Media Gateway Interface (MGI) Specification

Media Gateway Interface (MGI) 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 Media Gateway Interface (MGI) for OpenSS7. It contains recommendations on software architecture as well as platform and system applicability of the Media Gateway Interface (MGI). It provides abstraction of the Media Gateway (MG) interface to these components as well as providing a basis for Media Gateway control for other Media Gateway 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 137). 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 147) 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 Media Gateway Interface (MGI) for OpenSS7. It contains recommendations on software architecture as well as platform and system applicability of the Media Gateway Interface (MGI).

This document specifies a Media Gateway Interface (MGI) Specification in support of the OpenSS7 Media Gateway (MG) protocol stacks. It provides abstraction of the Media Gateway interface to these components as well as providing a basis for Media Gateway control for other Media Gateway protocols.

Purpose

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

Audience

The audience for this document is software developers, maintainers and users and integrators of the Media Gateway Interface (MGI). 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: mgi.texi,v $
Revision 1.1.2.3  2011-02-07 02:21:40  brian
- updated manuals

Revision 1.1.2.2  2010-03-10 08:42:18  brian
- added Optranex files

Revision 1.1.2.1  2010-02-22 14:25:53  brian
- added new documentation files
```

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 Media Gateway Interface (MGI) definition. The Media Gateway Interface (MGI) enables the user of a media gateway service to access and use any of a variety of conforming media gateway providers without specific knowledge of the provider's protocol. The service interface is designed to support any network media gateway protocol. This interface only specifies access to media gateway service providers, and does not address issues concerning media gateway management, protocol performance, and performance analysis tools.

This specification assumes that the reader is familiar with ITU-T state machines and media gateway interface (e.g. H.248) and STREAMS.

1.1 Related Documentation

— **ITU-T Recommendation H.248**

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

1.1.1 Role

This document specifies an interface that supports the services provided by the *Media Gateway* for ITU-T, ANSI and ETSI applications as described in ITU-T Recommendation H.248. These specifications are targeted for use by developers and testers of protocol modules that require media gateway 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.

ISO International Organization for Standardization

OSI Open Systems Interconnection

QOS Quality of Service

STREAMS A communication services development facility first available with UNIX System V Release 3.

2 The Media Gateway Layer

The Media Gateway Layer provides the means to manage the association of MG-User connections. It is responsible for the routing and management of data to and from media gateway connections between MG-user entities.

2.1 Model of the MGI

The MGI defines the services provided by the media gateway layer to the media gateway user at the boundary between the media gateway provider (MG) and the media gateway user (MGC) entity. The interface consists of a set of primitives defined as STREAMS messages that provide access to the media gateway layer services, and are transferred between the MGS user entity (MGC) and the MGS provider (MG). These primitives are of two types; ones that originate from the MGS user, and others that originate from the MGS provider. The primitives that originate from the MGS user make requests to the MGS provider, or respond to an indication of an event of the MGS provider. The primitives that originate from the MGS provider are either confirmations of a request or are indications to the MGS user that an event has occurred. Figure 2.1 show the model of the MGI.

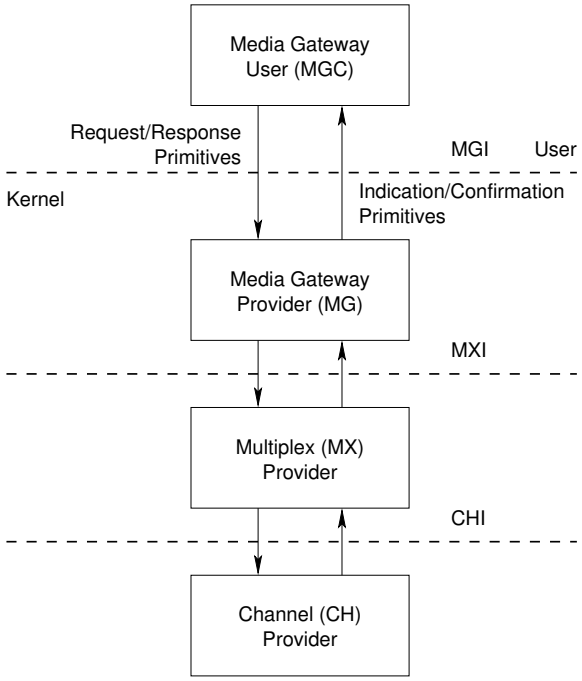

Figure 2.1: *Model of the MGI*

The MGI allows the MGS provider to be configured with any media gateway layer user (such as a swtiching application) that also conforms to the MGI. A media gateway layer user can also be a user program that conforms to the MGI and accesses the MGS provider via **putmsg**(2s) and **getmsg**(2s) system calls. A typical configuration, however, is to have a switching user-space application using the media gateway layer. Nevertheless, another typical configuration is to have an H.248 multiplexing driver connected to the MG provider Streams.

2.2 MGI Services

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

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

2.2.1 Local Management

Local management services are listed in Table 2.1.

Phase	Service	Primitives
Local Management	Acknowledgement	MG_OK_ACK, MG_ERROR_ACK
	Information Reporting	MG_INFO_REQ, MG_INFO_ACK
	PPA Attachment	MG_ATTACH_REQ, MG_DETACH_REQ, MG_OK_ACK
	Initialization	MG_ENABLE_REQ, MG_ENABLE_CON, MG_DISABLE_REQ, MG_DISABLE_CON
	Options Management	MG_OPTMGMT_REQ, MG_OPTMGMT_ACK
	Event Reporting	MG_ERROR_IND, MG_STATS_IND, MG_EVENT_IND

Table 2.1: *Local Management Services*

The local management services interface is described in Section 3.1 [Local Management Services], page 15, and the primitives are detailed in Section 4.1 [Local Management Service Primitives], page 31. The local management services interface is defined by the `sys/mgi.h` header file (see Appendix C [MGI Header Files], page 97).

2.2.2 Protocol

Protocol services are listed in Table 2.2.

Phase	Service	Primitives
Session Establishment	Join	MG_JOIN_REQ, MG_JOIN_CON
	Connection	MG_CONN_REQ, MG_CONN_CON
Session Control	Action	MG_ACTION_REQ, MG_ACTION_CON, MG_ABORT_REQ, MG_ACTION_IND
	Data Transfer	MG_DATA_REQ, MG_DATA_IND
Session Release	Disconnection	MG_DISCON_REQ, MG_DISCON_IND, MG_DISCON_CON
	Leave	MG_LEAVE_REQ, MG_LEAVE_IND, MG_LEAVE_CON

Table 2.2: *Protocol Services*

The protocol services interface is described in Section 3.2 [Protocol Services], page 21, and the primitives are detailed in Section 4.2 [Protocol Service Primitives], page 56. The protocol services interface is defined by the `sys/mgi.h` header file (see Appendix C [MGI Header Files], page 97).

2.3 Purpose of the MGI

The MGI is typically implemented as a device driver connecting and controlling a TDM (Time Division Mutliplexing) device that provides access to multiplexed media streams, and a network device that provides packet-based media streams. This is a high level control interface that can be used in conjunction with a media gateway control protocol or an integrated media gateway controller to provide media gateway or integrated softswitch functions.

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

2.4 Media Gateway Addressing

Each use of MGI must establish an identity to communicate with other media gateway users. The MGS user must identify the physical media over which communication will occur. This is particularly evident on a system that is attached to multiple physical media. Figure 2.2 illustrates the identification approach, which is explained below.

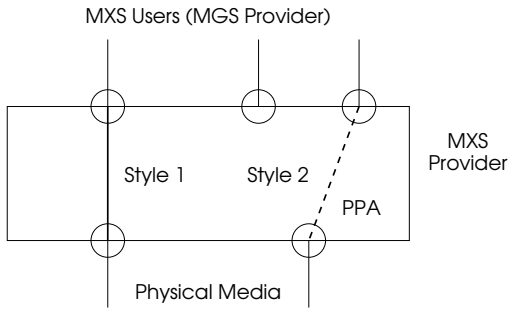

Figure 2.2: *Media Gateway Addressing Components*

2.4.1 Physical Attachment Identification

The physical point of attachment (PPA in Figure 2.2) is the point at which a system interface attaches itself to a physical communications medium (a channel, facility or network interface). All communication on that physical medium funnels through the PPA associated with that physical medium. On systems where a MGS provider supports more than one physical medium, the MGS user must identify the medium through which it will communicate. A PPA is identified by a unique PPA identifier.

For media that supports physical layer multiplexing of multiple channels over a single physical medium (such as the B and D channels of ISDN), the PPA identifier must identify the specific channel(s) over which communication will occur. See also [Multiplex Media], page 13.

Unlike the Data Link Provider Interface (DLPI), which also uses the concept of a PPA, MGI does not define a SAP for a MGS user.

Once a Stream has been associated with a PPA, all messages received on that medium are controlled by the attached MGS user. Only one major/minor device number combination (Stream head) can be associated with a given PPA and active for a range of channels at any point in time.

2.4.2 MGS Provider Styles

Two styles of MGS provider are defined by MGI, distinguished by the way they enable a MGS user to choose a particular PPA.

2.4.2.1 Style 1 MGS Provider

The *Style 1* provider assigns a PPA based on the major/minor device the MGS user opened. One possible implementation of a *Style 1* driver would reserve a major device for each PPA the media gateway device driver would support. This would allow the STREAMS clone open feature to be used for each PPA configured. This style of provider is appropriate when few PPAs will be supported.

