Signalling Link Interface (SLI) Specification

Signalling Link Interface (SLI) 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 Link Interface (SLI) for OpenSS7. It contains recommendations on software architecture as well as platform and system applicability of the Signalling Link Interface (SLI). It provides abstraction of the Signalling Link (SL) interface to these components as well as providing a basis for Signalling Link control for other Signalling 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:

Short Contents

Table of Contents

List of Figures

List of Tables

Preface

Notice

Software in this document and related software is released under the AGPL (see [GNU Affero General Public License], page 157). Please note, however, that there are different licensing terms for some of the manual package and some of the documentation. Consult permission notices contained in the documentation of those components for more information.

This document is released under the FDL (see [GNU Free Documentation License], page 167) with no invariant sections, no front-cover texts and no back-cover texts.

Abstract

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

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

Purpose

The purpose of this document is to provide technical documentation of the Signalling Link Interface (SLI). 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 Link Interface (SLI) 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 Link Interface (SLI). This document is intended to provide information for writers of OpenSS7 Signalling Link Interface (SLI) applications as well as writers of OpenSS7 Signalling Link Interface (SLI) Users.

Audience

The audience for this document is software developers, maintainers and users and integrators of the Signalling Link Interface (SLI). 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: sli.texi,v $
Revision 1.1.2.2  2011-02-07 02:21:46  brian
- updated manuals

Revision 1.1.2.1  2009-06-21 10:56:49  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 ITU-T Signalling Link Interface (SLI) definition. The Signalling Link Interface (SLI) enables the user of a a signalling link service to access and use any of a variety of conforming signalling link providers without specific knowledge of the provider's protocol. The service interface is designed to support any network signalling link protocol and user signalling link protocol. This interface only specifies access to signalling link service providers, and does not address issues concerning signalling link management, protocol performance, and performance analysis tools.

This specification assumes that the reader is familiar with ITU-T state machines and signalling 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 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 SL User
A SL-User that initiates a Signalling Link.

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

ISO International Organization for Standardization

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

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

SLI Signalling Link Interface

TIDU	Signalling Link Interface Data Unit
TSDU	Signalling 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 Link Layer

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

2.1 Model of the SLI

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

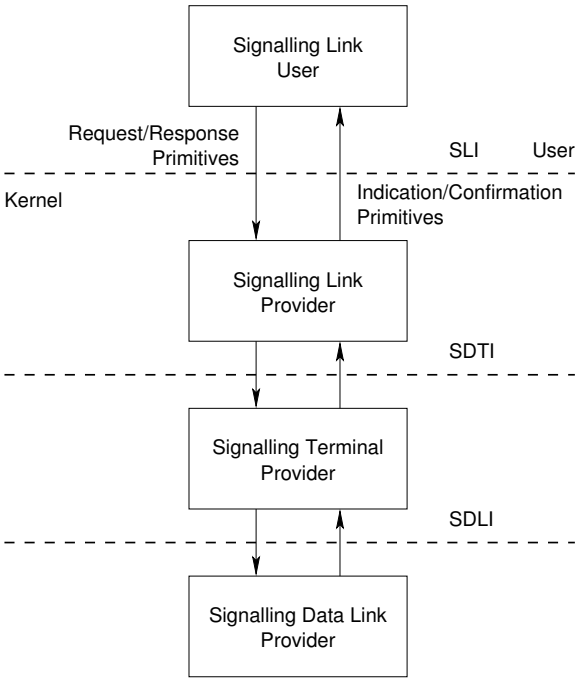

Figure 2.1: *Model of the SLI*

The SLI allows the SLS provider to be configured with any signalling link layer user (such as a signalling link application) that also conforms to the SLI. A signalling link layer user can also be a user program that conforms to the SLI and accesses the SLS provider via `putmsg(2s)` and `getmsg(2s)` system calls. The typical configuration, however, is to link a signalling link stream beneath a message transfer part multiplexing driver.

2.2 SLI Services

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

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 33. The local management services interface is defined by the `ss7/lmi.h` header file (see Appendix A [LMI Header File Listing], page 133).

2.2.2 Protocol

Protocol services are listed in Table 2.2.

Phase	Service	Primitives
Initialization	Power On	SL_POWER_ON_REQ,
	Emergency	SL_EMERGENCY_REQ, SL_EMERGENCY_CEASES_REQ,
	Start	SL_START_REQ, SL_IN_SERVICE_IND,
	Stop	SL_OUT_OF_SERVICE_IND, SL_STOP_REQ,
Data Transfer	Data Transfer	SL_PDU_REQ, SL_PDU_IND
Congestion	Transmit Congestion	SL_LINK_CONGESTED_IND, SL_LINK_CONGESTION_CEASED_IND
	Receive Congestion	SL_CONGESTION_DISCARD_REQ, SL_CONGESTION_ACCEPT_REQ, SL_NO_CONGESTION_REQ
Restoration	BSNT Retrieval	SL_RETRIEVE_BSNT_REQ, SL_BSNT_IND, SL_BSNT_NOT_RETRIEVABLE_IND
	Buffer Updating	SL_RETRIEVAL_REQUEST_AND_FSNC_REQ, SL_RETRIEVED_MESSAGE_IND, SL_RETRIEVAL_COMPLETE_IND, SL_RETRIEVAL_NOT_POSSIBLE_IND
	Buffer Clearing	SL_CLEAR_BUFFERS_REQ, SL_CLEAR_RTB_REQ, SL_RB_CLEARED_IND, SL_RTB_CLEARED_IND
Processor Outage	Local Processor Outage	SL_LOCAL_PROCESSOR_OUTAGE_REQ, SL_LOCAL_PROCESSOR_OUTAGE_IND, SL_RESUME_REQ, SL_LOCAL_PROCESSOR_RECOVERED_IND
	Remote Processor Outage	SL_REMOTE_PROCESSOR_OUTAGE_IND, SL_REMOTE_PROCESSOR_RECOVERED_IND, SL_CONTINUE_REQ
Options Management	Options Management	SL_OPTMGMT_REQ, SL_OPTMGMT_ACK
Event Notification	Event Notification	SL_NOTIFY_REQ, SL_NOTIFY_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 70. The protocol services interface is defined by the `ss7/sli.h` header file (see Appendix B [SLI Header File Listing], page 141).

2.3 Purpose of the SLI

The SLI is typically implemented as a device driver controlling an intelligent protocol controller 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 SS7 HDLC mode, where a data stream can be exchanged between the driver and the medium. The SLI provides and inteface that, once implemented as a driver for a new device, can provide complete and verified SS7 signalling

link capabilities by linking under a generic MTP (Message Transfer Part) multiplex driver an open device stream.
This allows MTP drivers to be verified independently for correct operation and then simply used for all manner of new device drivers that can implement the SLI interface.

3 SLI 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.

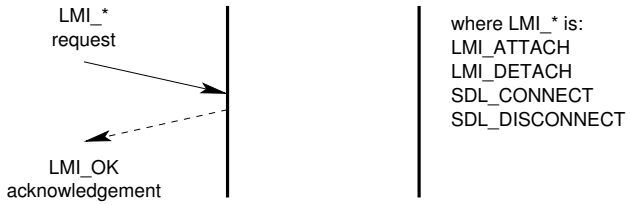

Figure 3.1: *Message Flow: Successful Acknowledgement Service*

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.

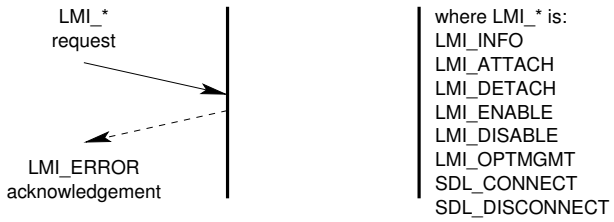

Figure 3.2: *Message Flow: Unsuccessful Acknowledgement Service*

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.

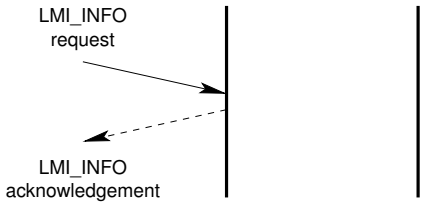

Figure 3.3: *Message Flow: Successful Information Reporting Service*

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.

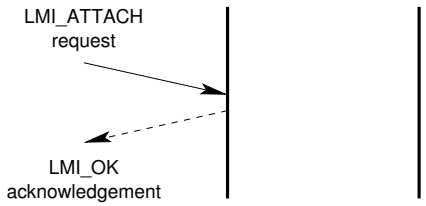

Figure 3.4: *Message Flow: Successful Attachment Service*

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.

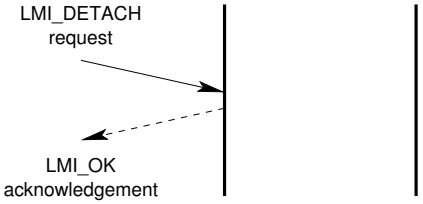

Figure 3.5: *Message Flow: Successful Detachment Service*

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.

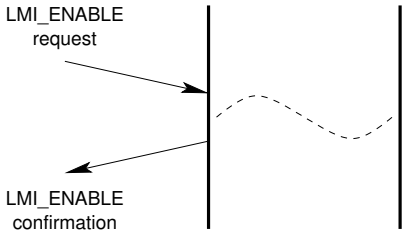

Figure 3.6: *Message Flow: Successful Enable Service*

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.

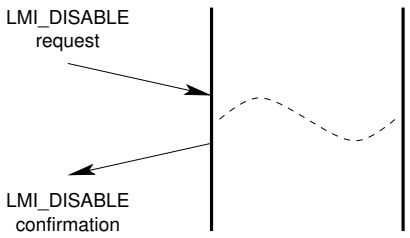

Figure 3.7: *Message Flow: Successful Disable Service*

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.

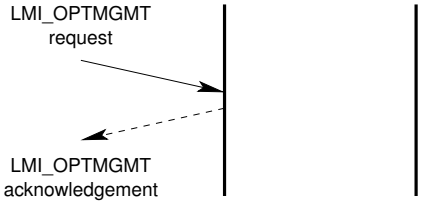

Figure 3.8: *Message Flow: Successful Options Management Service*

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.

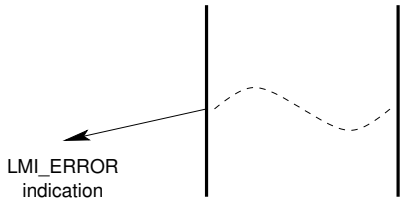

Figure 3.9: *Message Flow: Successful Error Reporting Service*

3.1.7 Statistics Reporting Service

- `LMI_STATS_IND`:

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

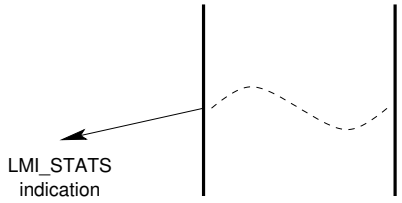

Figure 3.10: *Message Flow: Successful Statistics Reporting Service*

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.

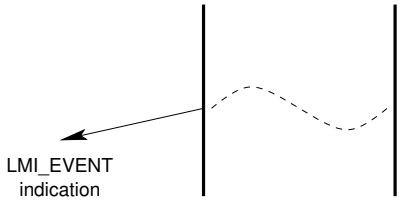

Figure 3.11: *Message Flow: Successful Event Reporting Service*

3.2 Protocol Services

Protocol services are specific to the Signalling Link interface. These services consist of initialization of the link and preparation for the transfer of signal units, the transfer of signal units, transmit and receive congestion control, BSNT retrieval, buffer updating, buffer clearing, local processor outage, remote processor outage, link options management and management event notification.

The service primitives that implement the protocol services are described in detail in Section 4.2 [Protocol Service Primitives], page 70.

3.2.1 Link Initialization Services

The link initialization services provide the SLS user with the ability to power on the terminal, set emergency status, start the signalling link and stop the signalling link. The service primitives that implement the link initialization services are described in detail in Section 4.2.1 [Link Initialization Service Primitives], page 70.

3.2.1.1 Power On Service

The power on service provides the SLS user with the ability to power on the signalling data terminal. The signalling data terminal must be powered on at least once before the link can be started.

- `SL_POWER_ON_REQ`: The `SL_POWER_ON_REQ` message is used by the SLS user to request that the SLS provider power on the signalling data terminal. If the signalling data terminal does not require power (such as a software module), this serves to initialize the signalling data terminal functions.

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

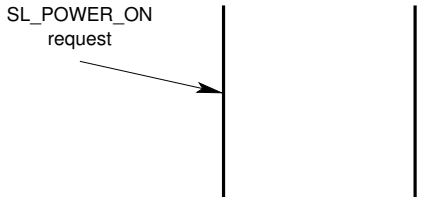

Figure 3.12: *Message Flow: Successful Power On Service*

3.2.1.2 Emergency Service

The emergency service provides the SLS user with the ability to specify whether normal or emergency alignment procedures should take effect on the current or next alignment procedure. Emergency alignment procedures have a shorter duration (short proving period) than normal alignment procedures. Some SS7 protocol variants (TTC) always use emergency alignment procedures and are not affected by this service.

- `SL_EMERGENCY_REQ`: The `SL_EMERGENCY_REQ` message is used by the SLS user to request that the emergency alignment procedure should take effect on the current or next alignment of the signalling link.

- `SL_EMERGENCY_CEASES_REQ`: The `SL_EMERGENCY_CEASES_REQ` message is used by the SLS user to request that the normal alignment procedure should take effect on the current or next alignment of the signalling link.

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

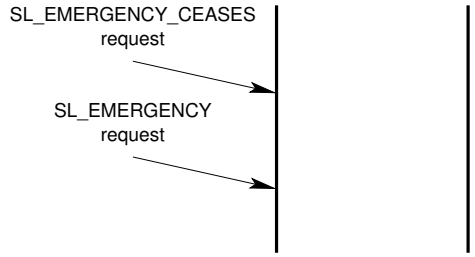

Figure 3.13: *Message Flow: Successful Emergency Service*

3.2.1.3 Start Service

The start service provides the SLS user with the ability to align the signalling link and have it placed into service. The start service must be successfully invoked on both sides of the signalling link before the signalling link is able to exchange message signal units.

- `SL_START_REQ`: The `SL_START_REQ` message is used by the SLS user to request that the signalling link be aligned a placed into service.

- `SL_IN_SERVICE_IND`: The `SL_IN_SERVICE_IND` message is used by the SLS provider to indicate that the signalling link has been successfully aligned and has been placed into service at Level 2.

A successful invocation of the start service is illustrated in Figure 3.14.

