:

```
typedef struct {
    sdl_long sdl_primitive;
} sdl_bits_for_transmission_req_t;
```

Parameters

The transmission request service primitive contains the following parameters:

sdl_primitive

Specifies the service primitive type. Always `SDL_BITS_FOR_TRANSMISSION_REQ`.

State

This primitive is only valid in the `LMI_ENABLED` state.

New State

The state remains unchanged.

Response

Reasons for Failure

4.2.2.2 SDL_RECEIVED_BITS_IND

Description

This SDLS provider originated primitive is issued by the SDLS provider to indicate bits that were received on the medium.

Format

The receive indication service primitive consists of one optional `M_PROTO` message block followed by one or more `M_DATA` message blocks containing the received bits. The `M_PROTO` message block is structured as follows:

```
typedef struct {
    sdl_long sdl_primitive;
} sdl_received_bits_ind_t;
```

Parameters

The receive indication service primitive contains the following parameters:

sdl_primitive
Indicates the service primitive type. Always `SDL_RECEIVED_BITS_IND`.

State

This primitive is only issued by the SDLS provider in the `LMI_ENABLED` state.

New State

The state remains unchanged.

Response

Reasons for Failure

4.2.3 Disconnection Service Primitives

The disconnection service primitives permit the SDLS user to disconnect the Stream from the line (circuit or channel) for the transmit, receive, or both, directions. They also allow the SDLS provider to indicate that a disconnection has occured outside of SDLS user control.

These service primitives implement the disconnection service (see Section 3.2.3 [Disconnection Service], page 20).

4.2.3.1 SDL_DISCONNECT_REQ

Description

This SDLS user originated service primitive allows the SDLS user to disconnect the SDLS user Stream from the bit-stream in the transmit, receive, or both, directions.

Format

The disconnect request primitive consists of one `M_PROTO` message block, structured as follows:

```
typedef struct {
    sdl_long sdl_primitive;
    sdl_ulong sdl_flags;
} sdl_disconnect_req_t;
```

Parameters

The disconnect request service primitive contains the following parameters:

sdl_primitive
: Specifies the service primitive type. Always `SDL_DISCONNECT_REQ`.

sdl_flags
: Specifies the direction from which to disconnect. This field can be a bitwise OR of one or more of the following flags:

  `SDL_RX_DIRECTION`
  : Specifies that the SDLS user Stream is to be disconnected from the medium in the receive direction.

  `SDL_TX_DIRECTION`
  : Specifies that the SDLS user Stream is to be disconnected from the medium in the transmit direction.

State

This service primitive is only valid in the `LMI_ENABLED` state.

New State

The state remains unchanged.

Response

Reasons for Failure

4.2.3.2 SDL_DISCONNECT_IND

Description

This SDLS provider originated primitive is issued by the SDLS provider if an autonomous event results in the disconnection of the transmit and receive bit-streams from the SDLS user without an explicit SDLS user request.

Format

The disconnect indication primitive consists of one `M_PROTO` message block, structured as follows:

```
typedef struct {
    sdl_long sdl_primitive;
} sdl_disconnect_ind_t;
```

Parameters

State

New State

Response

Reasons for Failure

5 Diagnostics Requirements

Two error handling facilities should be provided to the SDLS user: one to handle non-fatal errors, and the other to handle fatal errors.

5.1 Non-Fatal Error Handling Facility

These are errors that do not change the state of the SDLS interface as seen by the SDLS user and provide the user with the option of reissuing the SDL primitive with the corrected options specification. The non-fatal error handling is provided only to those primitives that require acknowledgements, and uses the `LMI_ERROR_ACK` to report these errors. These errors retain the state of the SDLS interface the same as it was before the SDL provider received the primitive that was in error. Syntax errors and rule violations are reported via the non-fatal error handling facility.

5.2 Fatal Error Handling Facility

These errors are issued by the SDL provider when it detects errors that are not correctable by the SDL user, or if it is unable to report a correctible error to the SDLS user. Fatal errors are indicated via the STREAMS message type `M_ERROR` with the UNIX system error `[EPROTO]`. The `M_ERROR` STREAMS message type will result in the failure of all the UNIX system calls on the Stream. The SDLS user can recover from a fatal error by having all the processes close the files associated with the Stream, and then reopening them for processing.

6 Input-Output Controls

These input-output controls can be used to interrogate, negotiate, reset, collect and manage a given signalling data link or group of signalling data links. When issued on a SDL user Stream, they can only be used to affect the data link or links associated with the SDL user Stream. Detached *Style 2* Streams have no associated data links. When issued on a mangement Stream, they can be used to affect the configuration of any data link or links accessible to the management Stream (i.e. provided by the same driver or module, or temporarily linked from the control Stream).

Data links can have characteristics at the data link level, as well as characteristics at the group level. For example, the data link may not be looped back at the data link, but may be looped back at the link group (span). Where the data link represents a link within a multiplexed medium (such as PCM TDM facility), the MXI input-output controls might be available to interrogate, negotiate and otherwise manage the link group characteristics providing that the SDL user has sufficient privilege to do so.

Note that these input-output controls are not normally issued on the global management Stream by user processes. Rather, the Management Agent (SNMP Agent) for the driver or module is normally responsible for managing channels within the driver or module using these input-output controls. Normally these input-output controls would only be issued by user processes to affect the data link or links associated with the attached SDL user Stream.

6.1 Configuration

These input-output controls can be used to interrogate or negotiate the configuration of a given data link or group of data links.

```
typedef struct sdl_config {
} sdl_config_t;
```

The signalling data link configuration structure, `sdl_config_t`, contains the following members:

6.1.1 Get Configuration

`SDL_IOCGCONFIG`

Gets the signalling data link configuration. Upon success, the signalling data link configuration is written to the memory extent indicated by the pointer argument to the `ioctl(2s)` call.

6.1.2 Set Configuration

`SDL_IOCSCONFIG`

Sets the signalling data link configuration. Upon success, the signalling data link configuration is read from the memory extent pointed to by the pointer argument to the `ioctl(2s)` call.

6.1.3 Test Configuration

`SDL_IOCTCONFIG`

Test the signalling data link configuration. Upon success, the signalling data link configuration is read from the memory extent specified by the pointer argument to the `ioctl(2s)` call, values adjusted according to the rules for configuration, and the resulting configuraiton written back to the memory extent specified by the pointer argumnet to the `ioctl(2s)` call. Actual configuration is not changed.

6.1.4 Commit Configuration

`SDL_IOCCCONFIG`

Confirms the signalling data link configuration. Upon success, the signalling data link configuration is read from the memory extent specified by the pointer argument to the `ioctl(2s)` call, values adjusted according to the rules for configuration, the configuration applied, and then the resulting configuration written back to the memory extent specified by the pointer argument to the `ioctl(2s)` call.