For example, a PCIe card that supports four SONET/SDH ports could assign a major device number to the card diver and a minor device number to each of the ports on each card in the system. To establish a Stream to a MGS provider for a given port, the minor device number '`1`' or '`2`' could be opened for port '`1`' or '`2`' on card '`1`', minor device number '`3`' or '`4`' could be opened for port '`1`' or '`2`' on card '`2`', and so on. One major device number for the driver could easily support 127 cards in a system, which is not possible for typical PCIe systems and, therefore, is ample.

Style 1 providers do not user the `MG_ATTACH_REQ` and `MG_DETACH_REQ` primitives and when freshly opened are in the `MGS_ATTACHED` state. That is, as illustrated in Figure 2.2, the *Style 1* MGS provider associates the Stream with the PPA during the `open(2s)` system call.

2.4.2.2 Style 2 MGS Provider

If the number of PPAs as MGS provider will support is large, a *Style 2* provider implementation is more suitable. The *Style 2* provider requires a MGS user to explicitly identify the desired PPA using a special attach service primitive. For a *Style 2* driver, the `open(2s)` system call creates a Stream between the MGS user and MGS provider, and the attach primitive then associated a particular PPA with that Stream. The format of the PPA identifier is specific to the MGS provider, and should be described in the provider-specific addendum documentation.

The MGS user uses the support primitvies(`MG_ATTACH_REQ`, `MG_ENABLE_REQ`) to associate a Stream with a given Physical Point of Appearance. *Style 2* MGS providers, when freshly opened, are in the `MGS_DETACHED` state. That is, the *Style 2* MGS provider does not associate the Stream with the

PPA during the open(2s) call, but only later when the MG_ATTACH_REQ primitive is issued by the MGS user.

2.4.3 Multiplex Media

To accommodate multiplexed media and multi-media channels, there are three kinds of PPA address:

1. A discrete PPA that specifies a non-multiplexed medium.

   This is the normal case of a *Style 1* or *Style 2* MGS provider supporting access to a non-multiplexed medium. An example is a MGS provider supporting access to a FXO/FXS interface.

2. A specific PPA that specifies a single channel to a multiplexed medium.

   This is again the normal case of a *Style 1* or *Style 2* MGS provider supporting access to a specific channel in a multiplexed medium. An example is a MGS provider supporting access to channel 16 of an E1 interface.

3. A general PPA that specifies a channel group for a multiplexed medium.

   This is th case of a *Style 1* or *Style 2* MGS provider supporting access to multiple channels in a multiplexed medium. An example is a MGS provider supporting statistically multiplexed channel access to a full or fractional T1 facilitiy. Another example is access to the left and right channels of a stereo program.

In the case of a general PPA, as enumerated in *3* above, some additional information is required to identify which slots in the group of channle forming the multiplex are associatedw the the MGS user Stream. This additional information is provided using the *mg_slot* parameter to the MG_CONN_REQ, MG_CONN_CON, MG_DATA_REQ, MG_DATA_IND, MG_EVENT_IND, MG_DISCON_REQ, MG_DISCON_CON and MG_DISCON_IND primitives.[1]

2.5 Media Gateway Parameters

[1] Note that it is the ability of the Media Gateway Interface to support fractional E1/T1 that distinguishes it from similar interfaces such as the SDLI and CDI.

3 MGI Services Definition

3.1 Local Management Services

3.1.1 Acknowledgement Service

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

- `MG_OK_ACK`: The `MG_OK_ACK` message is used by the MGS provider to indicate successful receipt and completion of a service primitive request that requires positive acknowledgement.
- `MG_ERROR_ACK`: The `MG_ERROR_ACK` message is used by the MGS 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.

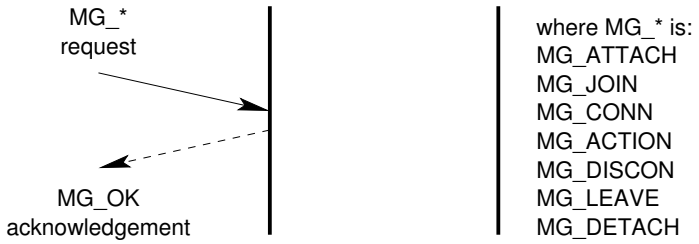

Figure 3.1: *Message Flow: Successful Acknowledgement Service*

As illustrated in Figure 3.1, the service primitives for which a positive acknowledgement may be returned are the `MG_ATTACH_REQ` and `MG_DETACH_REQ`.
An unsuccessful invocation of the acknowledgement service is illustrated in Figure 3.2.

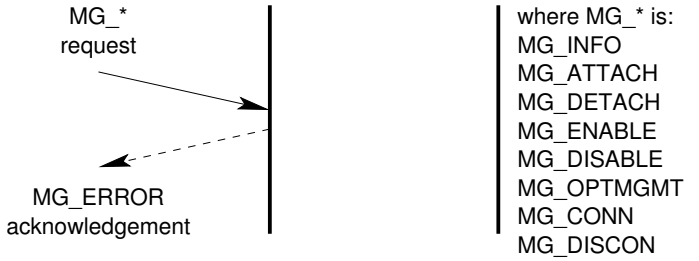

Figure 3.2: *Message Flow: Unsuccessful Acknowledgement Service*

As illustrated in Figure 3.2, the service primitives for which a negative acknowledgement may be returned are the `MG_INFO_REQ`, `MG_ATTACH_REQ`, `MG_DETACH_REQ`, `MG_ENABLE_REQ`, `MG_DISABLE_REQ` and `MG_OPTMGMT_REQ` messages.

3.1.2 Information Reporting Service

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

- `MG_INFO_REQ`: The `MG_INFO_REQ` message is used by the MGS user to request information about the MGS provider.
- `MG_INFO_ACK`: The `MG_INFO_ACK` message is issued by the MGS provider to provide requested information about the MGS provider.

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

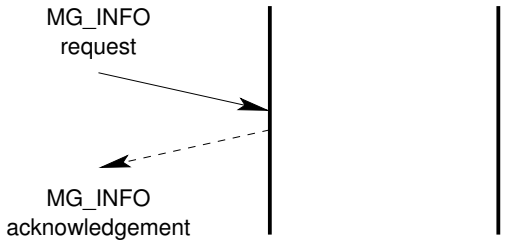

Figure 3.3: *Message Flow: Successful Information Reporting Service*

3.1.3 Physical Point of Attachment Service

The local management interface provides the MGS 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 MGS provider:[1]

Style 1 MGS Provider

A *Style 1* MGS 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* MGS provider Streams start life in the `MG_DISABLED` state.

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

Style 2 MGS Provider

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

[1] See also Section 2.4 [Media Gateway Addressing], page 11.

Freshly opened *Style 2* MGS provider Streams start life in the `MG_UNATTACHED` state.

This approach is suitable for MGS 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 MGS user with the ability to attach a *Style 2* MGS provider Stream to a physical point of appearance (PPA).

- `MG_ATTACH_REQ`: The `MG_ATTACH_REQ` message is issued by the MGS user to request that a *Style 2* MGS provider Stream be attached to a specified physical point of appearance (PPA).
- `MG_OK_ACK`: Upon successful receipt and processing of the `MG_ATTACH_REQ` message, the MGS provider acknowledges the success of the service completion with a `MG_OK_ACK` message.
- `MG_ERROR_ACK`: Upon successful receipt but failure to process the `MG_ATTACH_REQ` message, the MGS provider acknowledges the failure of the service completion with a `MG_ERROR_ACK` message.

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

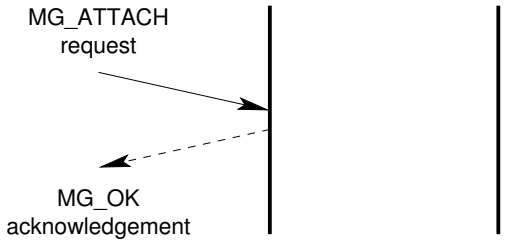

Figure 3.4: *Message Flow: Successful Attachment Service*

3.1.3.2 PPA Detachment Service

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

- `MG_DETACH_REQ`: The `MG_DETACH_REQ` message is issued by the MGS user to request that a *Style 2* MGS provider Stream be detached from the attached physical point of appearance (PPA).
- `MG_OK_ACK`: Upon successful receipt and processing of the `MG_DETACH_REQ` message, the MGS provider acknowledges the success of the service completion with a `MG_OK_ACK` message.
- `MG_ERROR_ACK`: Upon successful receipt but failure to process the `MG_DETACH_REQ` message, the MGS provider acknowledges the failure of the service completion with a `MG_ERROR_ACK` message.

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

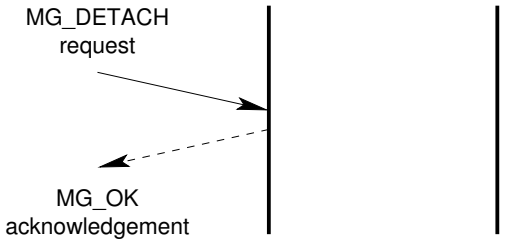

Figure 3.5: *Message Flow: Successful Detachment Service*

3.1.4 Initialization Service

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

- `MG_ENABLE_REQ`: The `MG_ENABLE_REQ` message is issued by the MGS user to request that the protocol service interface be enabled.
- `MG_ENABLE_CON`: Upon successful enabling of the protocol service interface, the MGS provider acknowledges successful completion of the service by issuing a `MG_ENABLE_CON` message to the MGS user.
- `MG_ERRORK_ACK`: Upon unsuccessful enabling of the protocol service interface, the MGS provider acknowledges the failure to complete the service by issuing an `MG_ERROR_ACK` message to the MGS user.

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

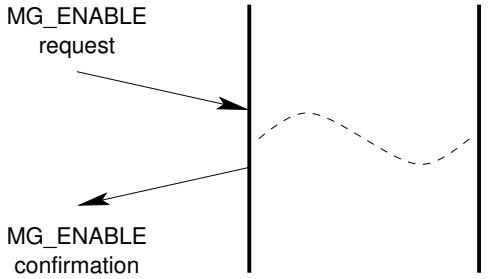

Figure 3.6: *Message Flow: Successful Enable Service*

3.1.4.2 Interface Disable Service

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

- `MG_DISABLE_REQ`: The `MG_DISABLE_REQ` message is issued by the MGS user to request that the protocol service interface be disabled.
- `MG_DISABLE_CON`: Upon successful disabling of the protocol service interface, the MGS provider acknowledges successful completion of the service by issuing a `MG_DISABLE_CON` message to the MGS user.
- `MG_ERRORK_ACK`: Upon unsuccessful disabling of the protocol service interface, the MGS provider acknowledges the failure to complete the service by issuing an `MG_ERROR_ACK` message to the MGS user.
- `MG_DISABLE_IND`: The `MG_DISABLE_IND` message is used by the MGS provider to indicate to the MGS user that the Stream has been autonomously disabled and the cause of the autonomous disabling.