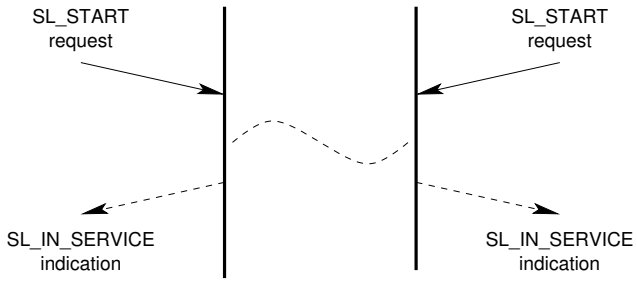

Figure 3.14: *Message Flow: Successful Start Service*

A unsuccessful invocation of the start service is illustrated in Figure 3.15.

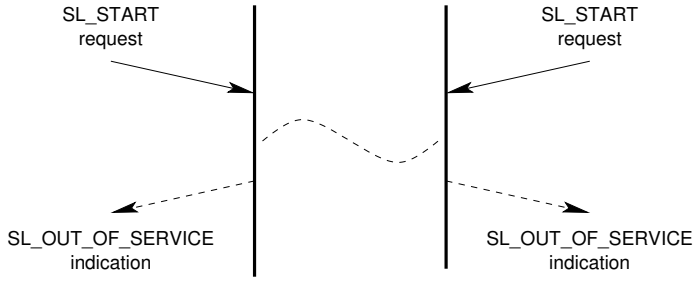

Figure 3.15: *Message Flow: Unsuccessful Start Service*

3.2.1.4 Stop Service

The stop service provides the SLS user and provider with the ability to take a signalling link out of service. Once the stop service has successfully completed, the signalling link is no longer able to exchange message signal units.

- `SL_STOP_REQ`: The `SL_STOP_REQ` message is used by the SLS user to request that the signalling link be taken out of service.
- `SL_OUT_OF_SERVICE_IND`: The `SL_OUT_OF_SERVICE_IND` message is used by the SLS provider to indicate that the signalling link has been taken out of service by the SLS provider.

A successful invocation of the stop service is illustrated in Figure 3.16.

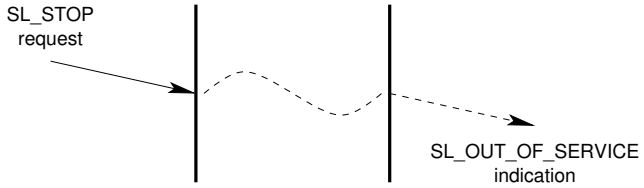

Figure 3.16: *Message Flow: Successful Stop Service*

3.2.2 Data Transfer Service

The data transfer service provides the SLS user with the ability to exchange message signal units on the signalling link. The service primitives that implement the data transfer service are described in detail in Section 4.2.2 [Data Transfer Service Primitives], page 83.

- `SL_PDU_REQ`: The `SL_PDU_REQ` message is used by the SLS user to request that a message signal unit be sent on the signalling link.
- `SL_PDU_IND`: The `SL_PDU_IND` message is used by the SLS provider to indicate that a message signal unit has been received on the signalling link.

A successful invocation of the data transfer service is illustrated in Figure 3.17.

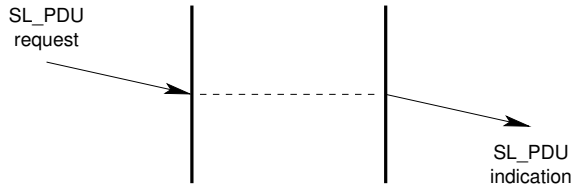

Figure 3.17: *Message Flow: Successful Data Transfer Service*

3.2.3 Congestion Services

The congestion services provide the SLS user with the ability to invoke a receive congestion policy. They also provide the SLS provider with the ability to indicate transmit congestion levels. The service primitives that implement the congestion services are described in detail in Section 4.2.3 [Congestion Service Primitives], page 85.

3.2.3.1 Transmit Congestion Service

The transmit congestion service provides the SLS provider with the ability to indicate transmit congestion (and corresponding levels) to the SLS user. There are 4 levels of congestion, 0, 1, 2 and 3. Each congestion level has an onset threshold and an abatement threshold. When the transmit buffer occupancy exceeds the onset threshold for the level, congestion is indicated at that level. When the transmit buffer occupancy falls below the abatement threshold for the level, congestion abatement is indicated. Some SS7 protocol variants do not have congestion levels and only signal the presence or lack of congestion.

When congestion is indicated at a level, the SLS user should discard messages that have a message priority that is less than the level at which congestion has been indicated.

- `SL_LINK_CONGESTED_IND`: The `SL_LINK_CONGESTED_IND` message is used by the SLS provider to indicate that congestion onset has occurred for the congestion level indicated in the message.
- `SL_LINK_CONGESTION_CEASED_IND`: The `SL_LINK_CONGESTION_CEASED_IND` message is used by the SLS provider to indicate that congestion abatement has occurred for the congestion level indicated in the message.

A successful indication of the transmit congestion service is illustrated in Figure 3.18.

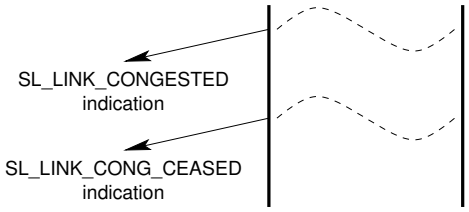

Figure 3.18: *Message Flow: Successful Transmit Congestion Service*

3.2.3.2 Receive Congestion Service

The receive congestion service provides the SLS user with the ability to specify that receive congestion is in effect or has abated and the policy to use for received message signal units under congestion. A discard policy indicates that received message signal units should be discarded (and not acknowledged); receive congestion is signalled to the sending side of the signalling link. An accept policy indicates that received message signal units should not be discarded and should be acknowledged; receive congestion is signalled to the sending side of the signalling link. When receive congestion abates, the abatement of receive congestion is signalled to the sending side of the signalling link.

The SLS provider may also perform its own receive congestion onset, abatement and policy. The SLS provider does not indicate its current receive congestion level or policy to the SLS user.

- `SL_NO_CONGESTION_REQ`: The `SL_NO_CONGESTION_REQ` message is used by the SLS user to specify that receive congestion has abated and that receive congestion should no longer be signalled to the sending side of the signalling link.
- `SL_CONGESTION_ACCEPT_REQ`: The `SL_CONGESTION_ACCEPT_REQ` message is used by the SLS user to specify that receive congestion has onset and that receive congestion should be signalled to the sending side of the signalling link. The congestion policy is an accept policy that allows message signal units to continue to be delivered to the SLS user and acknowledged to the remote end of the signalling link.
- `SL_CONGESTION_DISCARD_REQ`: The `SL_CONGESTION_DISCARD_REQ` message is used by the SLS user to specify that receive congestion has onset and that receive congestion should be signalled to the sending side of the signalling link. The congestion policy is a discard policy that requires the SLS provider to discard message signal units without delivering them to the SLS user and they are not to be acknowledged to the remote end of the signalling link.

A successful invocation of the receive congestion service is illustrated in Figure 3.19.

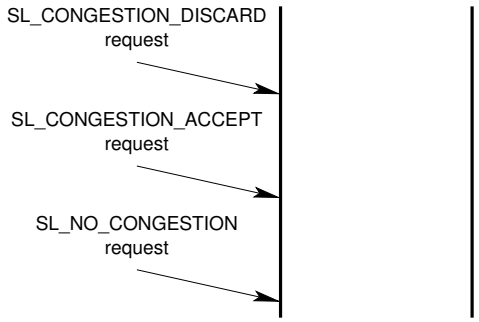

Figure 3.19: *Message Flow: Successful Receive Congestion Service*

3.2.4 Restoration Services

Restoration services consist of the services necessary to change over a link, update its buffers, and clearing any unnecessarily old MSUs from the receive buffer or retransmission buffer. The service primitives that implement the restoration services are detailed in Section 4.2.4 [Restoration Service Primitives], page 95.

3.2.4.1 BSNT Retrieval Service

The BSNT retrieval service is a somewhat optional service in support of the sequenced changeover procedure of the Message Transfer Part. It is '`somewhat`' optional due to the possibility that time-controlled changeover is always used, per ETSI ETS 300 008-1.

- `SL_RETRIEVE_BSNT_REQ`: The `SL_RETRIEVE_BSNT_REQ` message is used by the SLS user to request that the SLS provider indicate the last transmitted backward sequence number (BSNT).
- `SL_BSNT_IND`: The `SL_BSNT_IND` message is used by the SLS provider to indicate the last transmitted backward sequence number (BSNT) when requested by the SLS user with a `SL_RETRIEVE_BSNT_REQ` message.
- `SL_BSNT_NOT_RETRIEVABLE_IND`: The `SL_BSNT_NOT_RETRIEVABLE_IND` message is used by the SLS provider to indicate that the last transmitted backward sequence number (BSNT) is not available when requested by the SLS user with a `SL_RETRIEVE_BSNT_REQ` message. This may be due to hardware or other failures.

A successful invocation of the BSNT retrieval service is illustrated in Figure 3.20.

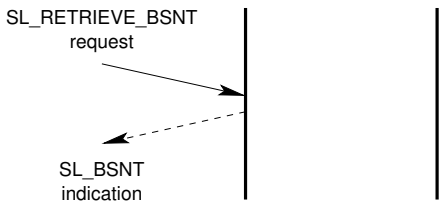

Figure 3.20: *Message Flow: Successful BSNT Retreival Service*

An unsuccessful invocation of the BSNT retrieval service is illustrated in Figure 3.21.

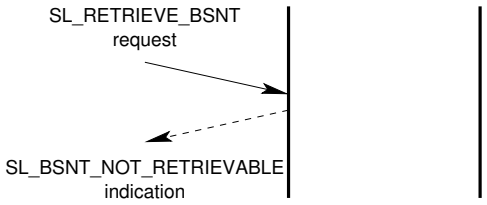

Figure 3.21: *Message Flow: Unsuccessful BSNT Retrieval Service*

3.2.4.2 Buffer Updating Service

The buffer updating service provides the SLS user with the ability to update the retransmission buffer and collect messages that have not been successfully received by the remote side of the signalling link during a sequenced changeover procedure. The SLS user specifies the FSNC (the forward sequence number confirmed received by the remote end of the signalling link). The SLS provider uses the

FSNC to purge successfully received messages from the retransmission buffer and then indicates the remaining contents of the retransmission buffer and the transmission buffer to the SLS user.

The SLS user may also clear the retransmission buffer using the buffer clearing service before retrieving messages. This this case, the messages retrieved by the SLS provider will be the contents of the transmission buffer. The combination of the two services are used to perform the time controlled changeover procedure.

- `SL_RETRIEVAL_REQUEST_AND_FSNC_REQ`: The `SL_RETRIEVAL_REQUEST_AND_FSNC_REQ` message is used by the SLS user to request the SLS provider update the retransmission buffer to reflect the value of the specified FSNC and retrieve and indicate the contents of the updated retransmission buffer followed by the contents of the transmission buffer to the SLS user.

- `SL_RETRIEVED_MESSAGE_IND`: The `SL_RETRIEVED_MESSAGE_IND` message is used by the SLS provider to indicate one message from the retransmission buffer or transmission buffer.

- `SL_RETRIEVAL_COMPLETE_IND`: The `SL_RETRIEVAL_COMPLETE_IND` message is used by the SLS provider to indicate that the retrieval of messages from the retransmission buffer and transmission buffer is complete.

- `SL_RETRIEVAL_NOT_POSSIBLE_IND`: The `SL_RETRIEVAL_NOT_POSSIBLE_IND` message is used by the SLS provider to indicate that the updating of the retransmission buffer to the specified FSNC and retrieval of messages from the retransmission buffer and transmission buffer is not possible. This may be due to hardware failure.

A successful invocation of the buffer updating service is illustrated in Figure 3.22.

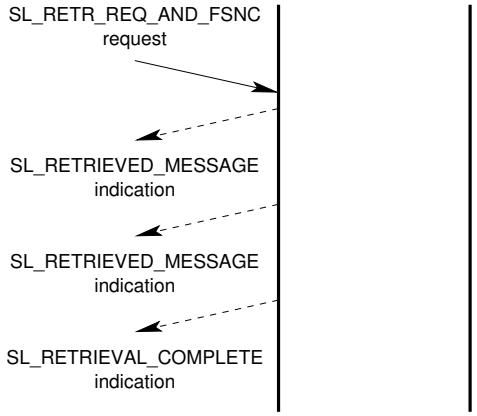

Figure 3.22: *Message Flow: Successful Buffer Updating Service*

An unsuccessful invocation of the buffer updating service is illustrated in Figure 3.23.

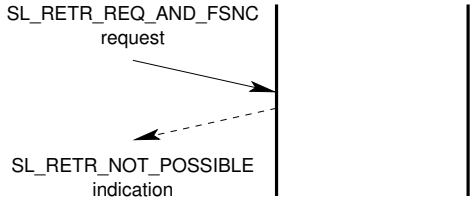

Figure 3.23: *Message Flow: Unsuccessful Buffer Updating Service*

3.2.4.3 Buffer Clearing Service

The buffer clearing service provides the SLS user with the ability to request that all message buffers be cleared (receive buffer, retransmission buffer, transmission buffer) and that the SLS provider indicate when the receive and retransmission buffer are cleared. It also provides the SLS user with the ability to clear only the retransmission buffer and receive and indication when the buffer is cleared.

Clearing of all buffers is performed when the signalling link has been blocked (local or remote processor outage) for a long duration and messages contained in the buffer are too old to be processed.

Clearing of the retransmission buffer is performed as part of the time-controlled changeover procedure, when the value of the FSNC has not been received in a sequenced changeover message from the adjacent signalling point.

- `SL_CLEAR_BUFFERS_REQ`: The `SL_CLEAR_BUFFERS_REQ` message is used by the SLS user to request that all message buffers (receive, retransmit, transmit) be cleared.
- `SL_CLEAR_RTB_REQ`: The `SL_CLEAR_RTB_REQ` message is used by the SLS user to request that only the retransmission buffer be cleared as part of a time-controlled changeover procedure.
- `SL_RB_CLEARED_IND`: The `SL_RB_CLEARED_IND` message is used by the SLS provider to indicate when the receive buffer has been successfully cleared.
- `SL_RTB_CLEARED_IND`: The `SL_RTB_CLEARED_IND` message is used by the SLS provider to indicate when the retransmission buffer has been successfully cleared.

A successful invocation of the buffer clearing service is illustrated in Figure 3.24 and Figure 3.25.