Normally, the argument to the `SDL_IOCCCONFIG` call is the same as to an immediately preceding `SDL_IOCTCONFIG` call.

6.2 Options

These input-output controls can used to interrogate or negotiate the options associated with a given data link or group of data links.

6.3 State

These input-output controls can be used to interrotate or reset the state associated with a data link or a group of data links.

State input-output controls all take an argument containing a poitner to a `sdl_statem_t` structure, formatted as follows:

```
typedef struct sdl_statem {
} sdl_statem_t;
```

The signalling data link state structure, `sdl_statem_t`, contains the following members:

6.3.1 Get State

`SDL_IOCGSTATEM`

Requests that the state information be obtained and written to the `sdl_statem_t` structure pointed to by the argument to the input-output control.

6.3.2 Reset State

`SDL_IOCCMRESET`

Request that the state associated with the data link be reset. This input-output control takes no argument.

6.4 Statistics

These input-output controls can be used to collect statistics or set statistics collection intervals associated with a data link or group of data links.

Statistic input-output controls all take an argument containing a pointer to a `sdl_stats_t` structure, formatted as follows:

```
typedef struct sdl_stats {
} sdl_stats_t;
```

The signalling data link statistics structure, `sdl_stats_t`, contains the following members:

6.5 Events

These input-output controls can be used to specify the events that will be reported by a data link or data links.

Notification input-output controls all take an argument containing a pointer to a `sdl_notify_t` structure, formatted as follows:

```
typedef struct sdl_notify {
    sdl_ulong events;
} sdl_notify_t;
```

The signalling data link events structure, `sdl_notify_t`, contains the following members:

events Specifies or indicates a bitwise OR of the events associated wtih the data link. When a bit is set, it specifies that event reporting for the specific event is enabled for the data link; when clear, that the event reporting is disabled.

6.5.1 Get Notify

`SDL_IOCGNOTIFY`

Requests that the events associated with the data link be obtained and written to the `sdl_notify_t` structure pointed to by the argument to the input-output control.

6.5.2 Set Notify

`SDL_IOCSNOTIFY`

Requests that the events associated with the data link be read from the `sdl_notify_t` structure pointed to by the argument to the input-output control and set for the data link. Each bit set in the *events* member specifies an event for which notification is to be set.

6.5.3 Clear Notify

`SDL_IOCCNOTIFY`

Request that the events associated with the data link be read from the `sdl_notify_t` structure pointed to by the argument to the input-output control and cleared for the data link. Each bit set in the *events* member specifies an event for which notification is to be cleared.

6.6 Commands

These input-output controls can be used to manage a data link or data links.

Management input-output controls all take an argument containing a pointer to a `sdl_mgmt_t` structure, formatted as follows:

```
typedef struct sdl_mgmt {
    sdl_ulong cmd;
} sdl_mgmt_t;
```

The signalling data link management structure, `sdl_mgmt_t`, contains the following members:

6.6.1 Command

`SDL_IOCCMGMT`

Request that the management command be read from the `sdl_mgmt_t` structure pointed to by the argument to the input-output control and acted upon for the data link.

7 Management Information Base

The OPENSS7-MTP-MIB provides the following Signalling Data Link specific tables:

- Section 7.1 [MTP Signalling Data Link (SDL) Configuration Table], page 73.

7.1 MTP Signalling Data Link (SDL) Configuration Table

The *MTP Signalling Data Link (SDL) Configuration Table*, mtpSdlTable, is a table that provides specific configuration information for various *MTP Signalling Data Link Entities*.

Provides a table of Signalling Data Link (SDL) Entities. Each Signalling Data Link entity represents the path termination of the signalling data link within the signalling point as defined in ITU-T Rec. Q.702. The operational state is 'enabled' in normal operation. In case of failure of the part of the siganlling data link that is within the control of the managed switching element, this state will be 'disabled'.

If the vcTTpPointer attribute is present, then the transmission rate is determined by the cell rate fixed in the trafficDescriptorPackage of the vcTTPBidirectional instance referenced by the signDataLinkTp.

Note that for a complete view fo the state of a signalling data link a network view is required.

mtpMsId
This attribute is used for naming instances and identifiers the Managed Switching Element to which the the Signalling Point and Signalling Data Link path termination belong.

mtpSpId
Provides a non-zero ordinal sub-index into the signalling point table. This attribute is used for naming instances. Signalling points are contained by managed switching elements and the managed switching element id is used to name instances of signalling points.

mtpSdlId
This attribute is used for naming instances. Along with the mtpSpId, It provides a non-zero ordinal index into the table. Signalling Data Links are contained by Signalling Points which are in turn contained by Managed Switching Elements.

mtpSdlType
Provides the type of signalling data link. The type of signalling data link may be:

- 'narrowBand(1)', for a narrow-band signalling data link. In this case, the following columns are present:
  - mtpSdlLoopDelay, which specifies the nominal loop delay of the narrowband signalling data link. This value is used to select the appropriate transmission and reception method (base or preventative cyclic retransmission) and the appropriate timer profile.
  - mtpSdlTransmissionRate, which specifies the tranmission bit rate. This value is used to select the appropriate timer profile in conjunction with the mtpSdlLoopDelay field.
  - mtpSdlStmChannel, which specifies the channel within a primary multiplex facility.
- 'broadband(2)', for a SAAL signalling data link. In this case, the following columns are present:
  - mtpSdlVcTTpPointer
- 'm2pa(3)', for an M2PA signalling data link. In this case, the following columns are present:

  - `mtpSdlSctpPointer`

`mtpSdlAdjPc`
: This columnar object is an entry of the SDL table that indicates the signalling point code of the adjacent signalling point (the signalling point at the opposite end of the link). Because the Signalling Point could be operating in multiple National and International networks, the `NetworkPointCode` contains a network identifier.

`mtpSdlLoopDelay`
: Specifies the nominal loop delay (in milliseconds) associated with the narrowband signalling data link. The value zero (0), indicates that the nominal loop delay of the signalling data link is unknown and unspecified.

`mtpSdlOperationalState`
: Provides the operational state of the signalling data link following the OperationalState textual convention of the `OPENSS7-SMI-MIB` module, and according to ITU-T Rec. X.721 | ISO/IEC 10165-2.

- '`disabled(0)`', the signalling data link is not operational and is unable to provide service to a signalling link;
- '`enabled(1)`', the signalling data link is operational and is able to provide service to a signalling link.