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

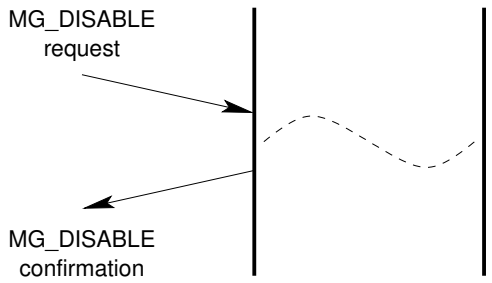

Figure 3.7: *Message Flow: Successful Disable Service*

3.1.5 Options Management Service

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

- `MG_OPTMGMT_REQ`: The MGS user issues a `MG_OPTMGMT_REQ` message when it wishes to interrogate or affect the setting of various generic or provider-specific options associated with the MGS provider for the Stream upon which the message is issued.
- `MG_OPTMGMT_ACK`: Upon successful receipt of the `MG_OPTMGMT_REQ` message, and successful options processing, the MGS provider acknowledges the successful completion of the service with an `MG_OPTMGMT_ACK` message.
- `MG_ERROR_ACK`: Upon successful receipt of the `MG_OPTMGMT_REQ` message, and unsuccessful options processing, the MGS provider acknowledges the failure to complete the service by issuing an `MG_ERROR_ACK` message to the MGS user.

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

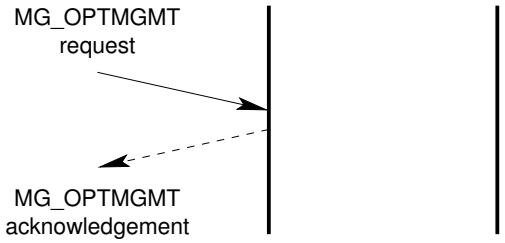

Figure 3.8: *Message Flow: Successful Options Management Service*

3.1.6 Error Reporting Service

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

- `MG_ERROR_IND`: The MGS provider issues the `MG_ERROR_IND` message to the MGS 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.

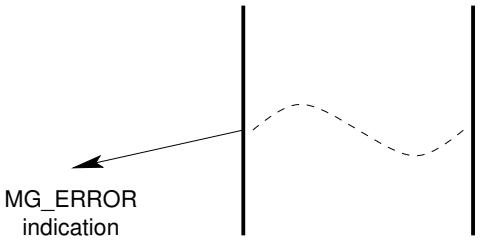

Figure 3.9: *Message Flow: Successful Error Reporting Service*

3.1.7 Statistics Reporting Service

- `MG_STATS_IND`:

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

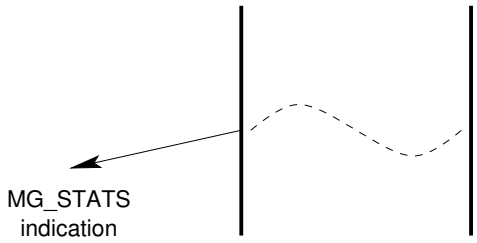

Figure 3.10: *Message Flow: Successful Statistics Reporting Service*

3.1.8 Event Reporting Service

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

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

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

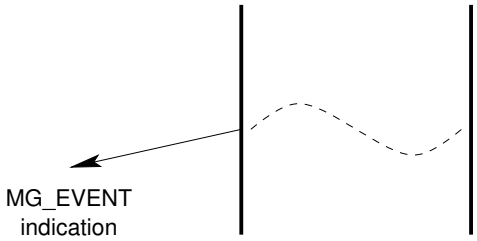

Figure 3.11: *Message Flow: Successful Event Reporting Service*

3.2 Protocol Services

Protocol services are specific to the Media Gateway interface. These services consist of connection services that permit the transmit and receive directions to be connected to or disconnected from the medium, and data transfer services that permit the exchange of bits between MGS users.

The service primitives that implement the protocol services are described in detail in Section 4.2 [Protocol Service Primitives], page 56.

3.2.1 Create Service

The create service provides the ability for the MGS user to create a session context. A session context so created does not have any termination points associated with it. Session contexts created with the *Create Service* can be destroyed with the Section 3.2.11 [Destroy Service], page 29. Session

contexts provide a mechanism whereby termination points can be enjoined into a communications session.

- `MG_CREATE_REQ`: The `MG_CREATE_REQ` primitive is used by the MGS user to create a session context. The newly created session context has no terminations associated with it.
- `MG_CREATE_ACK`: The `MG_CREATE_REQ` primitive is issued by the MGS provider to acknowledge creation of a session context.

Session contexts can be long-lived or short-lived depending on the needs of the MGS user. For example, it is possible for the *Media Gateway* to allocated all of the session contexts that it might need to perform its functions in advance of the need for communications within any given session context. Sessions may alternately be created on demand using the Section 3.2.2 [Join Service], page 22.

3.2.2 Join Service

The join service provides the ability for the MGS user to associate a termination point with a session. It is equivalent to the `Add` service for a single termination point in H.248. The join service may also be used to create a session context, on demand, by specifying a null session context identifier.

- `MG_JOIN_REQ`: The `MG_JOIN_REQ` primitive is used by the MGS user to associate a termination point with a session context. The primitive only affects a single termination point within a single session. The primitive may also be used to create a session context.
- `MG_JOIN_CON`: The `MG_JOIN_CON` primitive is issued by the MGS provider to confirm that a termination point has been associated with a session context. The primitive only confirms the association of a single termination point within a session. The primitive may also be used to confirm the creation of a session.

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

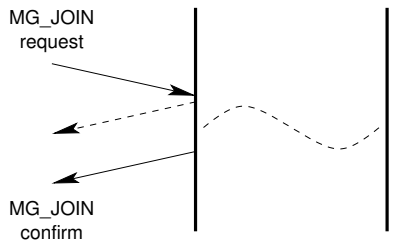

Figure 3.12: *Message Flow: Successful Join Service by MGS Provider*

The lifespan of a termination point within a session context can be long-lived or short-lived depending on the needs of the MGS user. For example, a *Media Gateway* can establish all of the session contexts that it might need and enjoin termination points into the session contexts in advance of any need for communication amoung the termination points within the sessions. This permits a pre-assigned association of termination points to session contexts.

3.2.3 Enable Service

The enable service provides the ability for the MGS user to reserve resources or set parameters associated with a termination point. It is equivalent in part to the `Add` service for a single termination point in *ITU-T Recommendation H.248*.

The MGS user can choose the point at which termination points are to be enabled. An enabled termination point is not necessarily connected into a session context. Where the nature of the termination point does not require a procedure to be enabled, the Section 3.2.4 [Connection Service], page 23 can be used to both enable the termination point and establish communications within the session in a single operation.

Note that a termination point does not need to be joined to a session context to be enabled.

- `MG_ENABLE_REQ`: The `MG_ENABLE_REQ` primitive is used by the MGS user to request that resources be allocated or parameters be set for a termination point. This may include, for example, the activation of an RTP session, or the seizure of a TDM trunk. Parameters that might be included when enabling a termination point might include, for an RTP stream, whether RTCP is used, the maximum jitter buffer size, the codec, and the IP address and port number of the remote end of the RTP stream.
- `MG_ENABLE_CON`: The `MG_ENABLE_CON` primitive is issued by the MGS provider to confirm that resources have been allocated and parameters negotiated for a termination point. This may include, for example, the activation of an RTP session, or the seizure of a TDM trunk.

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

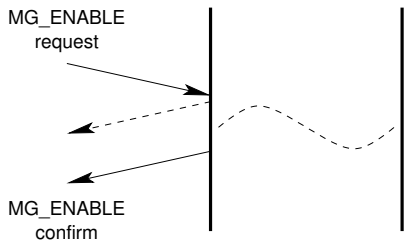

Figure 3.13: *Message Flow: Successful Enable Service by MGS Provider*

3.2.4 Connection Service

The connection service provides the ability for the MGS user to specify that a termination point be connected within a session context. Termination points can be connected for transmission (samples sent to the termination point) or for reception (samples received from the termination point) or both. Connecting a termination point within a session context causes transmission to be taken from the session context and receiption to be provided to the session context.

- `MG_CONN_REQ`: The `MG_CONN_REQ` message is used by the MGS user to request that the termination point be connected to the session context. Connection to the context might require some switching or other mechanism to prepare the termination point for data transmission and reception. Connections can be formed for the receive direction or the transmit direction independently.
- `MG_CONN_CON`: The `MG_CONN_CON` message is used by the MGS provider to confirm that the termination point has been connected to the session context. Connection to the session context may have required some switching or other mechanism to prepare the termination point for data transmission and receptoin. Connection can be confirmed for the receive or transmit directions independently.

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

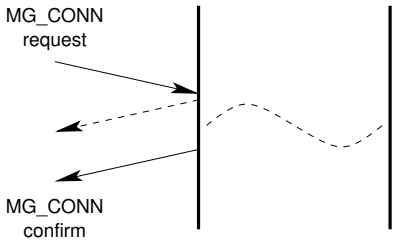

Figure 3.14: *Message Flow: Successful Connection Service*

3.2.5 Action Service

The action service provides the ability for the MGS user to request that an action be performed on a termination point within a session, or on an entire session. It allows the MGS user to provide tones, anouncements, test terminations, etc.

- `MG_ACTION_REQ`: The `MG_ACTION_REQ` message is used by the MGS user to request that an action be performed on a session or on a termination point within a session. The action can be an audio pattern provided by the MGS user in associated data blocks, or can be a pre-established pattern specified by the MGS user. The MGS user has control of the session and termination point within the session to which the action is applied, the duration or repetition of the action.