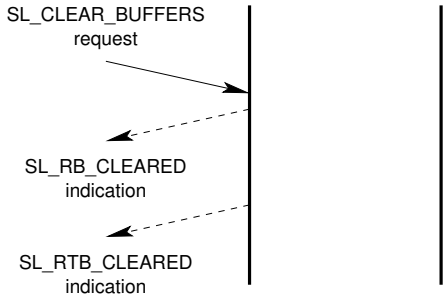

Figure 3.24: *Message Flow: Successful Buffer Clearing Service*

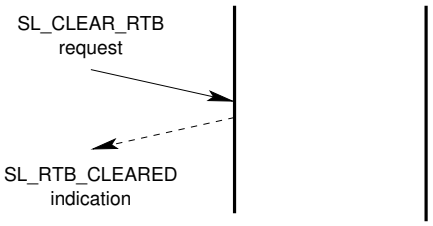

Figure 3.25: *Message Flow: Successful Buffer Clearing Service*

3.2.5 Processor Outage Services

The processor outage services provide the SLS user with the ability to request a local processor outage as well as being informed of a local or remote processor outage. The service primitives that implement the processor outage services are described in detail in Section 4.2.5 [Processor Outage Service Primitives], page 112.

3.2.5.1 Local Processor Outage Service

The local processor outage service provides the SLS user with the ability to both request a local processor outage as well as be informed of a local processor outage. Local processor outage occurs when the SLS user is unable to pass message signal units for transmission or accept received message signal units, or the SLS provider is unable to deliver received message signal units or accept message signal units for transmission. Local processor outage conditions can exist independently within the SLS user and within the SLS provider.

- `SL_LOCAL_PROCESSOR_OUTAGE_REQ`: The `SL_LOCAL_PROCESSOR_OUTAGE_REQ` message is used by the SLS user to specify that a local processor outage condition exists due to a condition within the SLS user.
- `SL_LOCAL_PROCESSOR_OUTAGE_IND`: The `SL_LOCAL_PROCESSOR_OUTAGE_IND` message is used by the SLS provider to indicate that a local processor outage condition exists due to a condition within the SLS provider.
- `SL_RESUME_REQ`: The `SL_RESUME_REQ` message is used by the SLS use to specify that a local processor outage condition no longer exists within the SLS user.
- `SL_LOCAL_PROCESSOR_RECOVERED_IND`: The `SL_LOCAL_PROCESSOR_RECOVERED_IND` message is used by the SLS provider to indicate that a local processor outage condition no longer exists within the SLS provider.

A successful invocation of the local processor outage service is illustrated in Figure 3.26.

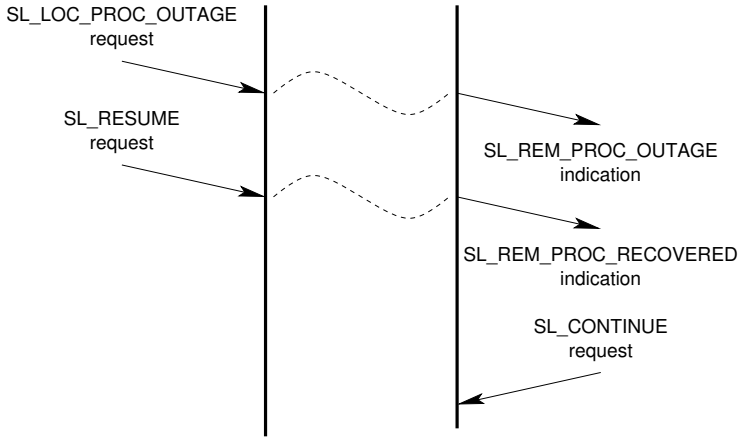

Figure 3.26: *Message Flow: Successful Processor Outage Service*

3.2.5.2 Remote Processor Outage Service

The remote processor outage service provides the SLS user with the ability to be informed of remote processor outage conditions. Remote processor outage occurs when the remote SLS user is experiencing a local processor outage. Remote processor outage conditions can exist independent of local processor outage conditions.

- `SL_REMOTE_PROCESSOR_OUTAGE_IND`: The `SL_REMOTE_PROCESSOR_OUTAGE_IND` message is used by the SLS provider to indicate that a remote processor outage condition exists.

- `SL_REMOTE_PROCESSOR_RECOVERED_IND`: The `SL_REMOTE_PROCESSOR_RECOVERED_IND` message is used by the SLS provider to indicate that a remote processor has recovered.

- `SL_CONTINUE_REQ`: The `SL_CONTINUE_REQ` message is used by the SLS user to request that a signalling link continue from where it left off after a remote processor has recovered.

A successful indication of the remote processor outage service is illustrated in Figure 3.27.

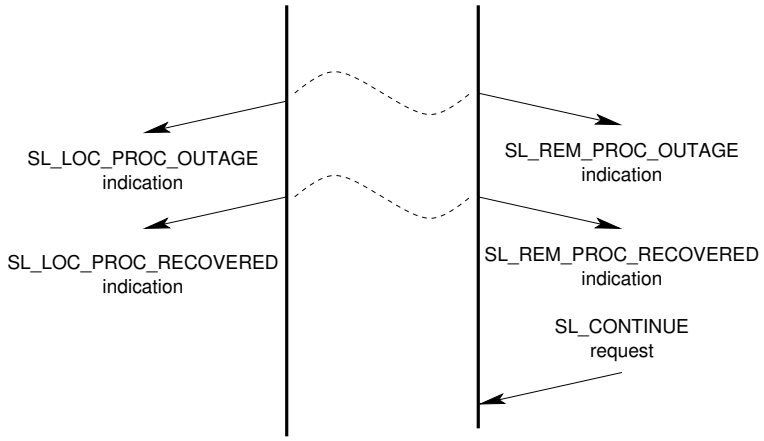

Figure 3.27: *Message Flow: Successful Processor Outage Service*

3.2.6 Link Option Management Service

The link option management service provides the SLS user with the ability to alter signalling link options. The service primitives that implement the link option management services are described in detail in Section 4.2.6 [Link Option Management Service Primitives], page 122.

- `SL_OPTMGMT_REQ`: The `SL_OPTMGMT_REQ` message is used by the SLS user to request that link options be managed.
- `SL_OPTMGMT_ACK`: The `SL_OPTMGMT_ACK` message is used by the SLS provider to acknowledge link option management actions.

A successful invocation of the link options management service is illustrated in Figure 3.28.

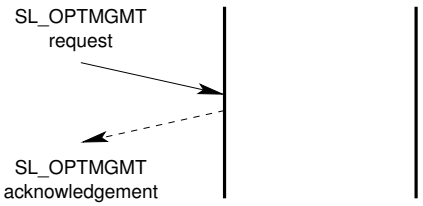

Figure 3.28: *Message Flow: Successful Link Options Management Service*

3.2.7 Event Notification Service

The event notification service provides the SLS user with the ability to register with the SLS provider to receive provider-specific event notifications. Event notifications normally correspond to management indications on the SS7 signalling link. The service primitives that implement the event notification services are described in detail in Section 4.2.7 [Event Notification Service Primitives], page 128.

- `SL_NOTIFY_REQ`: The `SL_NOTIFY_REQ` message is used by the SLS user to register with the SLS provider to receive specified event notifications.
- `SL_NOTIFY_IND`: The `SL_NOTIFY_IND` message is used by the SLS provider to indicate the occurrence of registered events to the SLS user.

A successful invocation of the event notification service is illustrated in Figure 3.29.

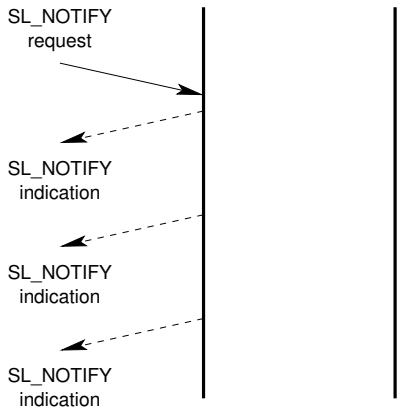

Figure 3.29: *Message Flow: Successful Event Notification Service*

4 SLI 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 thee 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 request 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 providers 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 by 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 associated 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 request 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 request 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 which 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 ore 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 follows 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 Link interface protocol. Protocol service primitives provide the SLS user with the ability to initialize the link, transfer data on the link, request and receive reports of receive and transmit congestion, restore failed signalling links, handle processor outage conditions, manage options and register for and receive event notifications.

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

4.2.1 Link Initialization Service Primitives

The link initialization primitives permit the SLS user to power on the signalling data terminal, specify emergency or normal alignment, start the signalling link and bring it into service, and stop the signalling link or be informed of link failures.

These service primitives implement the link initialization services (see Section 3.2.1 [Link Initialization Services], page 19).

4.2.1.1 SL_POWER_ON_REQ

Description

The SLS user originated service primitive request that the SLS provider power on the signalling data terminal. Not all signalling data terminals can be powered on independent of the existence of the signalling link interface. Software signalling data terminals will mark idle on signalling links until they are powered on, after which they will idle FISUs.

Format

The power on service primitive consists of one `M_PROTO` message block, structured as follows:

```
typedef struct {
    sl_long sl_primitive;
} sl_power_on_req_t;
```

Parameters

The power on service primitive contains the following parameters:

sl_primitive
: Specifies the service primitive type. Always `SL_POWER_ON_REQ`.

State

This primitive is only valid in the `LMI_ENABLED` management state. This primitive is valid in the `SL_STATE_POWER_OFF` link state; however, when issued in another link state the primitive is ignored and does not generate a non-fatal error.

New State

The new link state is `SL_STATE_OUT_OF_SERVICE`.

Rules

Response

The power on service primitive does not require receipt acknowledgement from the SLS provider.

- **Successful:** When successful, the power on service primitive does not require acknowledgement.
- **Unsuccessful (non-fatal errors):** When unsuccessful, the SLS provider indicates failure using an `LMI_ERROR_ACK` primitive containing the error.

Note that the SLS provider should ignore this primitive, and not generate a non-fatal error, when the management interface is in the `LMI_ENABLED` state and the link state is other than `SL_STATE_POWER_OFF`.

Reason for Failure

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

`[LMI_UNSPEC]`
: Unknown or unspecified.

`[LMI_DISC]`
: Disconnected.

`[LMI_EVENT]`
: Protocol-specific event occurred.

`[LMI_FATALERR]`
: Device has become unusable.

`[LMI_INITFAILED]`
: Link initialization failed.

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

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

`[LMI_SYSERR]`
: UNIX system error.

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

4.2.1.2 SL_EMERGENCY_REQ

Description

The emergency request service primitive provides the SLS user with the ability to specify that emergency alignment procedures should be used on the current or next alignment of the signalling link. Emergency alignment procedures a shorter in duration (shorter proving period) than normal alignment procedures.

Format

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

```
typedef struct {
    sl_long sl_primitive;
} sl_emergency_req_t;
```

Parameters

The emergency request service primitive contains the following parameters:

sl_primitive
: Specifies the service primitive type. Always `SL_EMERGENCY_REQ`.

State

This primitive is only valid in the `LMI_ENABLED` management state. The primitive is valid in any link state.

New State

The management and link state remains unchanged.

Response

The emergency request service primitive does not require receipt acknowledgement.

- **Successful:** When successful, the emergency request service primitive does not require receipt acknowledgement.
- **Unsuccessful (non-fatal errors):** When unsuccessful, the SLS provider negatively acknowledges the primitive with an `LMI_ERROR_ACK` primitive containing the error.

Reason for Failure

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

`[LMI_UNSPEC]`
: Unknown or unspecified.

`[LMI_EVENT]`
: Protocol-specific event occurred.

`[LMI_FATALERR]`
: Device has become unusable.

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

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

`[LMI_SYSERR]`
: UNIX system error.

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

4.2.1.3 SL_EMERGENCY_CEASES_REQ

Description

The emergency ceases request service primitive provides the SLS user with the ability to specify that normal alignment procedures should be used on the current or next alignment of the signalling link. Normal alignment procedures are longer in duration (longer proving period) than emergency alignment procedures.

Format

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

```
typedef struct {
    sl_long sl_primitive;
} sl_emergency_ceases_req_t;
```

Parameters

The emergency ceases request service primitive contains the following parameters:

sl_primitive
: Specifies the service primitive type. Always `SL_EMERGENCY_CEASES_REQ`.

State

This primitive is only valid in the `LMI_ENABLED` management state. The primitive is valid in any link state.

New State

The management and link state remains unchanged.

Response

The emergency ceases request service primitive does not require receipt acknowledgement.

- **Successful:** When successful, the emergency ceases request service primitive does not require receipt acknowledgement.
- **Unsuccessful (non-fatal errors):** When unsuccessful, the SLS provider negatively acknowledges the primitive with an `LMI_ERROR_ACK` primitive containing the error.

Reason for Failure

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

`[LMI_UNSPEC]`
: Unknown or unspecified.

`[LMI_EVENT]`
: Protocol-specific event occurred.

`[LMI_FATALERR]`
: Device has become unusable.

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

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

`[LMI_SYSERR]`
: UNIX system error.

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

4.2.1.4 SL_START_REQ

Description

The start request service primitive allows the SLS user to request that a signalling link be aligned and brought into service by the SLS provider.

Format

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

```
typedef struct {
    sl_long sl_primitive;
} sl_start_req_t;
```

Parameters

The start request service primitive contains the following parameters:

sl_primitive
: Specifies the service primitive type. Always SL_START_REQ.

State

This primitive is only valid in management state LMI_ENABLED. This primitive is valid in link state SL_STATE_OUT_OF_SERVICE.

New State

The new link state is SL_STATE_INITIAL_ALIGNMENT.

Response

The start request service primitive requires a response from the SLS provider indicating the success or failure of the start request.

- **Successful link start:** When successful, the SLS provider indicates success with the SL_IN_SERVICE_IND primitive indicating that the signalling link has been brought into service. A significant delay in time might exist between the request and the in-service indication. This results in the SL_STATE_IN_SERVICE link state.
- **Unuccessful link start:** When unsuccessful, the SLS provider indicates failure to bring the link in-service with the SL_OUT_OF_SERVICE_IND primitive, containing the reason for failure. This results int the SL_STATE_OUT_OF_SERVICE link state.
- **Non-fatal errors:** Non-fatal errors are indicated by the SLS provider using the LMI_ERROR_ACK primitive with the error number and reason contained.

When the management state is LMI_ENABLED, but the link state is other than SL_STATE_OUT_OF_SERVICE and SL_STATEPOWER_OFF, the SLS provider should ignore the SL_START_REQ primitive and not generate a non-fatal error.

Reason for Failure

Applicable reasons for unsuccessful link start are as follows:

[SL_FAIL_UNSPECIFIED]
: The signalling link failed for an unspecified reason.

`[SL_FAIL_CONG_TIMEOUT]`
: The signalling link failed because of congestion timeout (T6 expiry).