`mtpSdlEquipmentPointer`
: This attribute is used to reference physical equipment. The constraints on this pointer are determined by the `mtpSdlType` as follows:

- '`narrowBand(1)`', a narrow-band signalling data link, this pointer references a row in an equipment table that models the narrow-band interface. Note that in some circumstances this may be the same as the signalling terminal equipment pointer.
- '`broadband(2)`', an ATM SAAL signalling data link, this pointer references a row in an equipment table that models the ATM SAAL NNI interface. Note that this is normally the same as the signalling terminal equipment pointer.
- '`m2pa(3)`', of an M2PA signalling data link, this pointer references a row in a table that models the M2PA interface. Note that under some circumstances this may be the same as the signalling terminal equipment pointer.

Where the signalling link is accessed using M2UA or M3UA, the equipment pointer references a row in an M2UA or M3UA table that models the remote interface. Note that this is normally the same as the signalling terminal equipment pointer.

`mtpSdlCIC`
: This attribute is used to reference the SS No. 7 trunk used by the datalink. Its value has to be unique within the SP's connected by the trunk.

This columnar object is an entry of the SDL table that indicates the circuit identification code (CIC) which identifies the SDL channel between the local and adjacent signalling points. This value is only necessary when the automatic allocation of signalling data links is supported. For IUT-T based networks, the CIC can have a value 0..4095, for ANSI based networks, the CIC can have a value 0..16383.

`mtpSdlTransmissionRate`
: Specifies or indicates the nominal transmission rate associated with the narrowband signalling data link.

`mtpSdlStmChannel`

This is a circuit code which uniquely identifies the circuit group or circuit facility which provides the signalling data link.

This attribute denotes the STM channel which defines the signalling datalink on the PCM transmission system. Note that the range is (1..31) for E1 and (1..24) for T1 and J1.

For broadband (SAAL) and SIGTRAN (M2PA) signalling data links, this column is not present.

`mtpSdlVcTTpPointer`

This attribute references an object class defined in I.751. It must be present for a broadband signalling data link, it must not be present for a narrowband datalink (the term broadband signalling data link signifies a data link using the SAAL NNI). The referenced instance has a mandatory relation via its upstream- and downstreamConnectivityPointers to one instance of object class vcCTPBidirectional, whose Id represents the VCI of the virtual channel used by the data link. This vcCTPBidirectional is contained in a superior object vpTTPBidirectional, which has a mandatory relation via its upstream- and downstreamConnectivityPointers to one instance of the object class vpTTPBidirectional, whose Id represents the VPI of the virtual path used by the data link.

For narrowband (TDM) and SIGTRAN (M2PA) signalling data links, this column is not present.

`mtpSdlSctpPointer`

This attribute references an SCTP association defined in RFC 4960. It must be present for an M2PA signalling data link, it must not be present for a narrow-band or broadband signalling data link (the term broadband signalling data link signifies a data link using the SAAL NNI). The referenced instance identifies the IP addresses and port numbers of the association.

For narrowband (TDM) and broadband (ATM SAAL) signalling data links, this column is not present.

`mtpSdlName`

This attribute is an additional name for instances of the signDataLinkTp managed object class. This name, when provided, must be unique within the table. An attempt to create an entry in this table with a name that is used by another entry in this table will be rejected as having an inconsistent value.

`mtpSdlRowStatus`

Provides a mechanism whereby management stations may create and delete entries in this table.

A create request is rejected if the equipmentPointer would reference equipment that does not exist. If the name package is supported: a create request with a value for the name attribute that is already used by another instance of the same object class will be rejected. A create request may be rejected if the mtpSpId index does not correspond to an existing entry in the mtpSpTable.

A delete request, or a request to take an entry out of service, is rejected as an inconsistent value if the entry is referenced by an entry in the mtpSlTable.

7.2 MTP SDL Notifications

`sdlEventLostSync`
: This notification is sent when a frame-based interface (channel) group loses frame synchronization on the line. The argument of the notification is the mtpSdlTable index.

`sdlEventSuError`
: This notification is sent when a frame-based interface (channel) group receives an SU in Error. The argument of the notification is the mtpSdlTable index.

`sdlEventTxFail`
: This notification is sent when the transmit section of the interface fails. The argument of the notification is the mtpSdlTable index.

`sdlEventRxFail`
: This notification is sent when the receive section of the interface fails. The argument of the notification is the mtpSdlTable index.