- `MG_ACTION_IND`: The `MG_ACTION_IND` message is used by the MGS provider to indicate to the MGS user that a requested action has been initiated or has passed a repetition point.

- `MG_ACTION_CON`: The `MG_ACTION_CON` message is used by the MGS provider to confirm to the MGS user that an action of restricted duration or repetitions has reached the endi of its duration or repetitions.

- `MG_ABORT_REQ`: The `MG_ABORT_REQ` message is used by an MGS user to abort an ongoing actions that was previously initiated with an `MG_ACTION_REQ` message, but has not yet terminated (i.e. with an `MG_ACTION_CON`).

A successful invocation of the action service is illustrated in Figure 3.15.

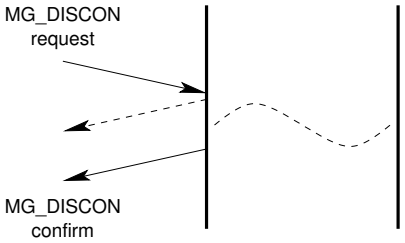

Figure 3.15: *Message Flow: Successful Action Service by MGS Provider*

An aborted invocation of the action service is illustrated in Figure 3.16.

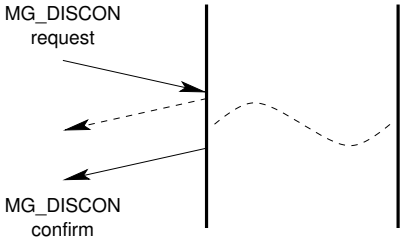

Figure 3.16: *Message Flow: Aborted Action Service by MGS Provider*

3.2.6 Data Transfer Service

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

- `MG_DATA_REQ`: The `MG_DATA_REQ` message is used by the MGS user to place raw bits onto the medium. The Stream must have first been successfully activated in the transmit direction using the `MG_CONN_REQ` message.

- `MG_DATA_IND`: The `MG_DATA_IND` message is issued by the MGS provider when activated for the receive direction with the `MG_CONN_REQ` message, to indicate bits received on the medium.

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

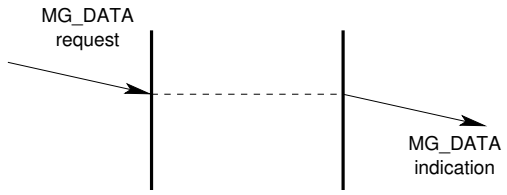

Figure 3.17: *Message Flow: Successful Data Transfer Service*

3.2.7 Notify Service

The notify service provides the MGS user with the ability to request notification of specific events or detected conditions. These notifications can be on a specific termination point, within a session context, or on a global basis.

- `MG_NOTIFY_REQ`: The `MG_NOTIFY_REQ` message is used by the MGS user to request that the MGS provider issue notification indications for specific events or detected conditions with the *Media Gateway*.
- `MG_NOTIFY_IND`: The `MG_NOTIFY_IND` message is issued by the MGS provider when a detected event requested by the MGS user has occured.

This service is roughly equivalent to the `Audit` service of *ITU-T Recommendation H.248.1*.

3.2.8 Disconnection Service

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

- `MG_DISCON_REQ`: The `MG_DISCON_REQ` message is used by the MGS user to request that the Stream be disconnected from the medium. Disconnection from the medium might require some switching or other mechanism. Disconnection can be performed for the receive direction or the transmit direction independently.
- `MG_DISCON_CON`: The `MG_DISCON_CON` message is used by the MGS provider to confirm that the Stream has been disconnected from the medium. Disconnect from the medium might require some switching or other mechanism. Disconnection can be confirmed for the receive or transmit directions independently.
- `MG_DISCON_IND`: The `MG_DISCON_IND` message is used by the MGS provider to indicate to the MGS user that the termination point has been disconnected from the session context. Disconnection is indicated for both the receive and transmit directions. Disconnection indications can result from such things as loss of an RTP stream, on-hook condition, trunk release, or hardware-failure oriented circuit blocking. This primitive is roughly equivalent to the `ServiceChange` service of *ITU-T Recommendation H.248* for a specific termination.

A successful invocation of the disconnection service by the MGS user is illustrated in Figure 3.18.

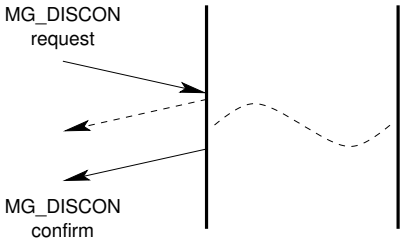

Figure 3.18: *Message Flow: Successful Disconnection Service by MGS User*

A successful invocation of the disconnection service by the MGS provider is illustrated in Figure 3.19.

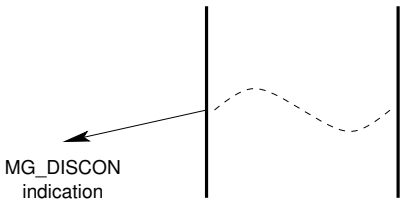

Figure 3.19: *Message Flow: Successful Disconnection Service by MGS Provider*

3.2.9 Disable Service

The disable service provides the ability for the MGS user to release resources associated with a termination point. It is equivalent in part to the `Subtract` service for a single termination point in *ITU-T Recommendation H.248*.

The MGS user can choose th point at which termination points are to be enabled and disabled. An enabled or disable termination point is not necessarily connected into a session context. Where the nature of the termination point does not require a procedure to be disabled, the Section 3.2.8 [Disconnection Service], page 26 can be used to both disable the termination point and disconnect communications within the session in a single operation.

Note that a termination point does not need to be joined to a session context to be disabled.

- `MG_DISABLE_REQ`: The `MG_DISABLE_REQ` primitive is used by the MGS user to request that resources be released for a termination point. This may include, for example, the deactivation of an RTP session, or the release of a TDM trunk.
- `MG_DISABLE_CON`: The `MG_DISABLE_CON` primitive is issued by the MGS provider to confirm that resources have been released for a termination point. This may include, for example, the release of an RTP stream, or the release of a TDM trunk.

A successful invocation of the disable service is illustrated in Figure 3.20. The disable service is an acknowledged and confirmed service that requries the immediate acknowledgement or refusal of the requires penultimately followed by a confirmation on success.

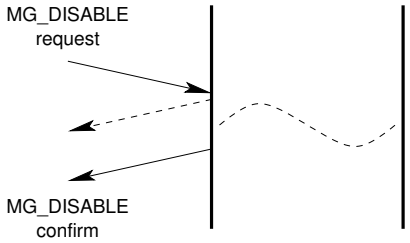

Figure 3.20: *Message Flow: Successful Disable Service by MGS Provider*

3.2.10 Leave Service

The leave service provides the ability for the MGS user to disassociate a termination point with a session context. It is roughly equivalent to the `Subtract` service for a single termination point in *ITU-T Recommendation H.248*. The leave service may also be used to disconnect a termination point in one operation.

- `MG_LEAVE_REQ`: The `MG_LEAVE_REQ` service provides the ability for the MGS user to disassociate a termination point from a session context. The primitive affects one or more termination points within a given session context. The primitive may also be used to disconnect termination points from the session context as they are being disassociated.
- `MG_LEAVE_CON`: The `MG_LEAVE_CON` primitive is issued by the MGS provider to confirm that a termination point, or a number of termination points, have been disconnected from and disassociated with a session context. The primitive confirms one or more termination point disconnects and removals from the session.

A successful invocation of the leave service is illustrated in Figure 3.21. The leave service is an acknowleged and confirmed service that requires the immediate acknowledgement or refusal of the request penultimately followed by a confirmation on success.

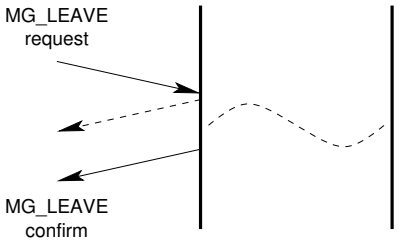

Figure 3.21: *Message Flow: Successful Leave Service by MGS Provider*

The lifespan of termination points within a session context can be long-lived or short-lived depending on the needs of the MGS user. For example, a *Media Gateway* can establish all of the session contexts that it might need in advance and enjoin termination points into the session contexts in advance of any need for communiction among the termination points within the sessions. This permits pre-

assigned association of termination points to session contexts. In this case, the leave service is only required when tearing down an entire *Media Gateway*.

3.2.11 Destroy Service

The destroy service allows an MGS user to request the destruction of a session context. Destruction of a session context may result in the disconnection and disassociation of all termination points that are currently connected or enjoined in the session context.

- `MG_DESTROY_REQ`: The `MG_DESTROY_REQ` primitive is used by an MGS user to request that a session context be deleted.
- `MG_DESTROY_ACK`: The `MG_DESTROY_ACK` primitive is issued by the MGS provider to acknowledge the destruction of a session context. The session context identfier is available again to be used for the creation of a session context.

Session contexts can be long-lived or short-lived depending on the needs of the MGS user. For example, it is possible for the *Media Gateway* to allocated all of the session contexts that it might need to perform its functions in advance of the need for communications within any given session context. In this case, the destroy service might only be required when tearing down an entire *Media Gateway*.

4 MGI Service Primitives

4.1 Local Management Service Primitives

These service primitives implement the local management services (see Section 3.1 [Local Management Services], page 15).

4.1.1 Acknowledgement Service Primitives

These service primitives implement the acknowledgement service (see Section 3.1.1 [Acknowledgement Service], page 15).

4.1.1.1 MG_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 MG_ok_ack {
    mg_ulong mg_primitive;
    mg_ulong mg_correct_prim;
    mg_ulong mg_state;
} MG_ok_ack_t;
```