`[SL_FAIL_ACK_TIMEOUT]`
: The signalling link failed because of acknowledgement timeout (T7 expiry).

`[SL_FAIL_ABNORMAL_BSNR]`
: The signalling link failed because of receipt of an abnormal backward sequence number (BSNR).

`[SL_FAIL_ABNORMAL_FIBR]`
: The signalling link failed because of receipt of an abnormal forward indicator bit (FIBR).

`[SL_FAIL_SUERM_EIM]`
: The signalling link failed because the SUERM or EIM error rate threshold was exceeded.

`[SL_FAIL_ALIGNMENT_NOT_POSSIBLE]`
: The signalling link failed because the AERM threshold was exceeded and the maximum number of proving periods was exceeded.

`[SL_FAIL_RECEIVED_SIO]`
: The signalling link failed due to receipt of an SIO during or after alignment.

`[SL_FAIL_RECEIVED_SIN]`
: The signalling link failed due to receipt of an SIN after proving.

`[SL_FAIL_RECEIVED_SIE]`
: The signalling link failed due to receipt of an SIE after proving.

`[SL_FAIL_RECEIVED_SIOS]`
: The signalling link failed due to receipt of an SIOS.

`[SL_FAIL_T1_TIMEOUT]`
: The signalling link failed due to failure to align with remote (T1 timeout).

Applicable non-fatal errors are as follows:

`[LMI_UNSPEC]`
: Unknown or unspecified.

`[LMI_DISC]`
: Disconnected.

`[LMI_EVENT]`
: Protocol-specific event occurred.

`[LMI_FATALERR]`
: Device has become unusable.

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

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

`[LMI_SYSERR]`
: UNIX system error.

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

4.2.1.5 SL_IN_SERVICE_IND

Description

The in-service indication service primitive is issued by the SLS provider to indicate to the SLS user that a previously invoked link start has successfully aligned and brought the signalling link into service.

Format

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

```
typedef struct {
    sl_long sl_primitive;
} sl_in_service_ind_t;
```

Parameters

The in-service indication service primitive contains the following parameters:

sl_primitive
: Indicates the service primitive type. Always `SL_IN_SERVICE_IND`.

State

This primitive is only issued in the `LMI_ENABLED` management state. This primitive is only issued in the `SL_STATE_ALIGNED_READY` state.

New State

The new link state is `SL_STATE_IN_SERVICE`.

Rules

The following rules are observed by the SLS provider when issuing the in-service indication primitive:

- The primitive is only issued in response to a `SL_START_REQ` primitive that was issued from the `SL_STATE_OUT_OF_SERVICE` state.
- The primitive is only issued once the signalling link has achieved the `SL_STATE_IN_SERVICE` state.

4.2.1.6 SL_OUT_OF_SERVICE_IND

Description

The out-of-service indication service primitive is issued by the SLS provider to indicate to the SLS user that a previously invoked link start has been unsuccessful, or that a previously in-service signalling link has failed.

Format

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

```
typedef struct {
    sl_long sl_primitive;
    sl_ulong sl_timestamp;
    sl_ulong sl_reason;
} sl_out_of_service_ind_t;
```

Parameters

The out-of-service indication service primitive contains the following parameters:

sl_primitive
: Indicates the service primitive type. Always `SL_OUT_OF_SERVICE_IND`.

sl_timestamp
: Indication the time of the failure. The time is indicated as UNIX time from epoch in milliseconds.

sl_reason
: Indicates the reason for failure to start the link or the reason for failure of an in-service link. The *sl_reason* field can assume one of the following values:

  `SL_FAIL_UNSPECIFIED`
  : The signalling link failed for an unspecified reason.

  `SL_FAIL_CONG_TIMEOUT`
  : The signalling link failed because of congestion timeout (T6 expiry).

  `SL_FAIL_ACK_TIMEOUT`
  : The signalling link failed because of acknowledgement timeout (T7 expiry).

  `SL_FAIL_ABNORMAL_BSNR`
  : The signalling link failed because of receipt of an abnormal backward sequence number (BSNR).

  `SL_FAIL_ABNORMAL_FIBR`
  : The signalling link failed because of receipt of an abnormal forward indicator bit (FIBR).

  `SL_FAIL_SUERM_EIM`
  : The signalling link failed because the SUERM or EIM error rate threshold was exceeded.

  `SL_FAIL_ALIGNMENT_NOT_POSSIBLE`
  : The signalling link failed because the AERM threshold was exceeded and the maximum number of proving periods was exceeded.

`SL_FAIL_RECEIVED_SIO`
: The signalling link failed due to receipt of an SIO during or after alignment.

`SL_FAIL_RECEIVED_SIN`
: The signalling link failed due to receipt of an SIN after proving.

`SL_FAIL_RECEIVED_SIE`
: The signalling link failed due to receipt of an SIE after proving.

`SL_FAIL_RECEIVED_SIOS`
: The signalling link failed due to receipt of an SIOS.

`SL_FAIL_T1_TIMEOUT`
: The signalling link failed due to failure to align with remote (T1 timeout).

State

This primitive is only issued in the `LMI_ENABLED` management state. This primitive is only issued from a link state other than `SL_STATE_OUT_OF_SERVICE` or `SL_STATE_POWER_OFF`.

New State

The new link state is `SL_STATE_OUT_OF_SERVICE`.

Rules

The following rules are observed by the SLS provider when issuing the out-of-service indication primitive:

— The primitive is only issued in response to a `SL_START_REQ` primitive that was issued from the `SL_STATE_OUT_OF_SERVICE` state, or as a result of a link failure from the `SL_STATE_IN_SERVICE` state.

— The primitive is only issued once the signalling link has achieved the `SL_STATE_OUT_OF_SERVICE` state.

4.2.1.7 SL_STOP_REQ

Description

The stop request primitive allows the SLS user to request that a signalling link be brought out of service by the SLS provider.

Format

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

```
typedef struct {
    sl_long sl_primitive;
} sl_stop_req_t;
```

Parameters

The stop request service primitive contains the following parameters:

sl_primitive
: Specifies the service primitive type. Always SL_STOP_REQ.

State

This primitive is only valid in management state LMI_ENABLED. This primitive is valid in link state SL_STATE_INITIAL_ALIGNMENT, SL_STATE_ALIGNED_READY, SL_STATE_ALIGNED_NOT_READY or SL_STATE_IN_SERVICE.

New State

The new link state is SL_STATE_OUT_OF_SERVICE.

Response

The stop request service primitive does not require receipt acknowledgement from the SLS provider.

- **Successful:** When successful, the SLS provider does not need to acknowledge the stop request service primitive. The resulting link state is SL_STATE_OUT_OF_SERVICE.
- **Unsuccessful (non-fatal errors):** When unsuccessful, the SLS provider negatively acknowledges the stop request service primitive with a LMI_ERROR_ACK primitive containing the error and reason. The resulting state is unchanged.

When the management state is LMI_ENABLED, but the link state is SL_STATE_POWER_OFF or SL_STATE_OUT_OF_SERVICE, the SLS provider should ignore the SL_STOP_REQ primitive and not generate a non-fatal error.

Reason for Failure

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

[LMI_UNSPEC]
: Unknown or unspecified.

[LMI_EVENT]
: Protocol-specific event occurred.

[LMI_FATALERR]
: Device has become unusable.

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

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

`[LMI_SYSERR]`
: UNIX system error.

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

4.2.2 Data Transfer Service Primitives

Data transfer service primitives provider the SLS user with the ability to send and receive message signal units on an in-service signalling link. These service primitives implement the data transfer service (see Section 3.2.2 [Data Transfer Service], page 22).

4.2.2.1 SL_PDU_REQ

Description

The PDU request service primitive provides the SLS user with the ability to request that a message signal unit be transmitted on an in-service signalling link.

Format

The PDU request service primitive consists of zero or one `M_PROTO` message block and one `M_DATA` message block containing the message signal unit. The structure of the `M_PROTO` message block is as follows:

```
typedef struct {
    sl_long sl_primitive;
    sl_ulong sl_mp;
} sl_pdu_req_t;
```

Parameters

The PDU request service primitive contains the following parameters:

sl_primitive
: Specifies the service primitive type. Always `SL_PDU_REQ`.

sl_mp
: Specifies the message priority for the message signal unit. Message priorities are provider-specific, but are typically between 0 and 3. This message priority field is only applicable to SS7 protocol variants that place message priority bits in a field of the Level 2 header (TTC).

State

This primitive is only valid in the `LMI_ENABLED` management state, and is valid from the `SL_STATE_IN_SERVICE` link state.

New State

The management and link state remains unchanged.

Rules

The following rules are observed when issuing the PDU request service primitive:

- The `M_PROTO` message block is optional and is only necessary for the TTC SS7 protocol variant, or an SS7 protocol variant which places message priority bits into the Level 2 header.
- The PDU request service primitive does not require a response from the SLS provider.

Response

The PDU request service primitive is not acknowledged.

4.2.2.2 SL_PDU_IND

Description

The PDU indication service primitive provides the SLS user with the ability to receive message signal units from a signalling link.

Format

The PDU indication service primitive consists of zero or more `M_PROTO` message blocks and one or more `M_DATA` message blocks containing the message signal unit. The structure of the `M_PROTO` message block is as follows:

```
typedef struct {
    sl_long sl_primitive;
    sl_ulong sl_mp;
} sl_pdu_ind_t;
```

Parameters

The PDU indication service primitive contains the following parameters:

sl_primitive
: Indicates the service primitive type. Always `SL_PDU_IND`.

sl_mp
: Indicates the message priority of the message signal unit. Message priorities are provider-specific, but are typically between 0 and 3. This message priority field is only applicable to SS7 protocol variants that place message priority bits in a field of the Level 2 header (e.g. TTC).

State

This primitive is only valid in the `LMI_ENABLED` management state, and is valid from the `SL_STATE_IN_SERVICE` link state.

New State

The management and link states remain unchanged.

Rules

The following rules are observed when issuing the PDU indication service primitive:

— The `M_PROTO` message block is optional and is only necessary for the TTC SS7 protocol variant, or an SS7 protocol variant that passes message priority bits from the Level 2 header.

— The PDU indication service primitive does not require a response from the SLS user.

4.2.3 Congestion Service Primitives

These service primitives implement the congestion services (see Section 3.2.3 [Congestion Services], page 23).

4.2.3.1 SL_LINK_CONGESTED_IND

Description

The link congested indication service primitive provides the SLS provider with the ability to inidicate link transmit congestion onset at a congestion level to the SLS user.

Format

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

```
typedef struct {
    sl_long sl_primitive;
    sl_ulong sl_timestamp;
    sl_ulong sl_cong_status;            /* congestion status */
    sl_ulong sl_disc_status;            /* discard status */
} sl_link_cong_ind_t;
```

Parameters

The link congested indication service primitive contains the following parameters:

sl_primitive
: Indicates the service primitive type. Always `SL_LINK_CONGESTED_IND`.

sl_timestamp
: Indicates the time at which the change in congestion status occurred. This is UNIX time from epoch timestamp in milliseconds.

sl_cong_status
: Indicates the congestion status. The congestion status is the maximum level at which transmit congestion onset has occurred. This field is provider-specific but can typically take on a value from 0 to 3. The SLS user should signal congestion to the senders of messages with message priority less than the congestion status but should not discard messages of that priority.

sl_disc_status
: Indicates the discard status. The discard status is the maximum level at which transmit congestion discard has occurred. This field is provider-specific but can typically take on a value from 0 to 3. The SLS user should signal congestion to senders of message with message priority less than the discard status and should also discard messages of that priority.

State

This primitive is only issued in the `LMI_ENABLED` management state and the `SL_STATE_IN_SERVICE` link state.

New State

The management and link state remain unchanged.

Rules

The SLS provider observes the following rules when issuing the link congested indication service primitive:

— The service primitive is only issued from the `SL_STATE_IN_SERVICE` link state.

— The service primitive is only issued from the `LMI_ENABLED` management state.

— The service primitive is only issued when the congestion status or discard status increases from the value that was last indicated with either a `SL_LINK_CONGESTION_IND` or `SL_LINK_CONGESTION_CEASED_IND` primitive.

Response

The SLS user upon receiving this primitive should avoid sending messages of message priority less than the transmit congestion status, and must not send messages of message priority less than the discard status. The SLS provider does not actually discard messages with message priority less than the discard status: it is the responsibility of the SLS user to discard lower priority messages.

Typically the SLS user is the SS7 Message Transfer Part. The SS7 MTP issues congestion indications to local MTP-Users and issues transfer-controlled messages to sending signalling points when transmit congestion onset occurs. When transmit congestion discard occurs, the SS7 MTP continues to issue congestion indications to local MTP-User and transfer-controlled message to sending signalling points, but also discards messages with insufficient priority for the discard level.

4.2.3.2 SL_LINK_CONGESTION_CEASED_IND

Description

The link congestion ceased indication service primitive allows the SLS provider to indicate to the SLS user when transmit congestion abates.

Format

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

```
typedef struct {
    sl_long sl_primitive;
    sl_ulong sl_timestamp;
    sl_ulong sl_cong_status;            /* congestion status */
    sl_ulong sl_disc_status;            /* discard status */
} sl_link_cong_ceased_ind_t;
```

Parameters

The link congestion ceased service primitive contains the following parameters:

sl_primitive
: Indicates the service primitive type. Always `SL_CONGESTION_CEASED_IND`.

sl_timestamp
: Indicates the time at which the change in transmit congestion status occurred. This is UNIX time from epoch timestamp in milliseconds.

sl_cong_status
: Indicates the congestion status. The congestion status is the maximum level at which transmit congestion onset has occurred. This field is provider-specific but can typically take on a value from 0 to 3. The SLS user should signal congestion to the senders of messages with message priority less than the congestion status but should not discard messages of that priority.

sl_disc_status
: Indicates the discard status. The discard status is the maximum level at which transmit congestion discard has occurred. This field is provider-specific but can typically take on a value from 0 to 3. The SLS user should signal congestion to senders of message with message priority less than the discard status and should also discard messages of that priority.

State

This primitive is only issued in the `LMI_ENABLED` management state and the `SL_STATE_IN_SERVICE` link state.

New State

The management and link state remain unchanged.

Rules

The SLS provider observes the following rules when issuing the link congestion ceased indication service primitive:

— The service primitive is only issued from the SL_STATE_IN_SERVICE link state.
— The service primitive is only issued from the LMI_ENABLED management state.
— The service primitive is only issued when the congestion status or discard status decreases from the value that was last indicated with either a SL_LINK_CONGESTION_IND or SL_LINK_CONGESTION_CEASED_IND primitive.

Response

The SLS user upon receiving this primitive should cease discarding or sending congestion indications or transfer-controlled messages for the congestion level which has abated.