Appendix A Header Files

A.1 LMI Header File Listing

```
#ifndef __LMI_H__
#define __LMI_H__

#define LMI_PROTO_BASE            16L

#define LMI_DSTR_FIRST          ( 1L + LMI_PROTO_BASE )
#define LMI_INFO_REQ            ( 1L + LMI_PROTO_BASE )
#define LMI_ATTACH_REQ          ( 2L + LMI_PROTO_BASE )
#define LMI_DETACH_REQ          ( 3L + LMI_PROTO_BASE )
#define LMI_ENABLE_REQ          ( 4L + LMI_PROTO_BASE )
#define LMI_DISABLE_REQ         ( 5L + LMI_PROTO_BASE )
#define LMI_OPTMGMT_REQ         ( 6L + LMI_PROTO_BASE )
#define LMI_DSTR_LAST           ( 6L + LMI_PROTO_BASE )

#define LMI_USTR_LAST           (-1L - LMI_PROTO_BASE )
#define LMI_INFO_ACK            (-1L - LMI_PROTO_BASE )
#define LMI_OK_ACK              (-2L - LMI_PROTO_BASE )
#define LMI_ERROR_ACK           (-3L - LMI_PROTO_BASE )
#define LMI_ENABLE_CON          (-4L - LMI_PROTO_BASE )
#define LMI_DISABLE_CON         (-5L - LMI_PROTO_BASE )
#define LMI_OPTMGMT_ACK         (-6L - LMI_PROTO_BASE )
#define LMI_ERROR_IND           (-7L - LMI_PROTO_BASE )
#define LMI_STATS_IND           (-8L - LMI_PROTO_BASE )
#define LMI_EVENT_IND           (-9L - LMI_PROTO_BASE )
#define LMI_USTR_FIRST          (-9L - LMI_PROTO_BASE )

#define LMI_UNATTACHED          1L      /* No PPA attached, awating LMI_ATTACH_REQ */
#define LMI_ATTACH_PENDING      2L      /* Waiting for attach */
#define LMI_UNUSABLE            3L      /* Device cannot be used, STREAM in hung state */
#define LMI_DISABLED            4L      /* PPA attached, awaiting LMI_ENABLE_REQ */
#define LMI_ENABLE_PENDING      5L      /* Waiting to send LMI_ENABLE_CON */
#define LMI_ENABLED             6L      /* Ready for use, awaiting primtiive exchange */
#define LMI_DISABLE_PENDING     7L      /* Waiting to send LMI_DISABLE_CON */
#define LMI_DETACH_PENDING      8L      /* Waiting for detach */

/*
 *  LMI_ERROR_ACK and LMI_ERROR_IND reason codes
 */
#define LMI_UNSPEC              0x00000000      /* Unknown or unspecified */
#define LMI_BADADDRESS          0x00010000      /* Address was invalid */
#define LMI_BADADDRTYPE         0x00020000      /* Invalid address type */
#define LMI_BADDIAL             0x00030000      /* (not used) */
#define LMI_BADDIALTYPE         0x00040000      /* (not used) */
#define LMI_BADDISPOSAL         0x00050000      /* Invalid disposal parameter */
#define LMI_BADFRAME            0x00060000      /* Defective SDU received */
#define LMI_BADPPA              0x00070000      /* Invalid PPA identifier */
#define LMI_BADPRIM             0x00080000      /* Unregognized primitive */
#define LMI_DISC                0x00090000      /* Disconnected */
#define LMI_EVENT               0x000a0000      /* Protocol-specific event ocurred */
#define LMI_FATALERR            0x000b0000      /* Device has become unusable */
```

```
#define LMI_INITFAILED          0x000c0000      /* Link initialization failed */
#define LMI_NOTSUPP             0x000d0000      /* Primitive not supported by this device */
#define LMI_OUTSTATE            0x000e0000      /* Primitive was issued from invalid state */
#define LMI_PROTOSHORT          0x000f0000      /* M_PROTO block too short */
#define LMI_SYSERR              0x00100000      /* UNIX system error */
#define LMI_WRITEFAIL           0x00110000      /* Unitdata request failed */
#define LMI_CRCERR              0x00120000      /* CRC or FCS error */
#define LMI_DLE_EOT             0x00130000      /* DLE EOT detected */
#define LMI_FORMAT              0x00140000      /* Format error detected */
#define LMI_HDLC_ABORT          0x00150000      /* Aborted frame detected */
#define LMI_OVERRUN             0x00160000      /* Input overrun */
#define LMI_TOOSHORT            0x00170000      /* Frame too short */
#define LMI_INCOMPLETE          0x00180000      /* Partial frame received */
#define LMI_BUSY                0x00190000      /* Telephone was busy */
#define LMI_NOANSWER            0x001a0000      /* Connection went unanswered */
#define LMI_CALLREJECT          0x001b0000      /* Connection rejected */
#define LMI_HDLC_IDLE           0x001c0000      /* HDLC line went idle */
#define LMI_HDLC_NOTIDLE        0x001d0000      /* HDLC link no longer idle */
#define LMI_QUIESCENT           0x001e0000      /* Line being reassigned */
#define LMI_RESUMED             0x001f0000      /* Line has been reassigned */
#define LMI_DSRTIMEOUT          0x00200000      /* Did not see DSR in time */
#define LMI_LAN_COLLISIONS      0x00210000      /* LAN excessive collisions */
#define LMI_LAN_REFUSED         0x00220000      /* LAN message refused */
#define LMI_LAN_NOSTATION       0x00230000      /* LAN no such station */
#define LMI_LOSTCTS             0x00240000      /* Lost Clear to Send signal */
#define LMI_DEVERR              0x00250000      /* Start of device-specific error codes */

typedef signed int lmi_long;
typedef unsigned int lmi_ulong;
typedef unsigned short lmi_ushort;
typedef unsigned char lmi_uchar;

/*
 *  LOCAL MANAGEMENT PRIMITIVES
 */

/*
   LMI_INFO_REQ, M_PROTO or M_PCPROTO
 */

typedef struct {
        lmi_long lmi_primitive;         /* LMI_INFO_REQ */
} lmi_info_req_t;

/*
   LMI_INFO_ACK, M_PROTO or M_PCPROTO
 */

typedef struct {
        lmi_long lmi_primitive;         /* LMI_INFO_ACK */
        lmi_ulong lmi_version;
        lmi_ulong lmi_state;
        lmi_ulong lmi_max_sdu;
        lmi_ulong lmi_min_sdu;
        lmi_ulong lmi_header_len;
        lmi_ulong lmi_ppa_style;
        lmi_ulong lmi_ppa_length;
        lmi_ulong lmi_ppa_offset;
        lmi_ulong lmi_prov_flags;       /* provider specific flags */
        lmi_ulong lmi_prov_state;       /* provider specific state */
        lmi_uchar lmi_ppa_addr[0];
} lmi_info_ack_t;

#define LMI_VERSION_1       1
#define LMI_VERSION_2       2
#define LMI_CURRENT_VERSION LMI_VERSION_2

/*
 *  LMI provider style.
 *
 *  The LMI provider style which determines whether a provider requires an
 *  LMI_ATTACH_REQ to inform the provider which PPA user messages should be
 *  sent/received on.
 */
#define LMI_STYLE1      0x00    /* PPA is implicitly bound by open(2) */
#define LMI_STYLE2      0x01    /* PPA must be explicitly bound via STD_ATTACH_REQ */

/*
   LMI_ATTACH_REQ, M_PROTO or M_PCPROTO
 */

typedef struct {
        lmi_long lmi_primitive;         /* LMI_ATTACH_REQ */
        lmi_ulong lmi_ppa_length;
        lmi_ulong lmi_ppa_offset;
        lmi_uchar lmi_ppa[0];
} lmi_attach_req_t;

/*
   LMI_DETACH_REQ, M_PROTO or M_PCPROTO
 */

typedef struct {
        lmi_long lmi_primitive;         /* LMI_DETACH_REQ */
} lmi_detach_req_t;

/*
   LMI_ENABLE_REQ, M_PROTO or M_PCPROTO
 */

typedef struct {
        lmi_long lmi_primitive;         /* LMI_ENABLE_REQ */
        lmi_ulong lmi_rem_length;
        lmi_ulong lmi_rem_offset;
        lmi_uchar lmi_rem[0];
} lmi_enable_req_t;

/*
   LMI_DISABLE_REQ, M_PROTO or M_PCPROTO
 */

typedef struct {
        lmi_long lmi_primitive;         /* LMI_DISABLE_REQ */
} lmi_disable_req_t;

/*
   LMI_OK_ACK, M_PROTO or M_PCPROTO
 */

typedef struct {
        lmi_long lmi_primitive;         /* LMI_OK_ACK */
        lmi_long lmi_correct_primitive;
        lmi_ulong lmi_state;
} lmi_ok_ack_t;

/*
   LMI_ERROR_ACK, M_CTL
 */

typedef struct {
        lmi_long lmi_primitive;         /* LMI_ERROR_ACK */
        lmi_ulong lmi_errno;
        lmi_ulong lmi_reason;
        lmi_long lmi_error_primitive;
        lmi_ulong lmi_state;
} lmi_error_ack_t;

/*
   LMI_ENABLE_CON, M_PROTO or M_PCPROTO
 */

typedef struct {
        lmi_long lmi_primitive;         /* LMI_ENABLE_CON */
        lmi_ulong lmi_state;
} lmi_enable_con_t;

/*
   LMI_DISABLE_CON, M_PROTO or M_PCPROTO
 */

typedef struct {
        lmi_long lmi_primitive;         /* LMI_DISABLE_CON */
        lmi_ulong lmi_state;
} lmi_disable_con_t;

/*
   LMI_OPTMGMT_REQ, M_PCPROTO
 */

typedef struct {
        lmi_long lmi_primitive;         /* LMI_OPTMGMT_REQ */
        lmi_ulong lmi_opt_length;
        lmi_ulong lmi_opt_offset;
        lmi_ulong lmi_mgmt_flags;
} lmi_optmgmt_req_t;

/*
   LMI_OPTMGMT_ACK, M_PCPROTO
 */

typedef struct {
        lmi_long lmi_primitive;         /* LMI_OPMGMT_ACK */
        lmi_ulong lmi_opt_length;
        lmi_ulong lmi_opt_offset;
        lmi_ulong lmi_mgmt_flags;
} lmi_optmgmt_ack_t;

#undef LMI_DEFAULT

#define LMI_NEGOTIATE           0x0004
#define LMI_CHECK               0x0008
#define LMI_DEFAULT             0x0010
#define LMI_SUCCESS             0x0020
#define LMI_FAILURE             0x0040
#define LMI_CURRENT             0x0080
#define LMI_PARTSUCCESS         0x0100
#define LMI_READONLY            0x0200
#define LMI_NOTSUPPORT          0x0400

/*
   LMI_ERROR_IND, M_PROTO or M_PCPROTO
 */

typedef struct {
        lmi_long lmi_primitive;         /* LMI_ERROR_IND */
        lmi_ulong lmi_errno;
        lmi_ulong lmi_reason;
        lmi_ulong lmi_state;
} lmi_error_ind_t;

/*
   LMI_STATS_IND, M_PROTO
 */

typedef struct {
        lmi_long lmi_primitive;         /* LMI_STATS_IND */
        lmi_ulong lmi_interval;
        lmi_ulong lmi_timestamp;
} lmi_stats_ind_t;

/*
   LMI_EVENT_IND, M_PROTO
 */

typedef struct {
        lmi_long lmi_primitive;         /* LMI_EVENT_IND */
        lmi_ulong lmi_objectid;
        lmi_ulong lmi_timestamp;
        lmi_ulong lmi_severity;
} lmi_event_ind_t;

union LMI_primitive {
        lmi_long lmi_primitive;
        lmi_ok_ack_t ok_ack;
        lmi_error_ack_t error_ack;
        lmi_error_ind_t error_ind;
        lmi_stats_ind_t stats_ind;
        lmi_event_ind_t event_ind;
};

union LMI_primitives {
        lmi_long lmi_primitive;
        lmi_info_req_t info_req;
        lmi_info_ack_t info_ack;
        lmi_attach_req_t attach_req;
        lmi_detach_req_t detach_req;
        lmi_enable_req_t enable_req;
        lmi_disable_req_t disable_req;
        lmi_ok_ack_t ok_ack;
        lmi_error_ack_t error_ack;
        lmi_enable_con_t enable_con;
        lmi_disable_con_t disable_con;
        lmi_error_ind_t error_ind;
        lmi_stats_ind_t stats_ind;
        lmi_event_ind_t event_ind;
        lmi_optmgmt_req_t optmgmt_req;
        lmi_optmgmt_ack_t optmgmt_ack;
};

#define LMI_INFO_REQ_SIZE       sizeof(lmi_info_req_t)
#define LMI_INFO_ACK_SIZE       sizeof(lmi_info_ack_t)
#define LMI_ATTACH_REQ_SIZE     sizeof(lmi_attach_req_t)
#define LMI_DETACH_REQ_SIZE     sizeof(lmi_detach_req_t)
#define LMI_ENABLE_REQ_SIZE     sizeof(lmi_enable_req_t)
#define LMI_DISABLE_REQ_SIZE    sizeof(lmi_disable_req_t)
#define LMI_OK_ACK_SIZE         sizeof(lmi_ok_ack_t)
#define LMI_ERROR_ACK_SIZE      sizeof(lmi_error_ack_t)
#define LMI_ENABLE_CON_SIZE     sizeof(lmi_enable_con_t)
#define LMI_DISABLE_CON_SIZE    sizeof(lmi_disable_con_t)
#define LMI_ERROR_IND_SIZE      sizeof(lmi_error_ind_t)
#define LMI_STATS_IND_SIZE      sizeof(lmi_stats_ind_t)
#define LMI_EVENT_IND_SIZE      sizeof(lmi_event_ind_t)

typedef struct lmi_opthdr {
        lmi_ulong level;
        lmi_ulong name;
        lmi_ulong length;
        lmi_ulong status;
        lmi_uchar value[0];
        /*
           followed by option value
         */
} lmi_opthdr_t;

#define LMI_LEVEL_COMMON        '\0'
#define LMI_LEVEL_SDL           'd'
#define LMI_LEVEL_SDT           't'
#define LMI_LEVEL_SL            'l'
#define LMI_LEVEL_SLS           's'
#define LMI_LEVEL_MTP           'M'
```

```
#define LMI_LEVEL_SCCP          'S'
#define LMI_LEVEL_ISUP          'I'
#define LMI_LEVEL_TCAP          'T'

#define LMI_OPT_PROTOCOL        1       /* use struct lmi_option */
#define LMI_OPT_STATISTICS      2       /* use struct lmi_sta */

#endif                          /* __LMI_H__ */
```