Parameters

The service primitive contains the following parameters:

mg_primitive
: Indicates the service primitive type. Always `MG_OK_ACK`.

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

`MG_ATTACH_REQ`	Attach request.
`MG_ENABLE_REQ`	Enable request.
`MG_CONN_REQ`	Connect request.
`MG_DISCON_REQ`	Disconnect request.
`MG_DISABLE_REQ`	Disable request.
`MG_DETACH_REQ`	Detach Request.

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

`MGS_UNINIT`	Unitialized.
`MGS_UNUSABLE`	Device cannot be used, Stream in hung state.
`MGS_DETACHED`	No PPA attached, awaiting `MG_ATTACH_REQ`.

MGS_ATTACHED	PPA attached, awaiting MG_ENABLE_REQ.
MGS_WCON_EREQ	Waiting to send MG_ENABLE_CON.
MGS_WCON_RREQ	Waiting to send MG_DISABLE_CON.
MGS_ENABLED	Ready for use, awaiting primitive exchange.
MGS_WCON_CREQ	Waiting to send MG_CONN_CON.
MGS_WCON_DREQ	Waiting to send MG_DISCON_CON.
MGS_CONNECTED	Connected, active data transfer.

State

This primitive is issued by the MGS provider in the MGS_WACK_AREQ, MGS_WACK_UREQ, MGS_WACK_CREQ or MGS_WACK_DREQ state.

New State

The new state is MGS_DETACHED, MGS_ATTACHED, MGS_ENABLED or MGS_CONNECTED, depending on the primitive to which the message is responding.

4.1.1.2 MG_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 MG_error_ack {
    mg_ulong mg_primtive;
    mg_ulong mg_error_primitive;
    mg_ulong mg_error_type;
    mg_ulong mg_unix_error;
    mg_ulong mg_state;
} MG_error_ack_t;
```

Parameters

The error acknowledgement primitive contains the following parameters:

mg_primitive

Indicates the primitive type. Always `MG_ERROR_ACK`.

mg_error_type

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

`[MGSYSERR]`	UNIX system error.
`[MGBADADDR]`	Bad address format or content.
`[MGOUTSTATE]`	Interface out of state.
`[MGBADOPT]`	Bad options format or content.
`[MGBADPARM]`	Bad parameter format or content.
`[MGBADPARMTYPE]`	Bad paramater structure type.
`[MGBADFLAG]`	Bad flag.
`[MGBADPRIM]`	Bad primitive.
`[MGNOTSUPP]`	Primitive not supported.
`[MGBADSLOT]`	Bad multplex slot.

mg_unix_error

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

mg_error_primitive

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

`MG_INFO_REQ`	Information request.
`MG_OPTMGMT_REQ`	Options management request.
`MG_ATTACH_REQ`	Attach request.
`MG_ENABLE_REQ`	Enable request.
`MG_CONN_REQ`	Connect request.
`MG_DATA_REQ`	Data request.
`MG_DISCON_REQ`	Disconnect request.
`MG_DISABLE_REQ`	Disable request.

MG_DETACH_REQ	Detach Request.
MG_INFO_ACK	Information acknowledgement.
MG_OPTMGMT_ACK	Options Management acknowledgement.
MG_OK_ACK	Successful receipt acknowledgement.
MG_ERROR_ACK	Error acknowledgement.
MG_ENABLE_CON	Enable confirmation.
MG_CONN_CON	Connect confirmation.
MG_DATA_IND	Data indication.
MG_DISCON_IND	Disconnect indication.
MG_DISCON_CON	Disconnect confirmation.
MG_DISABLE_IND	Disable indication.
MG_DISABLE_CON	Disable confirmation.
MG_EVENT_IND	Event indication.

mg_state

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

MGS_UNINIT	Unitialized.
MGS_UNUSABLE	Device cannot be used, Stream in hung state.
MGS_DETACHED	No PPA attached, awaiting MG_ATTACH_REQ.
MGS_WACK_AREQ	Waiting for attach.
MGS_WACK_UREQ	Waiting for detach.
MGS_ATTACHED	PPA attached, awaiting MG_ENABLE_REQ.
MGS_WCON_EREQ	Waiting to send MG_ENABLE_CON.
MGS_WCON_RREQ	Waiting to send MG_DISABLE_CON.
MGS_ENABLED	Ready for use, awaiting primitive exchange.
MGS_WACK_CREQ	Waiting acknowledgement of MG_CONN_REQ.
MGS_WCON_CREQ	Waiting to send MG_CONN_CON.
MGS_WACK_DREQ	Waiting acknowledgement of MG_DISCON_REQ.
MGS_WCON_DREQ	Waiting to send MG_DISCON_CON.
MGS_CONNECTED	Connected, active data transfer.

State

This primitive can be issued in any state for which a local acknowledgement is not pending. The MGS 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 16).

4.1.2.1 MG_INFO_REQ

Description

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

Format

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

```
typedef struct MG_info_req {
    mg_ulong mg_primitive;
} MG_info_req_t;
```

Parameters

This primitive contains the following parameters:

mg_primitive
: Specifies the primitive type. Always `MG_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 MGS provider to acknowledge receipt of the primitive as follows:

- **Successful**: The MGS provider is required to acknowledge receipt of the primitive and provide the requested information using the `MG_INFO_ACK` primitive.
- **Unsuccessful (non-fatal errors)**: The MGS provider is required to negatively acknowledge the primtive using the `MG_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:

`[MGSYSERR]`	UNIX system error.
`[MGBADADDR]`	Bad address format or content.
`[MGOUTSTATE]`	Interface out of state.
`[MGBADOPT]`	Bad options format or content.
`[MGBADPARM]`	Bad parameter format or content.
`[MGBADPARMTYPE]`	Bad paramater structure type.
`[MGBADFLAG]`	Bad flag.
`[MGBADPRIM]`	Bad primitive.
`[MGNOTSUPP]`	Primitive not supported.
`[MGBADSLOT]`	Bad multplex slot.

4.1.2.2 MG_INFO_ACK

Description

This MGS provider originated primitive acknowledges receipt and successful processing of the MG_INFO_REQ primitive and provides the requested information concerning the MGS provider.

Format

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

```
typedef struct MG_info_ack {
    mg_ulong mg_primitive;    /* always MG_INFO_ACK */
    mg_ulong mg_addr_length;  /* media gateway address length */
    mg_ulong mg_addr_offset;  /* media gateway address offset */
    mg_ulong mg_parm_length;  /* media gateway parameters length */
    mg_ulong mg_parm_offset;  /* media gateway parameters offset */
    mg_ulong mg_prov_flags;   /* provider options flags */
    mg_ulong mg_prov_class;   /* provider class */
    mg_ulong mg_style;        /* provider style */
    mg_ulong mg_version;      /* media gateway interface version */
    mg_ulong mg_state;        /* media gateway state */
} MG_info_ack_t;
```

Parameters

The information acknowledgement service primitive has the following parameters:

mg_primitive
: Indicates the service primitive type. Always MG_INFO_ACK.

mg_addr_length
: Indicates the length of the PPA address to which the provider is attached. When in states MGS_DETACHED or MGS_WACK_AREQ, this value will be zero ('0').

mg_addr_offset
: Indicates the offset, beginning from the start of the M_PCPROTO message block of the PPA address associated with the provider. When the *mg_addr_length* field is zero, this field is also zero ('0').

mg_parm_length
: Indicates the length of the parameters associated with the provider.

mg_parm_offset
: Indicates the offset, beginning from the start of the M_PCPROTO message block, of the parameters associated with the provider. When the *mg_parm_length* field is zero, this field is also zero ('0').

mg_prov_flags
: Indicates the options flags associated with the provider. This is a bitwise OR of zero or more of the following flags:

mg_prov_class
: Indicates the provider class. This can be one of the following values:

  MG_CIRCUIT Circuit provider class.

mg_addr_length
: This is a variable length field. The length of the field is determined by the length attribute.

  For a *Style 2* driver, when *mg_style* is `MG_STYLE2`, and when in an attached state, this field provides the current PPA associated with the Stream; the length is typically 4 bytes.

  For a *Style 1* driver, when *mg_ppa_stype* is `MG_STYLE1`, the length is 0 bytes.

mg_style
: Indicates the PPA style of the MGS provider. This value can be one of the following values;

  `MG_STYLE1` PPA is implicitly attached by `open(2s)`.
  `MG_STYLE2` PPA must be explicitly attached using `MG_ATTACH_REQ`.

mg_version
: The version of the interface. This version is `MG_VERSION_1_1`.

  `MG_VERSION_1_0` Version 1.0 of interface.
  `MG_VERSION_1_1` Version 1.1 of interface.
  `MG_VERSION` Always the current version of the header file.

mg_state
: Indicates the state of the MGS provider at the time that the information acknowledgement service primitive wsa issued. This field can be one of the following values:

  `MGS_UNINIT` Unitialized.
  `MGS_UNUSABLE` Device cannot be used, Stream in hung state.
  `MGS_DETACHED` No PPA attached, awaiting `MG_ATTACH_REQ`.
  `MGS_WACK_AREQ` Waiting for attach.
  `MGS_WACK_UREQ` Waiting for detach.
  `MGS_ATTACHED` PPA attached, awaiting `MG_ENABLE_REQ`.
  `MGS_WCON_EREQ` Waiting to send `MG_ENABLE_CON`.
  `MGS_WCON_RREQ` Waiting to send `MG_DISABLE_CON`.
  `MGS_ENABLED` Ready for use, awaiting primitive exchange.
  `MGS_WACK_CREQ` Waiting acknowledgement of `MG_CONN_REQ`.
  `MGS_WCON_CREQ` Waiting to send `MG_CONN_CON`.
  `MGS_WACK_DREQ` Waiting acknowledgement of `MG_DISCON_REQ`.
  `MGS_WCON_DREQ` Waiting to send `MG_DISCON_CON`.
  `MGS_CONNECTED` Connected, active data transfer.

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 16).

4.1.3.1 MG_ATTACH_REQ

Description

This MGS user originated primitive requests that the Stream upon which the primitive is issued be associated with the specified Physical Point of Attachment (PPA). This primitive is only applicable to *Style 2* MGS provider Streams, that is, Streams that return `MG_STYLE2` in the *mg_style* field of the `MG_INFO_ACK`.

Format

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