4.2.3.3 SL_CONGESTION_DISCARD_REQ

Description

The congestion discard request service primitive is used by the SLS user to specify receive congestion discard.

Normally an SLS user will first signal receive congestion onset with the `SL_CONGESTION_ACCEPT_REQ` primitive before signalling receive congestion discard with this `SL_CONGESTION_DISCARD_REQ` primitive. The congestion discard service primitive requests that the SLS provider discard all new undelivered message signal units and not acknowledge them to the remote SLS provider. The SLS provider will also generate receive congestion indications to the remote SLS provider (i.e. will periodically generate SIB).

Format

The congestion discard request service primitive consists of one `M_PCPROTO` message block, structured as follows:

```
typedef struct {
    sl_long sl_primitive;
} sl_cong_discard_req_t;
```

Parameters

The congestion discard request service primitive contains the following parameters:

sl_primitive
: Specifies the service primitive type. Always `SL_CONGESTION_DISCARD_REQ`.

State

This primitive is valid only in `LMI_ENABLED` management state. It is valid in `SL_STATE_IN_SERVICE` link state.

New State

The link and management state remains the same.

Rules

The SLS user should observe the following rules when issuing the congestion discard request service primitive:

— The SLS user should not generate a congestion discard request unless a congestion accept request was previously issued.

— The SLS user should not generate a congestion discard request unless a congestion accept request was previously issued *and* a message signal unit has been delivered since the congestion accept request was issued.

Response

The congestion discard request service primitive does not require receipt acknowledgement.

– **Successful:** When successful, this primitive does not require acknowledgement. The state remains the same.

– **Unsuccessful (non-fatal errors):** When unsuccessful, the SLS provider negatively acknowledges the primitive using the `LMI_ERROR_ACK` primitive containing the error and reason. The state remains the same.

Note that if the SLS provider is in the `LMI_ENABLED` state, but the link is not in the `SL_STATE_IN_SERVICE` state, the primitive should be ignored and no non-fatal error generated.

Reason for Failure

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

`[LMI_UNSPEC]`
: Unknown or unspecified.

`[LMI_EVENT]`
: Protocol-specific event occurred.

`[LMI_FATALERR]`
: Device has become unusable.

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

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

`[LMI_SYSERR]`
: UNIX system error.

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

4.2.3.4 SL_CONGESTION_ACCEPT_REQ

Description

The congestion accept request service primitive is used by the SLS user to specify receive congestion onset.

Format

The congestion accept request service primitive consists of one M_PCPROTO message block, structured as follows:

```
typedef struct {
    sl_long sl_primitive;
} sl_cong_accept_req_t;
```

Parameters

The congestion accept request service primitive contains the following parameters:

sl_primitive
: Specifies the service primitive type. Always SL_CONGESTION_ACCEPT_REQ.

State

This primitive is valid only in LMI_ENABLED management state. It is valid in SL_STATE_IN_SERVICE link state.

New State

The link and management state remains the same.

Response

The congestion accept request service primitive does not require receipt acknowledgement.

- **Successful:** When successful, this primitive does not require acknowledgement. The state remains the same.
- **Unsuccessful (non-fatal errors):** When unsuccessful, the SLS provider negatively acknowledges the primitive using the LMI_ERROR_ACK primitive containing the error and reason. The state remains the same.

Note that if the SLS provider is in the LMI_ENABLED state, but the link is not in the SL_STATE_IN_SERVICE state, the primitive should be ignored and no non-fatal error generated.

Reason for Failure

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

[LMI_UNSPEC]
: Unknown or unspecified.

[LMI_EVENT]
: Protocol-specific event occurred.

[LMI_FATALERR]
: Device has become unusable.

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

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

`[LMI_SYSERR]`
: UNIX system error.

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

4.2.3.5 SL_NO_CONGESTION_REQ

Description

The no congestion request service primitive is used by the SLS user to specify receive congestion abatement.

Format

The no congestion request service primitive consists of one M_PROTO or M_PCPROTO message block, structured as follows:

```
typedef struct {
    sl_long sl_primitive;
} sl_no_cong_req_t;
```

Parameters

The no congestion request service primitive contains the following parameters:

sl_primitive
: Specifies the service primitive type. Always SL_NO_CONGESTION_REQ.

State

This primitive is valid only in LMI_ENABLED management state. It is valid in SL_STATE_IN_SERVICE link state.

New State

The link and management state remains the same.

Response

The no congestion request service primitive does not require receipt acknowledgement.

- **Successful:** When successful, this primitive does not require acknowledgement. The state remains the same.
- **Unsuccessful (non-fatal errors):** When unsuccessful, the SLS provider negatively acknowledges the primitive using the LMI_ERROR_ACK primitive containing the error and reason. The state remains the same.

Note that if the SLS provider is in the LMI_ENABLED state, but the link is not in the SL_STATE_IN_SERVICE state, the primitive should be ignored and no non-fatal error generated.

Reason for Failure

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

[LMI_UNSPEC]
: Unknown or unspecified.

[LMI_EVENT]
: Protocol-specific event occurred.

[LMI_FATALERR]
: Device has become unusable.

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

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

`[LMI_SYSERR]`
: UNIX system error.

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

4.2.4 Restoration Service Primitives

The restoration service primitives permit the SLS user to perform functions necessary for BSNT retrieval to initiate or respond to sequenced changeover, buffer updating to respond to sequenced or time-controlled changeover, and buffer clearing to respond to time-controlled changeover or processor outage related failures.

These service primitives implement the restoration services (see Section 3.2.4 [Restoration Services], page 24).

4.2.4.1 SL_RETRIEVE_BSNT_REQ

Description

The retrieve BSNT request service primitive allows the SLS user to request retrieval of the BSNT (backward sequence number transmitted) which indicates the sequence number of the remove message signal unit sent that was last acknowledged. This function is necessary to properly generate or respond to a sequenced changeover procedure by the SLS user.

Format

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

```
typedef struct {
    sl_long sl_primitive;
} sl_retrieve_bsnt_req_t;
```

Parameters

The retrieve BSNT request service primitive contains the following parameters:

sl_primitive
: Specifies the service primitive type. Always `SL_RETRIEVE_BSNT_REQ`.

State

This primitive is valid only in the `LMI_ENABLED` management state. The primitive is valid in the `SL_STATE_OUT_OF_SERVICE` state.

New State

The new state is unchanged.

Rules

The SLS user should observe the following rules when issuing the retrieve BSNT request service primitive:

— The SLS user should ensure that the link is in the `SL_STATE_OUT_OF_SERVICE` state before issuing this primitive. One easy way to ensure that the link is in this state is to issue the stop request `SL_STOP_REQ`.

Response

This service primitive requires the SLS provider to acknowledge success of failure of the retrieval operation.

- **Successful retrieval:** When successful, the SLS provider indicate the retrieved BSNT value using the `SL_BSNT_IND` primitive containing the BSNT value. The management and link states remain the same.
- **Unsuccessful retrieval:** When unsuccessful, the SLS provider indicates that the BSNT value cannot be retrieved using the `SL_BSNT_NOT_RETRIEVABLE_IND`. The management and link states remain the same.
- **Non-fatal errors:** When a non-fatal error occurs, the SLS provider indicates the error using the `LMI_ERROR_ACK` primitive containing the error and the reason.

When the management state is `LMI_ENABLED` and the link state is other than `SL_STATE_OUT_OF_SERVICE`, the SLS provider should respond with `SL_BSNT_NOT_RETRIEVABLE_IND` instead of generating a non-fatal error.

Reason for Failure

Most SLS providers are always successful in retrieving the BSNT value. Applicable reasons for failing to retrieve the BSNT value are as follows:

1. Hardware failure.
2. The signalling link is in the incorrect state (e.g. the in-service state).

Applicable non-fatal errors are as follows:

`[LMI_UNSPEC]`
: Unknown or unspecified.

`[LMI_DISC]`
: Disconnected.

`[LMI_EVENT]`
: Protocol-specific event occurred.

`[LMI_FATALERR]`
: Device has become unusable.

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

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

`[LMI_SYSERR]`
: UNIX system error.

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

4.2.4.2 SL_BSNT_IND

Description

The BSNT indication service primitive is originated by the SLS provider to indicate the retrieved BSNT value in response to a SL_RETRIEVE_BSNT_REQ primitive from the SLS user.

Format

The BSNT indication service primitive consists of one M_PROTO or M_PCPROTO message block, structured as follows:

```
typedef struct {
    sl_long sl_primitive;
    sl_ulong sl_bsnt;
} sl_bsnt_ind_t;
```

Parameters

The BSNT indication service primitive contains the following parameters:

sl_primitive
: Indicates the service primitive type. Always SL_BSNT_IND.

sl_bsnt
: Indicates the value of the BSNT. The format of the BSNT value is provider-specific but is typically a 7-bit or 12-bit sequence number.

State

This primitive is valid in management state LMI_ENABLED and link state SL_STATE_OUT_OF_SERVICE.

New State

The new state remains unchanged.

Rules

The SLS provider observes the following rules when issuing a BSNT indication service primitive:

— The primitive is only issued from the LMI_ENABLED management state and the SL_STATE_OUT_OF_SERVICE link state.

— The primitive is only issued in response to an outstanding SL_RETRIEVE_BSNT_REQ primitive when it is possible for the SLS provider to retrieve the BSNT value.

Response

The primitive does not require a response from the SLS user.

4.2.4.3 SL_BSNT_NOT_RETRIEVABLE_IND

Description

The BSNT not retrievable indication service primitive is originated by the SLS provider to indicate that the BSNT value cannot be retrieved in response to a `SL_RETRIEVE_BSNT_REQ` primitive from the SLS user.

Format

The BSNT not retrievable indication service primitive consists of one `M_PROTO` or `M_PCPROTO` message block, structured as follows:

```
typedef struct {
    sl_long sl_primitive;
    sl_ulong sl_bsnt;
} sl_bsnt_not_retr_ind_t;
```

Parameters

The BSNT not retrievable indication service primitive contains the following parameters:

sl_primitive
: Indicates the service primitive type. Always `SL_BSNT_NOT_RETRIEVABLE_IND`.

sl_bsnt
: Indicates the value of the BSNT. This value is the known value of the last acknowledged message signal unit from the remote peer or minus one (-1UL) indicating that a reasonable BSNT value is not known. The format of the BSNT is provider-specific, but is typically a 7-bit or 12-bit sequence number.

State

This primitive is valid in management state `LMI_ENABLED` and is valid in any link state.

New State

The new state remains unchanged.

Rules

The SLS provider observes the following rules when issuing the BSNT not retrievable indication service primitive:

- The primitive is only issued from the `LMI_ENABLED` management state, but may be issued from any link state.
- The primitive is only issued in response to an outstanding `SL_RETRIEVE_BSNT_REQ` primitive when it is not possible for the SLS provider to retrieve the BSNT value.
- When issued, a non-fatal error for the same request will not be issued.

Response

The primitive does not require a response from the SLS user.

4.2.4.4 SL_RETRIEVAL_REQUEST_AND_FSNC_REQ

Description

The retrieval request and FSNC request service primitive is originated by the SLS user when it wishes to update the retransmission buffer with the last known acknowledged message (FSNC). The last known acknowledged message is acquired by the SLS user with the sequence changeover procedure of the message transfer part. The primitive requests that the SLS provider update the retransmission buffer and then deliver the contents of the updated retransmission buffer and transmit buffers to the SLS user.

Format

The retrieval request and FSNC request service primitive consists of one `M_PROTO` or `M_PCPROTO` message block, structured as follows:

```
typedef struct {
    sl_long sl_primitive;
    sl_ulong sl_fsnc;
} sl_retrieval_req_and_fsnc_t;
```

Parameters

The retrieval request and FSNC request service primitive contains the following parameters:

sl_primitive
: Specifies the service primitive type. Always `SL_RETRIEVAL_REQ_AND_FSNC_REQ`.

sl_fsnc
: Specifies the value of the FSNC (forward sequence number confirmed). This is the last known message to be acknowledge by the remote SLS provider. The format of the FSNC is provider-specific, but is typically a 7-bit or 12-bit sequence number.

State

This primitive is only valid in management state `LMI_ENABLED` and is valid in link state `SL_STATE_OUT_OF_SERVICE`.

New State

The new state remains unchanged.

Rules

Response

The retrieval request and FSNC request service primitive request the SLS provider to acknowledge the result of the retrieval action as follows:

- **Successful retrieval:** When successful, the SLS provider indicates the updated contents of the retransmission buffer and the contents of the transmission buffer using the `SL_RETRIEVED_MESSAGE_IND` primitive followed by a `SL_RETRIEVAL_COMPLETE_IND` primitive. The state remains unchanged.
- **Unsuccessful retrieval:** When unsuccessful, the SLS provider indicates failure to retrieve the contents of the buffers with the `SL_RETRIEVAL_NOT_POSSIBLE_IND` primitive.
- **Non-fatal errors:** When a non-fatal error occurs, the SLS provider indicates the error using the `LMI_ERROR_ACK` primitive containing the error and the reason. The state remains unchanged.

When the management stat is LMI_ENABLED and the link state is other than SL_STATE_OUT_OF_SERVICE, the SLS provider should respond with SL_RETRIEVAL_NOT_POSSIBLE_IND instead of generating a non-fatal error.

Reason for Failure

Most SLS providers are always successful in retrieving the updated contents of the retransmission buffer and transmission buffer. Applicable reasons for failing to retrieve the updated buffer contents are as follows:

1. Hardware failure.
2. The signalling link is in the incorrect link state (e.g. the in-service state).
3. The specified value of FSNC does not match and is not adjacent to a message contained in the retransmission buffer.

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

[LMI_UNSPEC]
: Unknown or unspecified.

[LMI_DISC]
: Disconnected.

[LMI_EVENT]
: Protocol-specific event occurred.

[LMI_FATALERR]
: Device has become unusable.

[LMI_OUTSTATE]
: Primitive was issued from invalid state.

[LMI_PROTOSHORT]
: M_PROTO block too short.

[LMI_SYSERR]
: UNIX system error.

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

4.2.4.5 SL_RETRIEVED_MESSAGE_IND

Description

The retrieved message indication service primitive is originated by the SLS provider to transfer the contents of the updated retransmission buffer and transmission buffer to the SLS user. One primitive is used for each message retrieved. The oldest message in the buffers is indicated first.