A.2 SDLI Header File Listing

```
#ifndef __SDLI_H__
#define __SDLI_H__

/*
 *  The purpose of the SDL interface is to provide separation between the
 *  SDTI (Signalling Data Terminal Interface) which provides SS7 Signalling
 *  Data Terminal (SDT) state machine services including DAEDR, DAEDT, AERM,
 *  SUERM and EIM, and the underlying driver which provides access to the
 *  line (L1).
 */

typedef lmi_long sdl_long;
typedef lmi_ulong sdl_ulong;
typedef lmi_ushort sdl_ushort;
typedef lmi_uchar sdl_uchar;

#define SDL_PROTO_BASE                   32L

#define SDL_DSTR_FIRST                  ( 1L + SDL_PROTO_BASE)
#define SDL_BITS_FOR_TRANSMISSION_REQ   ( 1L + SDL_PROTO_BASE)
#define SDL_CONNECT_REQ                 ( 2L + SDL_PROTO_BASE)
#define SDL_DISCONNECT_REQ              ( 3L + SDL_PROTO_BASE)
#define SDL_DSTR_LAST                   ( 3L + SDL_PROTO_BASE)

#define SDL_USTR_LAST                   (-1L - SDL_PROTO_BASE)
#define SDL_RECEIVED_BITS_IND           (-1L - SDL_PROTO_BASE)
#define SDL_DISCONNECT_IND              (-2L - SDL_PROTO_BASE)
#define SDL_USTR_FIRST                  (-2L - SDL_PROTO_BASE)

#define SDL_DISCONNECTED    0
#define SDL_CONNECTED       1

/*
 *  SDLI PROTOCOL PRIMITIVES
 */

/*
 *  SDL_BITS_FOR_TRANSMISSION_REQ, M_PROTO w/ M_DATA or M_DATA
 *  -------------------------------------------------------------------------
 *  Used by the SDT to send bits to the SDL.
 */
typedef struct {
        sdl_long sdl_primitive;         /* SDL_BITS_FOR_TRANSMISSION_REQ */
} sdl_bits_for_transmission_req_t;
```

```
/*
 *  SDL_CONNECT_REQ, M_PROTO or M_PCPROTO
 *  -------------------------------------------------------------------------
 *  Used by the SDT to request that it be connected to the line.    Connection
 *  to the line might require some switching or other mecahnism.
 */
typedef struct {
        sdl_long sdl_primitive;         /* SDL_CONNECT_REQ */
        sdl_ulong sdl_flags;            /* direction flags */
} sdl_connect_req_t;

#define SDL_RX_DIRECTION        0x01
#define SDL_TX_DIRECTION        0x02

/*
 *  SDL_DISCONNECT_REQ, M_PROTO or M_PCPROTO
 *  -------------------------------------------------------------------------
 *  Used by the SDT to request that it be disconnected from the line.
 *  Disconnection from the line might require some switching or other
 *  mecahnism.
 */
typedef struct {
        sdl_long sdl_primitive;         /* SDL_DISCONNECT_REQ */
        sdl_ulong sdl_flags;            /* direction flags */
} sdl_disconnect_req_t;

/*
 *  SDL_RECEIVED_BITS_IND, M_PROTO w/ M_DATA or M_DATA
 *  -------------------------------------------------------------------------
 *  Used by the SDL to send received bits to the SDT.
 */
typedef struct {
        sdl_long sdl_primitive;         /* SDL_RECEIVED_BITS_IND */
} sdl_received_bits_ind_t;

/*
 *  SDL_DISCONNECT_IND, M_PROTO or M_PCPROTO
 *  -------------------------------------------------------------------------
 *  Used by the SDL to indicated to the SDT that it has been disconnected from
 *  the line.
 */
typedef struct {
        sdl_long sdl_primitive;         /* SDL_DISCONNECT_IND */
} sdl_disconnect_ind_t;

union SDL_primitives {
        sdl_long sdl_primitive;
        sdl_bits_for_transmission_req_t bits_for_transmission_req;
        sdl_connect_req_t connect_req;
        sdl_disconnect_req_t disconnect_req;
        sdl_received_bits_ind_t received_bits_ind;
        sdl_disconnect_ind_t disconnect_ind;
};

#define SDL_BITS_FOR_TRANSMISSION_REQ_SIZE      sizeof(sdl_bits_for_transmission_req_t)
```

```
#define SDL_CONNECT_REQ_SIZE                    sizeof(sdl_connect_req_t)
#define SDL_DISCONNECT_REQ_SIZE                 sizeof(sdl_disconnect_req_t)
#define SDL_RECEIVED_BITS_IND_SIZE              sizeof(sdl_received_bits_ind_t)
#define SDL_DISCONNECT_IND_SIZE                 sizeof(sdl_disconnect_ind_t)

#endif                          /* __SDLI_H__ */
```