```
typedef MG_attach_req {
    mg_ulong mg_primitive;
    mg_ulong mg_addr_length;
    mg_ulong mg_addr_offset;
    mg_ulong mg_flags;
} MG_attach_req_t;
```

Parameters

The attach request primitive contains the following parameters:

mg_primitive
: Specifies the service primitive type. Always `MG_ATTACH_REQ`.

mg_addr_length
: Specifies the Physical Point of Attachment (PPA) to which to associate the *Style 2* Stream. This is a variable length identifier whose length is determined by the *mg_addr_length* value. Specifies the length of the Physical Point of Attachment (PPA) address. The form of the PPA address is provider-specific.

mg_addr_offset
: Specifies the offset, from the beginning of the `M_PROTO` message block, of the start of the Physical Point of Attachment (PPA) address.

mg_flags
: Specifies the options flags for attachment. Options flags are provider-specific.

State

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

New State

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

Response

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

– **Successful**: The MGS provider acknowledges receipt of the primitive and successful outcome of the attach service with a `MG_OK_ACK` primitive. The new state is `MGS_ATTACHED`.

– **Unsuccessful (non-fatal errors)**: The MGS provider acknowledges receipt of the primitive and failure of the attach service with a `MG_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:

`[MGSYSERR]`	UNIX system error.
`[MGBADADDR]`	Bad address format or content.
`[MGOUTSTATE]`	Interface out of state.
`[MGBADOPT]`	Bad options format or content.
`[MGBADPARM]`	Bad parameter format or content.
`[MGBADPARMTYPE]`	Bad paramater structure type.
`[MGBADFLAG]`	Bad flag.
`[MGBADPRIM]`	Bad primitive.
`[MGNOTSUPP]`	Primitive not supported.
`[MGBADSLOT]`	Bad multplex slot.

4.1.3.2 MG_DETACH_REQ

Description

This MGS user originated primitive requests that the Stream upon which the primitive is issued be disassociated from the Physical Point of Appearance (PPA) to which it is currently attached. This primitive is only applicable to *Style 2* MGS provider Streams, that is, Streams that return `MG_STYLE2` in the *mg_style* field of the `MG_INFO_ACK`.

Format

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

```
typedef struct MG_detach_req {
    mg_ulong mg_primitive;
} MG_detach_req_t;
```

Parameters

The detach request service primitive contains the following parameters:

mg_primitive
: Specifies the service primitive type. Always `MG_DETACH_REQ`.

State

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

New State

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

Response

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

- **Successful**: The MGS provider acknowledges receipt of the primitive and successful outcome of the detach service with a `MG_OK_ACK` primitive. The new state is `MGS_DETACHED`.
- **Unsuccessful (non-fatal errors)**: The MGS provider acknowledges receipt of the primitive and failure of the detach service with a `MG_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:

`[MGSYSERR]`	UNIX system error.
`[MGBADADDR]`	Bad address format or content.
`[MGOUTSTATE]`	Interface out of state.
`[MGBADOPT]`	Bad options format or content.
`[MGBADPARM]`	Bad parameter format or content.
`[MGBADPARMTYPE]`	Bad paramater structure type.
`[MGBADFLAG]`	Bad flag.
`[MGBADPRIM]`	Bad primitive.
`[MGNOTSUPP]`	Primitive not supported.
`[MGBADSLOT]`	Bad multplex slot.

4.1.4 Initialization Service Primitives

Initialization service primitives allow the MGS 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 multiplex identifier as defined in G.703, it may be necessary to perform switching to connect or disconnect the circuit identification code associated with the multiplex identifier.

These service primitives implement the initialization service (see Section 3.1.4 [Initialization Service], page 18).

4.1.4.1 MG_ENABLE_REQ

Description

This MGS user originated primitive requests that the MGS 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 MG_enable_req {
    mg_ulong mg_primitive;
    mg_ulong mg_addr_length;
    mg_ulong mg_addr_offset;
    mg_ulong mg_flags;
} MG_enable_req_t;
```

Parameters

The enable request service primitive contains the following parameters:

mg_primitive
: Specifies the service primitive type. Always `MG_ENABLE_REQ`.

mg_addr_length
: 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 value of this field. This remote address could be a circuit identification code, an IP address, or some other form of circuit or multiplex identifier.

mg_addr_offset
: Specifies the offset, from the beginning of the `M_PROTO` message block, of the start of the remote address.

mg_flags
: Specifies the options flags associated with the enable request. Options flags are provider-specific.

State

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

New State

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

Response

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

- **Successful**: When successful, the MGS provider acknowledges successful completion of the enable service with a `MG_ENABLE_CON` primitive. The new state is `MGS_ENABLED`.
- **Unsuccessful (non-fatal errors)**: When unsuccessful, the MGS provider acknowledges the failure of the enable service with a `MG_ERROR_ACK` primitive containing the error. The new state remains unchanged.

Reasons for Failure

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

`[MGSYSERR]`	UNIX system error.
`[MGBADADDR]`	Bad address format or content.
`[MGOUTSTATE]`	Interface out of state.
`[MGBADOPT]`	Bad options format or content.
`[MGBADPARM]`	Bad parameter format or content.
`[MGBADPARMTYPE]`	Bad paramater structure type.
`[MGBADFLAG]`	Bad flag.
`[MGBADPRIM]`	Bad primitive.
`[MGNOTSUPP]`	Primitive not supported.
`[MGBADSLOT]`	Bad multplex slot.

4.1.4.2 MG_ENABLE_CON

Description

This MGS provider originated primitive is issued by the MGS 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 MG_enable_con {
    mg_ulong mg_primitive;
    mg_ulong mg_addr_length;
    mg_ulong mg_addr_offset;
    mg_ulong mg_flags;
} MG_enable_con_t;
```

Parameters

The enable confirmation service primitive contains the following parameters:

mg_primitive
: Indicates the service primitive type. Always `MG_ENABLE_CON`.

mg_addr_length
: Confirms the length of the remote address to which the enable is confirmed.

mg_addr_offset
: Confirms the offset, from the beginning of the `M_PROTO` message block, of the start of the remote address.

mg_flags
: Confirms the options flags associated with the enable confirmation. Options flags are provider-specific.

State

This primitive is issued by the MGS provider in the `MGS_WCON_EREQ` state.

New State

The new state is `MGS_ENABLED`.

4.1.4.3 MG_DISABLE_REQ

Description

This MGS user originated primitive requests that the MGS 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 MG_disable_req {
    mg_ulong mg_primitive;
} MG_disable_req_t;
```

Parameters

The disable request service primitive contains the following parameters:

mg_primitive
: Specifies the service primitive type. Always `MG_DISABLE_REQ`.

State

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

New State

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

Response

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

- **Successful**: When successful, the MGS provider acknowledges successful completion of the disable service with an `MG_DISABLE_CON` primitive. The new state is `MGS_ATTACHED`.
- **Unsuccessful (non-fatal errors)**: When unsuccessful, the MGS provider acknowledges the failure of the disable service with a `MG_ERROR_ACK` primitive containing the error. The new state remains unchanged.

Reasons for Failure

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

`[MGSYSERR]`	UNIX system error.
`[MGBADADDR]`	Bad address format or content.
`[MGOUTSTATE]`	Interface out of state.
`[MGBADOPT]`	Bad options format or content.
`[MGBADPARM]`	Bad parameter format or content.
`[MGBADPARMTYPE]`	Bad paramater structure type.
`[MGBADFLAG]`	Bad flag.
`[MGBADPRIM]`	Bad primitive.
`[MGNOTSUPP]`	Primitive not supported.
`[MGBADSLOT]`	Bad multplex slot.

4.1.4.4 MG_DISABLE_CON

Description

This MGS provider originated primitive is issued by the MGS 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 MG_disable_con {
    mg_ulong mg_primitive;
} MG_disable_con_t;
```

Parameters

The disable confirmation service primitive contains the following parameters:

mg_primitive
: Indicates the service primitive type. Always `MG_DISABLE_CON`.

State

This primitive is issued by the MGS provider in the `MGS_WCON_RREQ` state.

New State

The new state is `MGS_ATTACHED`.

4.1.4.5 MG_DISABLE_IND

Description

This MGS provider originated primitive is issued by the MGS provider, if an autonomous event results in the disabling of the MGS use Stream without an explicity MGS user request.

Format

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

```
typedef struct MG_disable_ind {
    mg_ulong mg_primitive;
    mg_ulong mg_cause;
} MG_disable_ind_t;
```

Parameters

mg_primitive
: Indicates the service primitive type. Always `MG_DISABLE_IND`.

mg_cause
: Indicates the cause of the autonomous disabling of the MGS user Stream.

State

This primitive will only be issued by the MGS provider in the `MGS_ENABLED` state.

New State

The new state is `MGS_ATTACHED`.

4.1.5 Options Management Service Primitives

The options management service primitives allow the MGS user to negotiate options with the MGS 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 19).

4.1.5.1 MG_OPTMGMT_REQ

Description

This MGS user originated primitive requests that MGS 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 MG_optmgmt_req {
    mg_ulong mg_primitive;
    mg_ulong mg_opt_length;
    mg_ulong mg_opt_offset;
    mg_ulong mg_mgmt_flags;
} MG_optmgmt_req_t;
```

Parameters

The option management request service primitive contains the following parameters:

mg_primitive
: Specifies the service primitive type. Always `MG_OPTMGMT_REQ`.

mg_opt_length
: Specifies the length of the options.