Format

The retrieved message indication service primitive consists of one `M_PROTO` message block followed by one or more `M_DATA` message blocks containing the retrieved message signal unit in the same format as it was presented to the SLS provider for transmission. The `M_PROTO` message block is structured as follows:

```
typedef struct {
    sl_long sl_primitive;
    sl_ulong sl_mp;
} sl_retrieved_msg_ind_t;
```

Parameters

The retrieve message indication service primitive contains the following parameters:

sl_primitive
: Indicates the service primitive type. Always `SL_RETRIEVED_MESSAGE_IND`.

sl_mp
: Indicates the message priority for the message that was specified in the `SL_PDU_REQ` primitive from the SLS user when the message was submitted for transmission. Message priorities are provider-specific, but are typically between 0 and 3. This message priority field is only applicable to SS7 protocol variants that place message priority bits in a field of the Level 2 header, such as TTC.

State

This primitive is only issued in management state `LMI_ENABLED` and link state `SL_STATE_OUT_OF_SERVICE`.

New State

The new state remains unchanged.

Rules

The SLS provider observes the following rules when issuing a retrieved message indication service primitive:

— The primitive is only issued from the `LMI_ENABLED` management state and the `SL_STATE_OUT_OF_SERVICE` link state.

— The primitive is only issued in response to an outstanding `SL_RETRIEVAL_REQUEST_AND_FSNC_REQ` primitive when it is possible for the SLS provider to update and retrieve message signal units from the retransmission and transmission buffers.

— The primitive is not issued when the updated retransmission buffer and transmission buffer are empty.

Response

This primitive does not require response from the SLS user.

4.2.4.6 SL_RETRIEVAL_COMPLETE_IND

Description

The retrieval complete indication service primitive is originated by the SLS provider to indicate the completion of transfer of the contents of the updated retransmission buffer and transmission buffer to the SLS user. The primitive is issued in response to a `SL_RETRIEVAL_REQUEST_AND_FSNC_REQ` primitive issued by the SLS user.

Format

The retrieval complete indication service primitive consists of one `M_PROTO` message block and zero or more `M_DATA` message blocks containing the last retrieved message signal unit in the same format as it was presented to the SLS provider for transmission. The `M_PROTO` message block is structured as follows:

```
typedef struct {
    sl_long sl_primitive;
    sl_ulong sl_mp;
} sl_retrieval_comp_ind_t;
```

Parameters

The retrieval complete indication service primitive contains the following parameters:

sl_primitive
: Indicates the service primitive type. Always `SL_RETRIEVAL_COMPLETE_IND`.

sl_mp
: When accompanied by `M_DATA` message blocks containing the last retrieved message signal unit, the *sl_mp* field indicates the message priority for that message that was specified in the `SL_PDU_REQ` primitive from the SLSL user when the message was submitted for transmission. Message priorities are provider-specific, but are typically between 0 and 3. This message priority field is only applicable to SS7 protocol variants that place message priority bits in a field of the Level 2 header, such as TTC.

State

This primitive is only issued in management state `LMI_ENABLED` and link state `SL_STATE_OUT_OF_SERVICE`.

New State

The new state remains unchanged.

Rules

The SLS provider observes the following rules when issuing a retrieval complete indication service primitive:

- The primitive is only issued from the `LMI_ENABLED` management state and the `SL_STATE_OUT_OF_SERVICE` link state.
- The primitive is only issued in response to an outstanding `SL_RETRIEVAL_REQUEST_AND_FSNC_REQ` primitive when transfer of the updated retransmission buffer and transmission buffer is complete.
- A message signal unit is not attached to the primitive in `M_DATA` message blocks when the updated retransmission and transmission buffers were empty.

— Attaching the last retrieved message to the primitive in `M_DATA` message blocks is optional and not recommended: the `SL_RETRIEVED_MESSAGE_IND` primitive should be used to transfer all retrieved message signal units first.

— Upon receipt of the retrieval complete indication service primitive, the SLS user will consider the retrieval operation complete.

Response

This primitive does not require a response from the SLS user.

Reason for Failure

4.2.4.7 SL_RETRIEVAL_NOT_POSSIBLE_IND

Description

The retrieval not possible indication service primitive is originated by the SLS provider to indicate that the updated contents of the retransmission and transmission buffers is not possible. The primitive is issued in response to a `SL_RETREIVAL_REQUEST_AND_FSNC_REQ` primitive received from the SLS user.

Format

The retrieval not possible indication service primitive consists of one `M_PROTO` message block, structured as follows:

```
typedef struct {
    sl_long sl_primitive;
} sl_retrieval_not_poss_ind_t;
```

Parameters

The retrieval not possible indication service primitive contains the following parameters:

sl_primitive
: Indicates the service primitive type. Always `SL_RETRIEVAL_NOT_POSSIBLE_IND`.

State

This primitive is only issued from the `LMI_ENABLED` management state, but may be issued from any link state.

New State

The new state remains unchanged.

Rules

The SLS provider observes the following rules when issuing the retrieval not possible indication service primitive:

- The primitive is only issued from the `LMI_ENABLED` management state, but may be issued from any link state.
- The primitive is only issued in response to an outstanding `SL_RETRIEVAL_REQUEST_AND_FSNC_REQ` primitive when it is not possible to update and retrieve the updated contents of the retransmission and transmission buffers.
- When issued, a non-fatal error will not be issued for the same request.
- Upon receipt of the primitive, the SLS user shall consider the retrieval operation complete.

Response

The primitive does not require a response from the SLS user.

4.2.4.8 SL_CLEAR_BUFFERS_REQ

Description

The clear buffers request service primitive is originated by the SLS user to request that all message buffers be cleared by the SLS provider. This includes receive buffer, retransmission buffer and transmission buffers.

Format

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

```
typedef struct {
    sl_long sl_primitive;
} sl_clear_buffers_req_t;
```

Parameters

The clear buffers request service primitive contains the following parameters:

sl_primitive
: Specifies the service primitive type. Always SL_CLEAR_BUFFERS_REQ.

State

This primitive is only valid in the LMI_ENABLED management state and the SL_STATE_OUT_OF_SERVICE link state.

New State

The new state remains unchanged.

Response

The clear buffers request service primitive requires the SLS provider to indicate when the receive buffer and retransmission buffers are cleared, as follows:

- **Successful:** When successful, the SLS provider clears the receive buffer, retransmission buffer and transmission buffer. When the receive buffer is cleared, the SLS provider indicates the clearing with the SL_RB_CLEARED_IND primitive. When the retransmission buffer is cleared, the SLS provider indicates the clearing with the SL_RTB_CLEARED_IND primitive. The state remains unchanged.
- **Unsuccessful (non-fatal errors):** When unsuccessful, the SLS provider negatively acknowledges the primitive using the LMI_ERROR_ACK primitive containing the error and reason for failure. The state remains unchanged.

Reason for Failure

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

[LMI_UNSPEC]
: Unknown or unspecified.

[LMI_DISC]
: Disconnected.

`[LMI_EVENT]`
: Protocol-specific event occurred.

`[LMI_FATALERR]`
: Device has become unusable.

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

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

`[LMI_SYSERR]`
: UNIX system error.

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

4.2.4.9 SL_CLEAR_RTB_REQ

Description

The clear RTB request service primitive is originated by the SLS user to request that only the retransmission buffer be cleared by the SLS provider. This primitive is used in conjunction with the time-controlled changeover procedure of the message transfer part.

Format

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

```
typedef struct {
    sl_long sl_primitive;
} sl_clear_rtb_req_t;
```

Parameters

The clear RTB request service primitive contains the following parameters:

sl_primitive
: Specifies the service primitive type. Always `SL_CLEAR_RTB_REQ`.

State

This primitive is only valid in the `LMI_ENABLED` management state and the `SL_STATE_OUT_OF_SERVICE` link state.

New State

The new state remains unchanged.

Response

The clear RTB request service primitive requires the SLS provider to indicate when the retransmission buffer has been cleared, as follows:

- **Successful:** When successful, the SLS provider clears the retransmission buffer. When the retransmission buffer is cleared, the SLS provider indicates the clearing with the `SL_RTB_CLEARED_IND` primitive. The state remains unchanged.
- **Unsuccessful (non-fatal errors):** When unsuccessful, the SLS provider negatively acknowledges the primitive using the `LMI_ERROR_ACK` primitive containing the error and reason for failure. The state remains unchanged.

Reason for Failure

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

`[LMI_UNSPEC]`
: Unknown or unspecified.

`[LMI_DISC]`
: Disconnected.

`[LMI_EVENT]`
: Protocol-specific event occurred.

`[LMI_FATALERR]`
: Device has become unusable.

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

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

`[LMI_SYSERR]`
: UNIX system error.

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

4.2.4.10 SL_RB_CLEARED_IND

Description

The RB cleared indication service primitive is originated by the SLS provider whenever the receive buffer has been cleared; either in response to a SL_CLEAR_BUFFERS_REQ primitive from the SLS user, or due to internal state machine operations.

Format

The RB cleared indication service primitive consists of one M_PROTO or M_PCPROTO message block, structured as follows:

```
typedef struct {
    sl_long sl_primitive;
} sl_rb_cleared_ind_t;
```

Parameters

The RB cleared indication service primitive contains the following parameters:

sl_primitive
: Indicates the service primitive type. Always SL_RB_CLEARED_IND.

State

This primitive is only issued by the SLS provider in the LMI_ENABLED management state and the SL_STATE_OUT_OF_SERVICE link state.

New State

The new state remains unchanged.

Rules

The SLS provider observes the following rules when issuing the RB cleared indication service primitive:

- The primitive is only issued from the LMI_ENABLED management state and the SL_STATE_OUT_OF_SERVICE link state.
- The primitive is issued in response to a SL_CLEAR_BUFFERS_REQ primitive from the SLS user.
- The primitive is also issued in response to internal state machine transitions.

Response

This primitive does not require a response from the SLS user.

4.2.4.11 SL_RTB_CLEARED_IND

Description

The RTB cleared indication service primitive is originated by the SLS provider whenever the retransmission buffer has been cleared; either in response to a SL_CLEAR_BUFFERS_REQ or SL_CLEAR_RTB_REQ primitive, or due to internal state machine operations.

Format

The RTB cleared indication service primitive consists of one M_PROTO message block, structured as follows:

```
typedef struct {
    sl_long sl_primitive;
} sl_rtb_cleared_ind_t;
```

Parameters

The RTB cleared indication service primitive contains the following parameters:

sl_primitive
: Indicates the service primitive type. Always SL_RTB_CLEARED_IND.

State

The primitive is only issued by the SLS provider from the LMI_ENABLED management state and the SL_STATE_OUT_OF_SERVICE link state.

New State

The new state remains unchanged.

Rules

The SLS provider observes the following rules when issuing the RTB cleared indication service primitive:

- The primitive is only issued from the LMI_ENABLED management state and the SL_STATE_OUT_OF_SERVICE link state.
- The primitive is issued in response to a SL_CLEAR_BUFFERS_REQ or SL_CLEAR_RTB_REQ primitive from the SLS user.
- The primitive is also issued in response to internal state machine transitions.

Response

This primitive does not require a response from the SLS user.

4.2.5 Processor Outage Service Primitives

The processor outage service primitive permit the SLS user the ability to assert and resume from a local processor outage condition as well as being informed by the SLS provider when a local or remote processor outage condition is in effect or has cleared. The SLS user is also able, using these and other primitives, to recover from a local or remote processor outage condition.

These service primitives implement the processor outage services (see Section 3.2.5 [Processor Outage Services], page 28).

4.2.5.1 SL_LOCAL_PROCESSOR_OUTAGE_REQ

Description

The local processor outage request service primitive allows the SLS user to specify that a local processor outage condition exists.

Format

The local processor outage request service primitive consists of one `M_PROTO` or `M_PCPROTO` message block, structured as follows:

```
typedef struct {
    sl_long sl_primitive;
} sl_local_proc_outage_req_t;
```

Parameters

The local processor outage request service primitive contains the following parameters:

sl_primitive
: Specifies the service primitive type. Always `SL_LOCAL_PROCESSOR_OUTAGE_REQ`.

State

This primitive is only valid in the `LMI_ENABLED` management state but is valid from any link state.

New State

The new state is `SL_STATE_PROCESSOR_OUTAGE`.

Response

This primitive does not request a response from the SLS provider.

- **Successful:** When successful, the link moves to the `SL_STATE_PROCESSOR_OUTAGE` state and a local processor outage condition is asserted.
- **Unsuccessful (non-fatal errors):** When unsuccessful, the SLS provider will negatively acknowledge the primitive using the `LMI_ERROR_ACK` primitive containing the error and reason for failure. The state remains unchanged.

Reason for Failure

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

`[LMI_UNSPEC]`
: Unknown or unspecified.

`[LMI_DISC]`
: Disconnected.

`[LMI_EVENT]`
: Protocol-specific event occurred.

`[LMI_FATALERR]`
: Device has become unusable.

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

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

`[LMI_SYSERR]`
: UNIX system error.

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

4.2.5.2 SL_LOCAL_PROCESSOR_OUTAGE_IND

Description

The local processor outage indication service primitive is originated by the SLS provider when it detects a local processor outage condition internal to the SLS provider.

Format

The local processor outage indication service primitive consists of on `M_PROTO` message block, structured as follows:

```
typedef struct {
    sl_long sl_primitive;
    sl_ulong sl_timestamp;
} sl_loc_proc_out_ind_t;
```

Parameters

The local processor outage indication service primitive contains the following parameters:

sl_primitive
: Indicates the service primitive type. Always `SL_LOCAL_PROCESSOR_OUTAGE_IND`.

sl_timestamp
: Indicates the time at which the detection of local processor outage occurred. This is UNIX time from epoch timestamp in milliseconds.

State

This primitive is only issued by the SLS provider in the `LMI_ENABLED` management state and active or blocked link state.

New State

The new state is `SL_STATE_PROCESSOR_OUTAGE`.

Rules

The SLS provider observes the following rules when issuing the local processor outage indication service primitive:

— The primitive is only issued in the `LMI_ENABLED` management state.

— SLS provider detection of local processor outage and SLS user detection of local processor outage are independent conditions.

— The SLS provider will issue a `SL_LOCAL_PROCESSOR_RECOVERED_IND` primitive when the local processor outage condition is no longer in effect.

Response

This primitive does not require a response from the SLS user.

4.2.5.3 SL_RESUME_REQ

Description

The resume request service primitive allows the SLS user to specify that a local processor outage condition is no longer in effect. That is, that the local processor has recovered.

Format

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

```
typedef struct {
    sl_long sl_primitive;
} sl_resume_req_t;
```

Parameters

The resume request service primitive contains the following parameters:

sl_primitive
: Specifies the service primitive type. Always SL_RESUME_REQ.

State

This primitive is only valid in the LMI_ENABLED management state and when the link is in the SL_STATE_PROCESSOR_OUTAGE state with a local processor outage condition asserted by the SLS user with a previous SL_LOCAL_PROCESSOR_OUTAGE_REQ primitive.