Appendix B Drivers and Modules

The Signalling Data Link Interface (SDLI) is used to provide services to a number of STREAMS drivers and modules in addition to user-space applications. *OpenSS7* provides a range of STREAMS multiplexing drivers, pseudo-device drivers, and pushable modules that complement the drivers that provide signalling data link services at their service interfaces.

B.1 Drivers

Although theoretically a driver can provide the SDL interface directly, there are currently no drivers that do so. Any driver that provides the SDL interface (SDLI) can easily provide the Channel interface (CHI) instead and provide much wider use of the driver. Drivers that provide the Channel Interface (CHI) can specify the `sdl(4)` module in an `autopush(8)` specification to transparently provide both the CHI and SDLI interfaces.

B.2 Modules

B.2.1 SDL Module

The SDL module, `sdl(4)`, is a pushable STREAMS module named **`sdl`**. Its purpose is to take an *OpenSS7* Channel Interface (CHI) Stream and convert it for use as an SDL interface Stream by applications programs, drivers or modules expecting the SDLI interface. The insertion and use of this module is illustrated in Figure B.1.

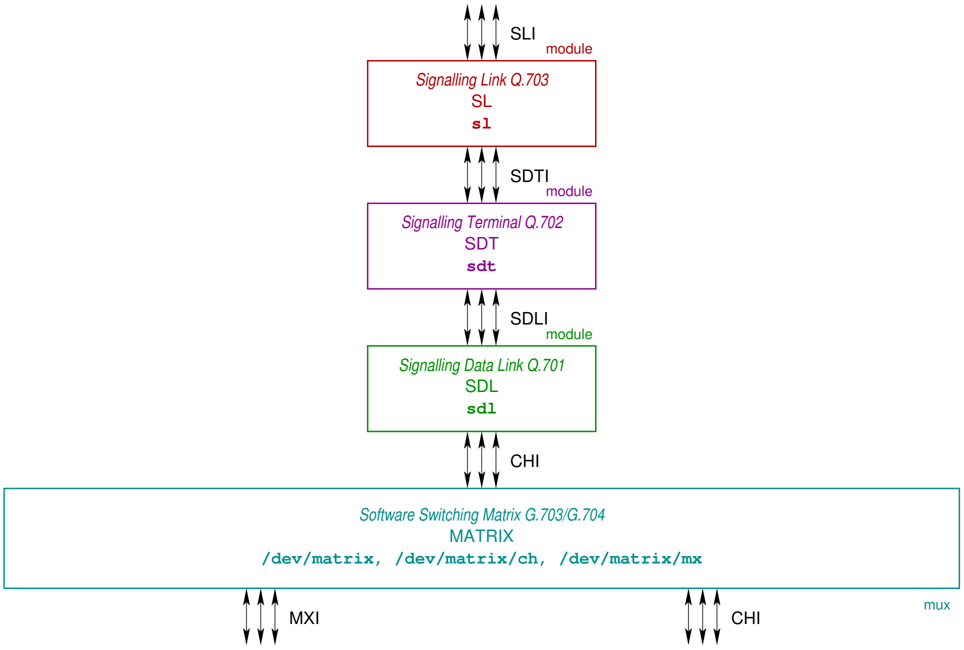
SLI
module
Signalling Link Q.703
SL
sl
SDTI
module
Signalling Terminal Q.702
SDT
sdt
SDLI
module
Signalling Data Link Q.701
SDL
sdl
CHI
Software Switching Matrix G.703/G.704
MATRIX
/dev/matrix, /dev/matrix/ch, /dev/matrix/mx
mux
MXI
CHI