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

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

  `MG_NEGOTIATE`
  : Negotiate the specified value of each specified option and return the negotiated value.

  `MG_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 MGS provider.

  `MG_DEFAULT`
  : Return the default value for the specified options (or all options). Do not alter the current value assumed by the MGS provider.

`MG_CURRENT`
: Return the current value for the specified options (or all options). Do not alter the current value assumed by the MGS 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 MGS provider to acknowledge receipt of the primitive as follows:

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

Reasons for Failure

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

`[MGSYSERR]`	UNIX system error.
`[MGBADADDR]`	Bad address format or content.
`[MGOUTSTATE]`	Interface out of state.
`[MGBADOPT]`	Bad options format or content.
`[MGBADPARM]`	Bad parameter format or content.
`[MGBADPARMTYPE]`	Bad paramater structure type.
`[MGBADFLAG]`	Bad flag.
`[MGBADPRIM]`	Bad primitive.
`[MGNOTSUPP]`	Primitive not supported.
`[MGBADSLOT]`	Bad multplex slot.

4.1.5.2 MG_OPTMGMT_ACK

Description

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

Format

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

```
typedef struct MG_optmgmt_ack {
    mg_ulong mg_primitive;
    mg_ulong mg_opt_length;
    mg_ulong mg_opt_offset;
    mg_ulong mg_mgmt_flags;
} MG_optmgmt_ack_t;
```

Parameters

The option management acknowledgement service primitive contains the following parameters:

mg_primitive
: Indicates the service primitive type. Always `MG_OPTMGMT_ACK`.

mg_opt_length
: Indicates the length of the returned options.

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

mg_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:

  `MG_SUCCESS`
  : The MGS provider succeeded in negotiating or returning all of the options specified by the MGS user in the `MG_OPTMGMT_REQ` primitive.

  `MG_FAILURE`
  : The MGS provider failed to negotiate one or more of the options specified by the MGS user.

  `MG_PARTSUCCESS`
  : The MGS provider negotiated a value of lower quality for one or more of the options specified by the MGS user.

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

  `MG_NOTSUPPORT`
  : The MGS provider failed to recognize one or more of the options specified by the MGS user.

State

This primitive is issued by the MGS provider in direct response to a `MG_OPTMGMT_REQ` primitive.

New State

The new state remains unchangted.

Rules

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

— When the *mg_mgmt_flags* field in the `MG_OPTMGMT_REQ` primitive is set to `MG_NEGOTIATE`, the MGS provider will attempt to negotiate a value for each of the options specified in the request.

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

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

— When the flags are `MG_CHECK`, the MGS 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 MGS provider to indicate asynchronous errors, events and statistics collection to the MGS user.

These service primitives implement the event reporting service (see Section 3.1.8 [Event Reporting Service], page 21).

4.1.6.1 MG_ERROR_IND

Description

This MGS provider originated service primitive is issued by the MGS 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 MG_error_ind {
    mg_ulong mg_primitive;
    mg_ulong mg_error_type;
    mg_ulong mg_unix_error;
    mg_ulong mg_state;
} MG_error_ind_t;
```

Parameters

The error indication service primitive contains the following parameters:

mg_primitive
: Indicates the service primitive type. Always `MG_ERROR_IND`.

MG_error_type
: Indicates the MGI error number describing the error. This field can have one of the following values:

`[MGSYSERR]`	UNIX system error.
`[MGBADADDR]`	Bad address format or content.
`[MGOUTSTATE]`	Interface out of state.
`[MGBADOPT]`	Bad options format or content.
`[MGBADPARM]`	Bad parameter format or content.
`[MGBADPARMTYPE]`	Bad paramater structure type.
`[MGBADFLAG]`	Bad flag.
`[MGBADPRIM]`	Bad primitive.
`[MGNOTSUPP]`	Primitive not supported.
`[MGBADSLOT]`	Bad multplex slot.

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

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

MGS_UNINIT	Unitialized.
MGS_UNUSABLE	Device cannot be used, Stream in hung state.
MGS_DETACHED	No PPA attached, awaiting MG_ATTACH_REQ.
MGS_WACK_AREQ	Waiting for attach.
MGS_WACK_UREQ	Waiting for detach.
MGS_ATTACHED	PPA attached, awaiting MG_ENABLE_REQ.
MGS_WCON_EREQ	Waiting to send MG_ENABLE_CON.
MGS_WCON_RREQ	Waiting to send MG_DISABLE_CON.
MGS_ENABLED	Ready for use, awaiting primitive exchange.
MGS_WACK_CREQ	Waiting acknowledgement of MG_CONN_REQ.
MGS_WCON_CREQ	Waiting to send MG_CONN_CON.
MGS_WACK_DREQ	Waiting acknowledgement of MG_DISCON_REQ.
MGS_WCON_DREQ	Waiting to send MG_DISCON_CON.
MGS_CONNECTED	Connected, active data transfer.

State

This primitive can be issued in any state for which a local acknowledgement is not pending. The MGS 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 MG_STATS_IND

Description

This MGS provider originated primitive is issued by the MGS 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 MG_stats_ind {
    mg_ulong mg_primitive;
    mg_ulong mg_interval;
    mg_ulong mg_timestamp;
} MG_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:

mg_primitive
: Indicates the service primitive type. Always `MG_STATS_IND`.

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

mg_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 MGS provider in any state in which a local acknowledgement is not pending.

New State

The new state remains unchanged.

4.1.6.3 MG_EVENT_IND

Description

This MGS provider originated primitive is issued by the MGS 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 MG_event_ind {
    mg_ulong mg_primitive;
    mg_ulong mg_event;
    mg_ulong mg_slot;
} MG_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:

mg_primitive
: Indicates the service primitive type. Always `MG_EVENT_IND`.

mg_event
: Indicates the provider-specific event that has occured.

`MGF_EVT_DCD_ASSERT`	Data carrier detect lead asserted.
`MGF_EVT_DCD_DEASSERT`	Data carrier detect lead deasserted.
`MGF_EVT_DSR_ASSERT`	Data set ready lead asserted.
`MGF_EVT_DSR_DEASSERT`	Data set ready lead deasserted.
`MGF_EVT_DTR_ASSERT`	Data terminal ready lead asserted.
`MGF_EVT_DTR_DEASSERT`	Data terminal ready lead deasserted.
`MGF_EVT_RTS_ASSERT`	Request to send lead asserted.
`MGF_EVT_RTS_DEASSERT`	Request to send lead deasserted.
`MGF_EVT_CTS_ASSERT`	Clear to send lead asserted.
`MGF_EVT_CTS_DEASSERT`	Clear to send lead deasserted.
`MGF_EVT_RI_ASSERT`	Ring indicator asserted.
`MGF_EVT_RI_DEASSERT`	Ring indicator deasserted.
`MGF_EVT_YEL_ALARM`	Yellow alarm condition.
`MGF_EVT_BLU_ALARM`	Blue alarm condition.
`MGF_EVT_RED_ALARM`	Red alarm condition.
`MGF_EVT_NO_ALARM`	Alarm recovery condition.

mg_slot
: Where the PPA is associated with a multiplexed medium, this parameter indicates the slots within the mutliplexed media to which the event corresponds. The form of the slot specification is provider- and media-specific. See also [Multiplex Media], page 13.

  Where the PPA specifies a single channel for a medium, this parameter is set to zero (‘`0`’) by the MGS provider on MGS provider originated primitives and is ignored by the MGS provider on MGS user originated primitives.

State

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

New State

The new state remains unchanged.

4.2 Protocol Service Primitives

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

These service primitives implement the protocol services (see Section 3.2 [Protocol Services], page 21).

4.2.1 Connection Service Primitives

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

These service primitives implement the connection service (see Section 3.2.4 [Connection Service], page 23).

4.2.1.1 MG_CONN_REQ

Description

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

Format

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

```
typedef struct MG_connect_req {
    mg_ulong mg_primitive;
    mg_ulong mg_conn_flags;
    mg_ulong mg_slot;
} MG_connect_req_t;
```

Parameters

The connect request service primitive contains the following parameters:

mg_primitive
: Specifies the service primitive type. Always `MG_CONN_REQ`.

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

  `MGF_RX_DIR` — Specifies that the MGS user Stream is to be connected to the medium in the receive direction.

  `MGF_TX_DIR` — Specifies that the MGS user Stream is to be connected to the medium in the transmit direction.

  `MGF_MONITOR` — Specifies that the MGS user Stream is to be connected to the medium in monitoring (tap) mode.

mg_slot
: Where the PPA is associated with a multiplexed medium, this parameter specifies the slots within the mutliplexed media to be connected to the MGS User Stream. The form of the slot specification is provider- and media-specific. See also [Multiplex Media], page 13.

Where the PPA specifies a single channel for a medium, this parameter is set to zero (‘0’) by the MGS provider on MGS provider originated primitives and is ignored by the MGS provider on MGS user originated primitives.

State

This service primitive is only valid in the MGS_ENABLED state.

New State

The new state is the MGS_WACK_CREQ state.

Response

The connect request service primitive requires that the MGS provider acknowledge receipt of the primitive as follows:

- **Successful**: When successful, the MGS provider acknowledges successful completion of the connect service with a MG_OK_ACK primitive. The new state is MGS_WCON_CREQ. When the MGS provider eventually completes the connection, it confirms the connection with a MG_CONN_CON primitive and the new state is then MGS_CONNECTED.
- **Unsuccessful (non-fatal errors)**: When unsuccessful, the MGS provider acknowledges the failure of the connect service with a MG_ERROR_ACK primitive containing the error. The new state remains unchanged.

Reasons for Failure

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

[MGSYSERR]	UNIX system error.
[MGBADADDR]	Bad address format or content.
[MGOUTSTATE]	Interface out of state.
[MGBADOPT]	Bad options format or content.
[MGBADPARM]	Bad parameter format or content.
[MGBADPARMTYPE]	Bad paramater structure type.
[MGBADFLAG]	Bad flag.
[MGBADPRIM]	Bad primitive.
[MGNOTSUPP]	Primitive not supported.
[MGBADSLOT]	Bad multplex slot.

4.2.1.2 MG_CONN_CON

Description

This MGS provider originated service primitive allows the MGS provider to confirm the succesful completion of the connect servivce with the connection of the user Stream to the medium in the transmit, receive, or both, directions.

Format

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