New State

The new state is SL_STATE_IN_SERVICE provided that no other processor outage condition is currently asserted.

Response

This primitive does not request a response from the SLS provider.

- **Successful:** When successful, the link moves to the SL_STATE_IN_SERVICE state and the local processor outage condition is removed.
- **Unsuccessful (non-fatal errors):** When unsuccessful, the SLS provider will negatively acknowledge the primitive using the LMI_ERROR_ACK primitive containing the error and reason for failure. The state remains unchanged.

Reason for Failure

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

[LMI_UNSPEC]
: Unknown or unspecified.

[LMI_DISC]
: Disconnected.

[LMI_EVENT]
: Protocol-specific event occurred.

`[LMI_FATALERR]`
: Device has become unusable.

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

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

`[LMI_SYSERR]`
: UNIX system error.

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

4.2.5.4 SL_LOCAL_PROCESSOR_RECOVERED_IND

Description

The local processor recovered indication service primitive is originated by the SLS provider when it detects a remote processor recovery condition.

Format

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

```
typedef struct {
    sl_long sl_primitive;
    sl_ulong sl_timestamp;
} sl_loc_proc_recovered_ind_t;
```

Parameters

The local processor recovered indication service primitive contains the following parameters:

sl_primitive
: Indicates the service primitive type. Always `SL_LOCAL_PROCESSOR_RECOVERED_IND`.

sl_timestamp
: Indicates the time at which the detection of local processor recovery occurred. This is UNIX time from epoch timestamp in milliseconds.

State

This primitive is only issued by the SLS provider in the `LMI_ENABLED` management state and the link state of `SL_STATE_PROCESSOR_OUTAGE` with local outage asserted by the SLS provider.

New State

The new state is `SL_STATE_IN_SERVICE` provided that no other processor outage condition (SLS user local, or remote) exists.

Rules

The SLS provider observes the following rules when issuing a local processor recovered indication service primitive:

— The primitive is only issued in the `LMI_ENABLED` management state.

— The SLS provider will only issue this primitive after it has issued a `SL_LOCAL_PROCESSOR_OUTAGE_IND` primitive and when the local processor outage condition is no longer in effect.

Response

This primitive does not require a respnose from the SLS user, nevertheless, the SLS user will typically attempt to continue on the link or restore it using restoration service primitives.

4.2.5.5 SL_REMOTE_PROCESSOR_OUTAGE_IND

Description

The remote processour outage indication service primitive is originated by the SLS provider when it detects a remote processor outage condition.

Format

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

```
typedef struct {
    sl_long sl_primitive;
    sl_ulong sl_timestamp;
} sl_rem_proc_out_ind_t;
```

Parameters

The remove processor outage indication service primitive contains the following parameters:

sl_primitive
: Indicates the service primitive type. Always `SL_REMOTE_PROCESSOR_OUTAGE_IND`.

sl_timestamp
: Indicates the time at which the detection of remote processor outage occurred. This is UNIX time from epoch timestamp in milliseconds.

State

This primitive is only issued by the SLS provider in the `LMI_ENABLED` management state and active or blocked link state.

New State

The new state is `SL_STATE_PROCESSOR_OUTAGE`.

Rules

The SLS provider observes the following rules when issuing the remote processor outage indication service primitive:

— The primitive is only issued in the `LMI_ENABLED` management state.

— The SLS provider will issue a `SL_REMOTE_PROCESSOR_RECOVERED_IND` primitive when the remote processor outage condition is no longer in effect.

Response

This primitive does not require a response from the SLS user.

4.2.5.6 SL_REMOTE_PROCESSOR_RECOVERED_IND

Description

The remote processor recovered indication service primitive is originated by the SLS provider when it detects a remote processor recovery condition.

Format

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

```
typedef struct {
    sl_long sl_primitive;
    sl_ulong sl_timestamp;
} sl_rem_proc_recovered_ind_t;
```

Parameters

The remote processor recovered indication service primitive contains the following parameters:

sl_primitive
: Indicates the service primitive type. Always `SL_REMOTE_PROCESSOR_RECOVERED_IND`.

sl_timestamp
: Indicates the time at which the detection of remote processor recovery occurred. This is UNIX time from epoch timestamp in milliseconds.

State

This primitive is only issued by the SLS provider in the `LMI_ENABLED` management state and the link state of `SL_STATE_PROCESSOR_OUTAGE` with remote process outage asserted.

New State

The new state is unchanged.

Rules

The SLS provider observes the following rules when issuing a remote processor recovered indication service primitive:

— The primitive is only issued in the `LMI_ENABLED` management state.

— The SLS provider will only issue this primitive after it was issued a `SL_REMOTE_PROCESSOR_OUTAGE_IND` primitive when the remote processor outage condition is no longer in effect.

Response

This primitive does not require a response from the SLS user, nevertheless, the SLS user will typically attempt to continue on the link or restore it using restoration service primitives.

4.2.5.7 SL_CONTINUE_REQ

Description

The continue request service primitive is originated by the SLS user to request that a link previously in a remote processor outage condition, or a SLS provider detected local process outage condition, be continued. This action is normally performed where processor outage has not been of a long duration and it is not necessary to fail or otherwise restore the signalling link.

Format

The continue request service primitive consists of one `M_PROTO` or `M_PCPROTO` message block, formatted as follows:

```
typedef struct {
    sl_long sl_primitive;
} sl_continue_req_t;
```

Parameters

The continue request service primitive contains the following parameters:

sl_primitive
: Specifies the service primitive type. Always `SL_CONTINUE_REQ`.

State

This primitive is only valid in the `LMI_ENABLED` management state and valid in the `SL_STATE_PROCESSOR_OUTAGE` state where local (SLS provider detected) or remote processor recovery has been indicated.

New State

The new state is `SL_STATE_IN_SERVICE`, provided that there is no other processor outage condition in effect.

Response

This primitive does not require receipt acknowledgement by the SLS provider.

- **Successful:** When successful, the primtiive does not require acknowledgement and the link moves to the `SL_STATE_IN_SERVICE` state.
- **Unsuccessful (non-fatal errors):** When unsuccessful, the SLS provider negatively acknowledges the primitive using an `LMI_ERROR_ACK` primitive containing the error and reason for failure. The state reamins unchanged.

Reason for Failure

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

`[LMI_UNSPEC]`
: Unknown or unspecified.

`[LMI_DISC]`
: Disconnected.

`[LMI_EVENT]`
: Protocol-specific event occurred.

`[LMI_FATALERR]`
: Device has become unusable.

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

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

`[LMI_SYSERR]`
: UNIX system error.

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

4.2.6 Link Option Management Service Primitives

The link option management service primitives provide another mechanism for options management separate from the local management interface (i.e. the `LMI_OPTMGMT_REQ` and `LMI_OPTMGMT_ACK` primitives). These service primitives are not currently supported by any SLS provider and their use is *deprecated*.

These service primitives implement the link option management service (see Section 3.2.6 [Link Option Management Service], page 30).

4.2.6.1 SL_OPTMGMT_REQ

Description

This SLS user originated primitive requests that the SLS provider options be managed.

Format

The link 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 link option management request service primitive contains the following parameters:

lmi_primitive
: Specifies the service primitive type. Always `SL_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 which 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.

Rules

Response

The link 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 link options management service with an `SL_OPTMGMT_ACK` primitive containing the link 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 link options management service with an `LMI_ERROR_ACK` primitive containing the error. The state remains unchanged.

Reason 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.

4.2.6.2 SL_OPTMGMT_ACK

Description

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

Format

The link 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 link option management acknowledgement service primitive contains the following parameters:

lmi_primitive
: Indicates the service primitive type. Always `SL_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 link 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 ore 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 `SL_OPTMGMT_REQ` primitive.

New State

The new state remains unchanged.

Rules

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

— When the *lmi_mgmt_flags* field in the `SL_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.

4.2.7 Event Notification Service Primitives

The event notification service primitives provide another mechanism for event notification separate from the local management interface (i.e. the `LMI_EVENT_IND` primitive). These service primitives are not currently supported by any SLS provider and their use is *deprecated*.
These service primitives implement the event notification service (see Section 3.2.7 [Event Notification Service], page 30).

4.2.7.1 SL_NOTIFY_REQ

Description

This SLS user originated primitives requests that the SLS provider register the SLS user for various events.

Format

Not documented.

Parameters

sl_primitive
: Specifies the service primitive type. Always `SL_NOTIFY_REQ`.

State

Any state.

New State

Unchanged.

Response

This primitive does not require receipt acknolwedgement from the SLS provider.

- **Successful:** When successful, the events are registered and no acknowledgement is required. The state remains unchanged.
- **Unsuccessful (non-fatal errors):** When unsuccessful, the SLS provider generates a negative acknowledgement using a `LMI_ERROR_ACK` primitive containing the error and reason for failure. The state remains unchanged.

Reason for Failure

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

`[LMI_UNSPEC]`
: Unknown or unspecified.

`[LMI_DISC]`
: Disconnected.

`[LMI_EVENT]`
: Protocol-specific event occurred.

`[LMI_FATALERR]`
: Device has become unusable.

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

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

`[LMI_SYSERR]`
: UNIX system error.

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

Notes

This primitive is *deprecated* and has been replaced by the local management inteface event reporting service discussed in Section 3.1.8 [Event Reporting Service], page 19.

4.2.7.2 SL_NOTIFY_IND

Description

This SLS provider originated primitive indicates that an event for which the SLS provider has registered has occurred.

Format

Not documented.

Parameters

sl_primitive
: Specifies the service primitive type. Always `SL_NOTIFY_IND`.

State

Any state.

New State

Unchanged.

Rules

The SLS provider observes the following rules when issuing the event notification indication service primtiive:

— This primitive is only issued by the SLS provider for event for which the SLS user has explicitly registered with the `SL_NOTIFY_REQ` primitive.

— Specific events are provider-specific.

Notes

This primitive is *deprecated* and has been replaced by the local management inteface event reporting service discussed in Section 3.1.8 [Event Reporting Service], page 19.

5 Diagnostics Requirements

Two error handling facilities should be provided to the SLS 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 SLS interface as seen by the SLS user and provide the user with the option of reissuing the SL 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 SLS interface the same as it was before the SL 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 SL provider when it detects errors that are not correctable by the SL user, or if it is unable to report a correctible error to the SLS 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 SLS 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.

Appendix A 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__ */
```

Appendix B SLI Header File Listing

```
#ifndef __SS7_SLI_H__
#define __SS7_SLI_H__

typedef lmi_long sl_long;
typedef lmi_ulong sl_ulong;
typedef lmi_ushort sl_ushort;
typedef lmi_uchar sl_uchar;

#define SL_PROTO_BASE                           64

#define SL_DSTR_FIRST                          (  1 + SL_PROTO_BASE)
#define SL_PDU_REQ                             (  1 + SL_PROTO_BASE)
#define SL_EMERGENCY_REQ                       (  2 + SL_PROTO_BASE)
#define SL_EMERGENCY_CEASES_REQ                (  3 + SL_PROTO_BASE)
#define SL_START_REQ                           (  4 + SL_PROTO_BASE)
#define SL_STOP_REQ                            (  5 + SL_PROTO_BASE)
#define SL_RETRIEVE_BSNT_REQ                   (  6 + SL_PROTO_BASE)
#define SL_RETRIEVAL_REQUEST_AND_FSNC_REQ      (  7 + SL_PROTO_BASE)
#define SL_CLEAR_BUFFERS_REQ                   (  8 + SL_PROTO_BASE)
#define SL_CLEAR_RTB_REQ                       (  9 + SL_PROTO_BASE)
#define SL_CONTINUE_REQ                        ( 10 + SL_PROTO_BASE)
#define SL_LOCAL_PROCESSOR_OUTAGE_REQ          ( 11 + SL_PROTO_BASE)
#define SL_RESUME_REQ                          ( 12 + SL_PROTO_BASE)
#define SL_CONGESTION_DISCARD_REQ              ( 13 + SL_PROTO_BASE)
#define SL_CONGESTION_ACCEPT_REQ               ( 14 + SL_PROTO_BASE)
#define SL_NO_CONGESTION_REQ                   ( 15 + SL_PROTO_BASE)
#define SL_POWER_ON_REQ                        ( 16 + SL_PROTO_BASE)
#define SL_OPTMGMT_REQ                         ( 17 + SL_PROTO_BASE)
#define SL_NOTIFY_REQ                          ( 18 + SL_PROTO_BASE)
#define SL_DSTR_LAST                           ( 18 + SL_PROTO_BASE)

#define SL_USTR_LAST                           ( -1 - SL_PROTO_BASE)
#define SL_PDU_IND                             ( -1 - SL_PROTO_BASE)
#define SL_LINK_CONGESTED_IND                  ( -2 - SL_PROTO_BASE)
#define SL_LINK_CONGESTION_CEASED_IND          ( -3 - SL_PROTO_BASE)
#define SL_RETRIEVED_MESSAGE_IND               ( -4 - SL_PROTO_BASE)
#define SL_RETRIEVAL_COMPLETE_IND              ( -5 - SL_PROTO_BASE)
#define SL_RB_CLEARED_IND                      ( -6 - SL_PROTO_BASE)
#define SL_BSNT_IND                            ( -7 - SL_PROTO_BASE)
#define SL_IN_SERVICE_IND                      ( -8 - SL_PROTO_BASE)
#define SL_OUT_OF_SERVICE_IND                  ( -9 - SL_PROTO_BASE)
#define SL_REMOTE_PROCESSOR_OUTAGE_IND         (-10 - SL_PROTO_BASE)
#define SL_REMOTE_PROCESSOR_RECOVERED_IND      (-11 - SL_PROTO_BASE)
#define SL_RTB_CLEARED_IND                     (-12 - SL_PROTO_BASE)
#define SL_RETRIEVAL_NOT_POSSIBLE_IND          (-13 - SL_PROTO_BASE)
#define SL_BSNT_NOT_RETRIEVABLE_IND            (-14 - SL_PROTO_BASE)
#define SL_OPTMGMT_ACK                         (-15 - SL_PROTO_BASE)
#define SL_NOTIFY_IND                          (-16 - SL_PROTO_BASE)
#define SL_LOCAL_PROCESSOR_OUTAGE_IND          (-17 - SL_PROTO_BASE)
#define SL_LOCAL_PROCESSOR_RECOVERED_IND       (-18 - SL_PROTO_BASE)
#define SL_USTR_FIRST                          (-18 - SL_PROTO_BASE)