Figure B.1: *SS7 Protocol Stack*

The `sdl` pushable STREAMS module accepts a Channel Inteface (CHI) at it lower service boundary and provides a Signalling Data Link Inteface (SDLI) at its upper service boundary.

Note that, as `sdl` is a pushable module, it is possible to include an `autopush(8)` specification for a driver providing the Channel Interface (CHI), to provide a specialized device minor or minor name that clones channel device layers following the SDLI approach.

The role of the SDL module in the *OpenSS7* SS7 protocol suite is illustrated in Figure B.1 and Figure C.1.

Appendix C Applications

C.1 SDLI in SS7 Protocol Suite

The signalling data link interface is an important lower layer component of the *OpenSS7* SS7 signalling stack.

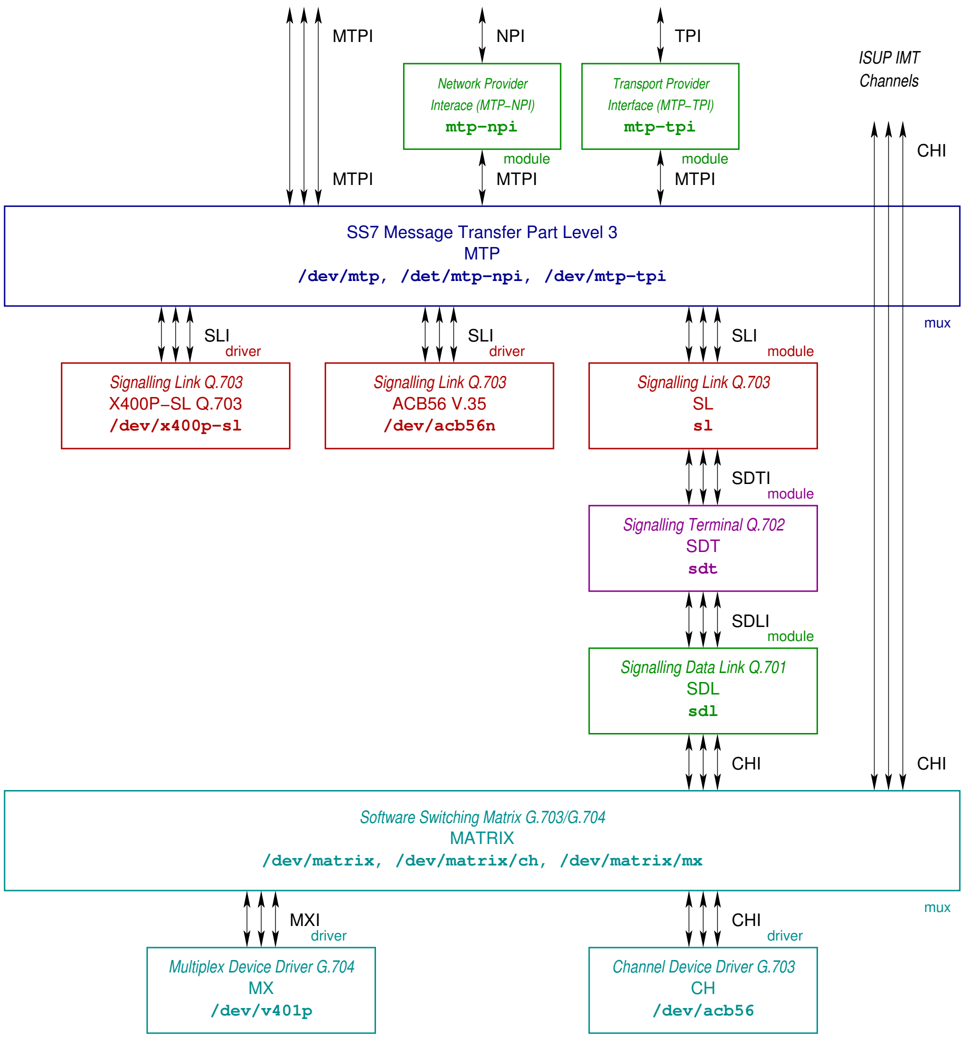

Figure C.1: *SS7 Protocol Stack*

Figure C.1 illustrates the use of the SDLI interfae specifications in the formation of the SS7 (Signalling System No. 7) protocol stack.

The SDLI interface is responsible for providing access to the signalling data links necessary for implementing signalling terminals and signalling links in accordance with ITU-T Recommendationds Q.702 and Q.703 as well as similar national standards (e.g. ANSI T1.111).

Use of the *OpenSS7* softswitch matrix a the lowest level, as illustrated in Figure C.1, provides a mechanism whereby any communications channel available to the host can be used as an SS7 link.

Due to the high performance of the *Linux Fast-STREAMS* and power of recent CPUs, it is possible to break the SS7 atack into multiple lower layers. This has the following advantages:

— Because the driver is no longer closely integrated, it is easy to reuse the same driver for different purposes (such as X.25, Frame Relay, ISDN, Voice) and not just SS7 data links.

— Drivers are no longer specific ot SS7.

— Drivers can easily be used for voice and switching.

— Devices can be share across protocol suites and applications.

— The SDLI interface can support fractional E1/T1 spans.

The advent of the high-performance *Linux Fast-STREAMS* as well as extremely powerful COTS processors, it is easily possibl to spearate protocol levels.[1] Thus, the drivers provide the generic Multiplex Interface (MXI) that provides direct access to the multiplexed spans, or the generic Multiplex Interface (MXI) to provide direct access to non-multiplexed discrete channel devices, and these generic driver interfaces can be linked under the switching matrix multiplexing driver so that a sngle upper MXI user Stream can provide access to any channel, span, or fractional span within the entire host.

[1] As it turns out, *Linux Fast-STREAMS* has such high peformance that higher levels of performance can be acheived by splitting functions into narrowly defined modules that can use STREAMS flow control to keep code paths scortchingly hot.