/*
```

```
 *  SLI PROVIDER STATE
 */
#define SLS_POWER_OFF                   0
#define SLS_OUT_OF_SERVICE              1
#define SLS_NOT_ALIGNED                 2
#define SLS_INITIAL_ALIGNMENT           3
#define SLS_PROVING                     4
#define SLS_ALIGNED_READY               5
#define SLS_ALIGNED_NOT_READY           6
#define SLS_IN_SERVICE                  7
#define SLS_PROCESSOR_OUTAGE            8

/*
 *  SLI PROVIDER FLAGS
 */
#define SLF_LOC_PROC_OUT        (1<< 0)
#define SLF_REM_PROC_OUT        (1<< 1)
#define SLF_LOC_IN_SERV         (1<< 2)
#define SLF_REM_IN_SERV         (1<< 3)
#define SLF_LOC_BUSY            (1<< 4)
#define SLF_REM_BUSY            (1<< 5)
#define SLF_LOC_EMERG           (1<< 6)
#define SLF_EMERGENCY           SLF_LOC_EMERG
#define SLF_REM_EMERG           (1<< 7)
#define SLF_RECV_MSU            (1<< 8)
#define SLF_SEND_MSU            (1<< 9)
#define SLF_CONG_ACCEPT         (1<<10)
#define SLF_CONG_DISCARD        (1<<11)
#define SLF_RTB_FULL            (1<<12)
#define SLF_L3_CONG_DETECT      (1<<13)
#define SLF_L2_CONG_DETECT      (1<<14)
#define SLF_LINK_CONGESTED      SLF_L2_CONG_DETECT
#define SLF_CONTINUE            (1<<15)
#define SLF_LEVEL_3_IND         SLF_CONTINUE
#define SLF_CLEAR_RTB           (1<<16)
#define SLF_NEED_FLUSH          (1<<17)
#define SLF_WAIT_SYNC           (1<<18)
#define SLF_REM_ALIGN           (1<<19)

/*
 *  SLI PROTOCOL PRIMITIVES
 */

/*
 *  SL_PDU_REQ, optional M_PROTO type, with M_DATA block(s)
 */
typedef struct {
        sl_long sl_primitive;
        sl_ulong sl_mp;
} sl_pdu_req_t;

/*
 *  SL_PDU_IND, optional M_PROTO type, with M_DATA block(s)
 */
typedef struct {
        sl_long sl_primitive;
```

```
        sl_ulong sl_mp;
} sl_pdu_ind_t;

/*
 *  PROTOCOL CONTROL PRIMITIVES
 */

/*
 *  SL_EMERGENCY_REQ, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
} sl_emergency_req_t;

/*
 *  SL_EMERGENCY_CEASES_REQ, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
} sl_emergency_ceases_req_t;

/*
 *  SL_START_REQ, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
} sl_start_req_t;

/*
 *  SL_STOP_REQ, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
} sl_stop_req_t;

/*
 *  SL_RETRIEVE_BSNT_REQ, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
} sl_retrieve_bsnt_req_t;

/*
 *  SL_RETRIEVAL_REQUEST_AND_FSNC_REQ, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
        sl_ulong sl_fsnc;
} sl_retrieval_req_and_fsnc_t;

/*
 *  SL_CLEAR_BUFFERS_REQ, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
} sl_clear_buffers_req_t;
```

```
/*
 *  SL_CLEAR_RTB_REQ, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
} sl_clear_rtb_req_t;

/*
 *  SL_CONTINUE_REQ, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
} sl_continue_req_t;

/*
 *  SL_LOCAL_PROCESSOR_OUTAGE_REQ, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
} sl_local_proc_outage_req_t;

/*
 *  SL_RESUME_REQ, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
} sl_resume_req_t;

/*
 *  SL_CONGESTION_DISCARD_REQ, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
} sl_cong_discard_req_t;

/*
 *  SL_CONGESTION_ACCEPT_REQ, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
} sl_cong_accept_req_t;

/*
 *  SL_NO_CONGESTION_REQ, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
} sl_no_cong_req_t;

/*
 * SL_POWER_ON_REQ, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
} sl_power_on_req_t;
```

```
/*
 *  SL_LINK_CONGESTED_IND, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
        sl_ulong sl_timestamp;
        sl_ulong sl_cong_status;        /* congestion status */
        sl_ulong sl_disc_status;        /* discard status */
} sl_link_cong_ind_t;

/*
 *  SL_LINK_CONGESTION_CEASED_IND, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
        sl_ulong sl_timestamp;
        sl_ulong sl_cong_status;        /* congestion status */
        sl_ulong sl_disc_status;        /* discard status */
} sl_link_cong_ceased_ind_t;

/*
 *  SL_RETRIEVED_MESSAGE_IND, M_PROTO or M_PCPROTO type with M_DATA block(s)
 */
typedef struct {
        sl_long sl_primitive;
        sl_ulong sl_mp;
} sl_retrieved_msg_ind_t;

/*
 *  SL_RETRIEVAL_COMPLETE_IND, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
        sl_ulong sl_mp;
} sl_retrieval_comp_ind_t;

/*
 *  SL_RETRIEVAL_NOT_POSSIBLE_IND, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
} sl_retrieval_not_poss_ind_t;

/*
 *  SL_RB_CLEARED_IND, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
} sl_rb_cleared_ind_t;

/*
 *  SL_BSNT_IND, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
```

```
        sl_ulong sl_bsnt;
} sl_bsnt_ind_t;

/*
 *  SL_BSNT_NOT_RETRIEVABLE_IND, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
        sl_ulong sl_bsnt;
} sl_bsnt_not_retr_ind_t;

/*
 *  SL_IN_SERVICE_IND, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
} sl_in_service_ind_t;

/*
 *  SL_OUT_OF_SERVICE_IND, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
        sl_ulong sl_timestamp;
        sl_ulong sl_reason;
} sl_out_of_service_ind_t;

/*
 *  These reasons for failure as so that upstream module can
 *  collect statistics per link per ITU-T Q.752 Table 1
 *  requirements.
 */
#define SL_FAIL_UNSPECIFIED             0x0001
#define SL_FAIL_CONG_TIMEOUT            0x0002
#define SL_FAIL_ACK_TIMEOUT             0x0004
#define SL_FAIL_ABNORMAL_BSNR           0x0008
#define SL_FAIL_ABNORMAL_FIBR           0x0010
#define SL_FAIL_SUERM_EIM               0x0020
#define SL_FAIL_ALIGNMENT_NOT_POSSIBLE  0x0040
#define SL_FAIL_RECEIVED_SIO            0x0080
#define SL_FAIL_RECEIVED_SIN            0x0100
#define SL_FAIL_RECEIVED_SIE            0x0200
#define SL_FAIL_RECEIVED_SIOS           0x0400
#define SL_FAIL_T1_TIMEOUT              0x0800

/*
 *  SL_REMOTE_PROCESSOR_OUTAGE_IND, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
        sl_ulong sl_timestamp;
} sl_rem_proc_out_ind_t;

/*
 *  SL_REMOTE_PROCESSOR_RECOVERED_IND, M_PROTO or M_PCPROTO type
 */
```

```
typedef struct {
        sl_long sl_primitive;
        sl_ulong sl_timestamp;
} sl_rem_proc_recovered_ind_t;

/*
 *  SL_RTB_CLEARED_IND, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
} sl_rtb_cleared_ind_t;

/*
 *  SL_LOCAL_PROCESSOR_OUTAGE_IND, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
        sl_ulong sl_timestamp;
} sl_loc_proc_out_ind_t;

/*
 *  SL_LOCAL_PROCESSOR_RECOVERED_IND, M_PROTO or M_PCPROTO type
 */
typedef struct {
        sl_long sl_primitive;
        sl_ulong sl_timestamp;
} sl_loc_proc_recovered_ind_t;

/*
 *  Generic single argument type
 */
typedef struct {
        sl_ulong sl_cmd;
        sl_ulong sl_arg;
} sl_cmd_arg_t;

/*
 *  Generic double argument type
 */
typedef struct {
        sl_ulong sl_cmd;
        sl_ulong sl_arg1;
        sl_ulong sl_arg2;
} sl_cmd_2arg_t;

/*
 *  Generic triple argument type
 */
typedef struct {
        sl_ulong sl_cmd;
        sl_ulong sl_arg1;
        sl_ulong sl_arg2;
        sl_ulong sl_arg3;
} sl_cmd_3arg_t;

union SL_primitives {
```

```
        sl_long sl_primitive;
        sl_cmd_arg_t cmd_arg;
        sl_cmd_2arg_t cmd_2arg;
        sl_cmd_3arg_t cmd_3arg;
        sl_pdu_req_t pdu_req;
        sl_pdu_ind_t pdu_ind;
        sl_emergency_req_t emergency_req;
        sl_emergency_ceases_req_t emergency_ceases_req;
        sl_start_req_t start_req;
        sl_stop_req_t stop_req;
        sl_retrieve_bsnt_req_t retrieve_bsnt_req;
        sl_retrieval_req_and_fsnc_t retrieval_req_and_fsnc;
        sl_resume_req_t resume_req;
        sl_continue_req_t continue_req;
        sl_clear_buffers_req_t clear_buffers_req;
        sl_clear_rtb_req_t clear_rtb_req;
        sl_local_proc_outage_req_t local_proc_outage_req;
        sl_cong_discard_req_t cong_discard_req;
        sl_cong_accept_req_t cong_accept_req;
        sl_no_cong_req_t no_cong_req;
        sl_power_on_req_t power_on_req;
        sl_link_cong_ind_t link_cong_ind;
        sl_link_cong_ceased_ind_t link_cong_ceased_ind;
        sl_retrieved_msg_ind_t retrieved_msg_ind;
        sl_retrieval_comp_ind_t retrieval_comp_ind;
        sl_retrieval_not_poss_ind_t retrieval_not_poss_ind;
        sl_rb_cleared_ind_t rb_cleared_ind;
        sl_bsnt_ind_t bsnt_ind;
        sl_bsnt_not_retr_ind_t bsnt_not_retr_ind;
        sl_in_service_ind_t in_service_ind;
        sl_out_of_service_ind_t out_of_service_ind;
        sl_rem_proc_out_ind_t rem_proc_out_ind;
        sl_rem_proc_recovered_ind_t rem_proc_recovered_ind;
        sl_rtb_cleared_ind_t rtb_cleared_ind;
        sl_loc_proc_out_ind_t loc_proc_out_ind;
        sl_loc_proc_recovered_ind_t loc_proc_recovered_ind;
};

typedef union SL_primitives sl_prim_t;

#define SL_CMD_ARG_SIZE                 sizeof(sl_cmd_arg_t)
#define SL_CMD_2ARG_SIZE                sizeof(sl_cmd_2arg_t)
#define SL_CMD_3ARG_SIZE                sizeof(sl_cmd_3arg_t)
#define SL_PDU_REQ_SIZE                 sizeof(sl_pdu_req_t)
#define SL_PDU_IND_SIZE                 sizeof(sl_pdu_ind_t)
#define SL_EMERGENCY_REQ_SIZE           sizeof(sl_emergency_req_t)
#define SL_EMERGENCY_CEASES_REQ_SIZE    sizeof(sl_emergency_ceases_req_t)
#define SL_START_REQ_SIZE               sizeof(sl_start_req_t)
#define SL_STOP_REQ_SIZE                sizeof(sl_stop_req_t)
#define SL_RETRIEVE_BSNT_REQ_SIZE       sizeof(sl_retrieve_bsnt_req_t)
#define SL_RETRIEVAL_REQ_AND_FSNC_SIZE  sizeof(sl_retrieval_req_and_fsnc_t)
#define SL_RESUME_REQ_SIZE              sizeof(sl_resume_req_t)
#define SL_CONTINUE_REQ_SIZE            sizeof(sl_continue_req_t)
#define SL_CLEAR_BUFFERS_REQ_SIZE       sizeof(sl_clear_buffers_req_t)
#define SL_CLEAR_RTB_REQ_SIZE           sizeof(sl_clear_rtb_req_t)
#define SL_LOCAL_PROC_OUTAGE_REQ_SIZE   sizeof(sl_local_proc_outage_req_t)
```

```
#define SL_CONG_DISCARD_REQ_SIZE        sizeof(sl_cong_discard_req_t)
#define SL_CONG_ACCEPT_REQ_SIZE         sizeof(sl_cong_accept_req_t)
#define SL_NO_CONG_REQ_SIZE             sizeof(sl_no_cong_req_t)
#define SL_POWER_ON_REQ_SIZE            sizeof(sl_power_on_req_t)
#define SL_LINK_CONG_IND_SIZE           sizeof(sl_link_cong_ind_t)
#define SL_LINK_CONG_CEASED_IND_SIZE    sizeof(sl_link_cong_ceased_ind_t)
#define SL_RETRIEVED_MSG_IND_SIZE       sizeof(sl_retrieved_msg_ind_t)
#define SL_RETRIEVAL_COMP_IND_SIZE      sizeof(sl_retrieval_comp_ind_t)
#define SL_RETRIEVAL_NOT_POSS_IND_SIZE  sizeof(sl_retrieval_not_poss_ind_t)
#define SL_RB_CLEARED_IND_SIZE          sizeof(sl_rb_cleared_ind_t)
#define SL_BSNT_IND_SIZE                sizeof(sl_bsnt_ind_t)
#define SL_BSNT_NOT_RETR_IND_SIZE       sizeof(sl_bsnt_not_retr_ind_t)
#define SL_IN_SERVICE_IND_SIZE          sizeof(sl_in_service_ind_t)
#define SL_OUT_OF_SERVICE_SIZE          sizeof(sl_out_of_service_ind_t)
#define SL_REM_PROC_OUT_IND_SIZE        sizeof(sl_rem_proc_out_ind_t)
#define SL_REM_PROC_RECOVERED_IND_SIZE  sizeof(sl_rem_proc_recovered_ind_t)
#define SL_RTB_CLEARED_IND_SIZE         sizeof(sl_rtb_cleared_ind_t)
#define SL_LOC_PROC_OUT_IND_SIZE        sizeof(sl_loc_proc_out_ind_t)
#define SL_LOC_PROC_RECOVERED_IND_SIZE  sizeof(sl_loc_proc_recovered_ind_t)

#define SL_OPT_PROTOCOL         LMI_OPT_PROTOCOL
#define SL_OPT_STATISTICS       LMI_OPT_STATISTICS
#define SL_OPT_CONFIG           3       /* use struct sl_config */
#define SL_OPT_STATEM           4       /* use struct sl_statem */
#define SL_OPT_STATS            5       /* use struct sl_stats */

#endif                          /* __SS7_SLI_H__ */
```

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

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
PPA	Physical Point of Attachment
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
SS7	Signalling System No. 7

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 157. The text of this manual is licensed under the [GNU Free Documentation License], page 167, 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

The GNU Affero General Public License.
Version 3, 19 November 2007

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

M

O

P

S