Appendix D Utilities

D.1 SDL Configuration Utility

`sdlconfig(8)`

Appendix E File Formats

Glossary

Signalling Data Link Service Data Unit

A grouping of SDL user data whose boundaries are preserved from one end of the signalling data link connection to the other.

Data transfer

The phase in connection and connectionless modes that supports the transfer of data between to signalling data link users.

SDL provider

The signalling data link layer protocol that provides the services of the signalling data link interface.

SDL user

The user-level application or user-level or kernel-level protocol that accesses the services of the signalling data link layer.

Local management

The phase in connection and connectionless modes in which a SDL user initializes a Stream and attaches a PPA address to the Stream. Primitives in this phase generate local operations only.

PPA

The point at which a system attaches itself to a physical communications medium.

PPA identifier

An identifier of a particular physical medium over which communication transpires.

Acronyms

AERM	Alignment Error Rate Monitor
CC	Congestion Control
DAEDR	Delimitation Alignment and Error Detection (Receive)
DAEDT	Delimitation Alignment and Error Detection (Transmit)
EIM	Errored Interval Monitor
IAC	Initial Alignment Control
ITU-T	International Telecommunications Union - Telecom Sector
LMS Provider	A provider of Local Management Services
LMS	Local Management Service
LMS User	A user of Local Management Services
LM	Local Management
LSC	Link State Control
PPA	Physical Point of Attachment
RC	Reception Control
SDLI	Signalling Data Link Interface
SDL SDU	Signalling Data Link Service Data Unit
SDLS	Signalling Data Link Service
SDL	Signalling Data Link
SDTI	Signalling Data Terminal Interface
SDTS	Signalling Data Terminal Service
SDT	Signalling Data Terminal
SLI	Signalling Link Interface
SLS	Signalling Link Service
SL	Signalling Link
SL	Signalling Link
SS7	Signalling System No. 7
TXC	Transmission Control

References

[1] ITU-T Recommendation Q.700, *Introduction to CCITT Signalling System No. 7*, March 1993, (Geneva), ITU, ITU-T Telecommunication Standardization Sector of ITU, (Previously “CCITT Recommendation”).

[2] ITU-T Recommendation Q.701, *Functional Description of the Message Transfer Part (MTP) of Signalling System No. 7*, March 1993, (Geneva), ITU, ITU-T Telecommunication Standardization Sector of ITU, (Previously “CCITT Recommendation”).

[3] ITU-T Recommendation Q.702, *Signalling System No. 7—Signalling Data Link*, March 1993, (Geneva), ITU, ITU-T Telecommunication Standardization Sector of ITU, (Previously “CCITT Recommendation”).

[4] ITU-T Recommendation Q.703, *Signalling System No. 7—Signalling Link*, March 1993, (Geneva), ITU, ITU-T Telecommunication Standardization Sector of ITU, (Previously “CCITT Recommendation”).

[5] ITU-T Recommendation Q.704, *Message Transfer Part—Signalling Network Functions and Messages*, March 1993, (Geneva), ITU, ITU-T Telecommunication Standardization Sector of ITU, (Previously “CCITT Recommendation”).

[6] Geoffrey Gerrietts; Dave Grothe, Mikel Matthews, Dave Healy, *CDI—Application Program Interface Guide*, March 1999, (Savoy, IL), GCOM, Inc.

[7] ITU-T Recommendation Q.771, *Signalling System No. 7—Functional Description of Transaction Capabilities*, March 1993, (Geneva), ITU, ITU-T Telecommunication Standardization Sector of ITU, (Previously “CCITT Recommendation”).

Licenses

All code presented in this manual is licensed under the [GNU Affero General Public License], page 101. The text of this manual is licensed under the [GNU Free Documentation License], page 111, with no invariant sections, no front-cover texts and no back-cover texts. Please note, however, that it is just plain wrong to modify statements of, or attribute statements to, the Author or *OpenSS7 Corporation*.

GNU Affero General Public License

Preamble

The GNU Affero General Public License is a free, copyleft license for software and other kinds of works, specifically designed to ensure cooperation with the community in the case of network server software.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, our General Public Licenses are intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

Developers that use our General Public Licenses protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License which gives you legal permission to copy, distribute and/or modify the software.

A secondary benefit of defending all users' freedom is that improvements made in alternate versions of the program, if they receive widespread use, become available for other developers to incorporate. Many developers of free software are heartened and encouraged by the resulting cooperation. However, in the case of software used on network servers, this result may fail to come about. The GNU General Public License permits making a modified version and letting the public access it on a server without ever releasing its source code to the public.

The GNU Affero General Public License is designed specifically to ensure that, in such cases, the modified source code becomes available to the community. It requires the operator of a network server to provide the source code of the modified version running there to the users of that server. Therefore, public use of a modified version, on a publicly accessible server, gives the public access to the source code of the modified version.

An older license, called the Affero General Public License and published by Affero, was designed to accomplish similar goals. This is a different license, not a version of the Affero GPL, but Affero has released a new version of the Affero GPL which permits relicensing under this license.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions

0. Definitions.

   "This License" refers to version 3 of the GNU Affero General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

   To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

   An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

   The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

   The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

   When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   a. The work must carry prominent notices stating that you modified it, and giving a relevant date.

b. The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c. You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d. If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c. Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d. Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e. Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

   "Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

   When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a. Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b. Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c. Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d. Limiting the use for publicity purposes of names of licensors or authors of the material; or

e. Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f. Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

   You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

    You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

    A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

    A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

    Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

    In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

    If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others’ Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Remote Network Interaction; Use with the GNU General Public License.

    Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software. This Corresponding Source shall include the Corresponding Source for any work covered by version 3 of the GNU General Public License that is incorporated pursuant to the following paragraph.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the work with which it is combined will remain governed by version 3 of the GNU General Public License.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU Affero General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU Affero General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU Affero General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU Affero General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and a brief idea of what it does.
Copyright (C) year name of author

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as published by
the Free Software Foundation, either version 3 of the License, or (at
your option) any later version.

This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see http://www.gnu.org/licenses/.
```

Also add information on how to contact you by electronic and paper mail.

If your software can interact with users remotely through a network, you should also make sure that it provides a way for users to get its source. For example, if your program is a web application, its interface could display a "Source" link that leads users to an archive of the code. There are many ways you could offer source, and different solutions will be better for different programs; see section 13 for the specific requirements.

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU AGPL, see http://www.gnu.org/licenses/.

GNU Free Documentation License

GNU FREE DOCUMENTATION LICENSE
Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “publisher” means any person or entity that distributes copies of the Document to the public.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

   It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

   A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

   B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

   C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

   D. Preserve all the copyright notices of the Document.

   E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

   F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements.”

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

    "Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

    "CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

    "Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

    An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

    The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ''GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index