```
typedef struct MG_connect_con {
    mg_ulong mg_primitive;
    mg_ulong mg_conn_flags;
    mg_ulong mg_slot;
} MG_connect_con_t;
```

Parameters

mg_primitive
: Indicates the service primitive type. Always `MG_CONN_CON`.

mg_conn_flags
: Indicates the connect flags. This field is a bitwise OR of zero or more of the following flags:

  `MGF_RX_DIR` Confirms that the MGS user Stream was connected to the medium in the receive direction.

  `MGF_TX_DIR` Confirms that the MGS user Stream was connected to the medium in the transmit direction.

  `MGF_MONITOR` Confirms that the MGS user Stream was connected to the medium in monitoring (tap) mode.

mg_slot
: Where the PPA is associated with a multiplexed medium, this parameter specifies the slots within the mutliplexed media that are confirmed connected to the MGS user Stream. The form of the slot specification is provider- and media-specific. See also [Multiplex Media], page 13.

  Where the PPA specifies a single channel for a medium, this parameter is set to zero (‘`0`’) by the MGS provider on MGS provider originated primitives and is ignored by the MGS provider on MGS user originated primitives.

State

This primitive will only be issued by the MGS provider in the `MGS_WCON_CREQ` state.

New State

The new state of the interface is the `MGS_CONNECTED` state.

4.2.2 Data Transfer Service Primitives

The data transfer service primitives permit the MGS user to pass bits for transmission to the MGS provider and accept received bits from the MGS provider.
These service primitives implement the data transfer service (see Section 3.2.6 [Data Transfer Service], page 25).

4.2.2.1 MG_DATA_REQ

Description

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

Format

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

```
typedef struct MG_data_req {
    mg_ulong mg_primitive;
    mg_ulong mg_slot;
} MG_data_req_t;
```

Parameters

The transmission request service primitive contains the following parameters:

mg_primitive
: Specifies the service primitive type. Always `MG_DATA_REQ`.

mg_slot
: Where the PPA is associated with a multiplexed medium, this parameter specifies the slots within the mutliplexed media upon which the user data is to be transmitted. The form of the slot specification is provider- and media-specific. See also [Multiplex Media], page 13.

  Where the PPA specifies a single channel for a medium, this parameter is set to zero (‘`0`’) by the MGS provider on MGS provider originated primitives and is ignored by the MGS provider on MGS user originated primitives.

State

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

New State

The state remains unchanged.

Response

Reasons for Failure

4.2.2.2 MG_DATA_IND

Description

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

Format

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

```
typedef struct MG_data_ind {
    mg_ulong mg_primitive;
    mg_ulong mg_slot;
} MG_data_ind_t;
```

Parameters

The receive indication service primitive contains the following parameters:

mg_primitive
: Indicates the service primitive type. Always `MG_DATA_IND`.

mg_slot
: Where the PPA corresponds to a multiplexed media, this parameter specifies to which of the media streams the data indicated corresponds. The form of the slot specification is provider- and media-specific. See also [Multiplex Media], page 13.

  Where the PPA specifies a single channel for a medium, this parameter is set to zero (‘`0`’) by the MGS provider on MGS provider originated primitives and is ignored by the MGS provider on MGS user originated primitives.

State

This primitive is only issued by the MGS provider in the `MGS_CONNECTED` state.

New State

The state remains unchanged.

Response

Reasons for Failure

4.2.3 Action Service Primitives

The action service primitives permit the MGS user to initiate and control and action that is to be performed on a session or on a termination point within a session.
These service primitives implement the action service (see Section 3.2.5 [Action Service], page 24).

4.2.3.1 MG_ACTION_REQ

Description

This MGS user originated service primitive allows the MGS user to initiate an action on an existing session or on an existing termination point within an existing session.

Format

The action request primitive consists of one `M_PROTO` message block, followed by zero or more `M_DATA` blocks containing audio information. The `M_PROTO` message block is formatted as follows:

```
typedef struct MG_action_req {
    mg_ulong mg_primitive;      /* always MG_ACTION_REQ */
    mg_ulong mg_action;         /* requested action */
    mg_ulong mg_se_id;          /* session id */
    mg_ulong mg_tp_id;          /* termination id to perform action */
    mg_ulong mg_duration;       /* duration in milliseconds */
    mg_ulong mg_flags;          /* option flags */
} MG_action_req_t;
```

Parameters

The action request service primitive contains the following parameters:

mg_primitive
: Specifies the service primitive type. Always `MG_ACTION_REQ`.

mg_action
: Specifies the action to perform. For specific values, see 'Flags' below.

mg_se_id
: Specifies the session in which to apply the action. When specified as zero (0), the session specified is that associated with the Stream upon which the `MG_ACTION_REQ` was issued.

mg_tp_id
: Specifies the termination point to which to apply the action. When specified as zero (0), the termination point(s) specified are all connected termination points within the specified session.

mg_duration
: Specifies the duration of the pattern or repetition in milliseconds. When specified as zero (0), the default duration for the pattern or action will be used.

mg_flags
: Specifies the flags associated with the action. The flags ar described below under 'Flags'.

Flags

The *mg_flags* field can assume one of the following values:

`MG_MORE_DATA`
: The data contained in `M_DATA` message blocks associated with the `M_PROTO` message block of the primitive do not represent the entire pattern. Subsequent `M_DATA` message

blocks (either on their own, or combined with a MG_DATA_REQ primitive) represent additional data corresponding to the action.

The *mg_action* field can assume one of the following values:

MG_ACTION_SEND_PATTERN
: Specifies that the MGS provider is to send the pattern provided in associated M_DATA message blocks accompanying the M_PROTO message block of the MG_ACTION_REQ primitive. The default duration is to send the pattern once.

MG_ACTION_REPEAT_PATTERN
: Specifies that the MGS provider is to repeat the pattern provided in the associated M_DATA message blocks accompanying the M_PROTO message block of the MG_ACTION_REQ primitive. The default duration is to repeat the pattern indefinitely.[1]

MG_ACTION_LOOPBACK
: Specifies that the MGS provider is to loopback the received media stream to the sent media stream for a specific (or all connected) termination point(s) in the specified session. The default duration is to repeat the condition indefinitely.

MG_ACTION_TEST_CONT
: Specifies that the MGS provider is to apply a continuity test tone to a specific (or all connected) termination point(s) in the specified session. The default duration is to repeat the pattern indefinitely.

MG_ACTION_TEST_MILLIWATT
: Specifies that the MGS provider is to apply milliwatt test tone to a specific (or all connected) termination point(s) in the specified session. This capability is used to provide standard 100 test lines. The default duration is to repeat the pattern indefinitely.

MG_ACTION_TEST_SILENT
: Specifies that the MGS provider is to apply silent termination to a specific (or all connected) termination point(s) in the specified session. This capability is used to provide standard 100 test lines. The default duration is to repeat the pattern indefinitely.

MG_ACTION_TEST_BALANCED
: Specifies that the MGS provider is to apply balanced termination to a specific (or all connected) termination point(s) in the specified session. This capability is used to provide standard 100 test lines. The default duration is to repeat the pattern indefinitely.

MG_ACTION_US_RINGBACK
: Specifies that the MGS provider is to apply a ringback pattern, according to US standards, to the specific (or all connected) termination point(s) in the specified session. The default duration is to repeat the pattern indefinitely.

MG_ACTION_US_BUSY
: The MGS provider is to apply US busy pattern (T60). The default duration is to repeat the pattern indefinitely.

MG_ACTION_US_REORDER
: The MGS provider is to apply US reorder tone (T120). The default duration is to repeat the pattern indefinitely.

[1] Note: care should be taken with attempting to repeat very long patterns. The MGS provider is permitted to refuse repetition of patterns on the basis of their length.

`MG_ACTION_US_PERM_SIGNAL`
: The MGS provider is to apply US receiver off hook (ROH). The default duration is to repeat the pattern indefinitely.

`MG_ACTION_US_BONG`
: The MGS provider is to apply US bong tone. The default duration is to repeat the pattern once only.

`MG_ACTION_EU_RINGBACK`
: The MGS provider is to apply EU ringback. The default duration is to repeat the pattern indefinitely.

`MG_ACTION_EU_BUSY`
: The MGS provider is to apply EU busy. The default duration is to repeat the pattern indefinitely.

`MG_ACTION_EU_REORDER`
: The MGS provider is to apply EU reorder. The default duration is to repeat the pattern indefinitely.

`MG_ACTION_EU_PERM_SIGNAL`
: The MGS provider is to apply EU receiver of hook.

`MG_ACTION_EU_BONG`
: The MGS provider is to apply EU bong tone. The default duration is to repeat the pattern once only.

`MG_ACTION_MF_0`
: The MGS provider is to apply DTMF tones corresponding to digit 0. If the duration is not specified, the default duration is 200 milliseconds.

`MG_ACTION_MF_1`
: The MGS provider is to apply DTMF tones corresponding to digit 1. If the duration is not specified, the default duration is 200 milliseconds.

`MG_ACTION_MF_2`
: The MGS provider is to apply DTMF tones corresponding to digit 2. If the duration is not specified, the default duration is 200 milliseconds.

`MG_ACTION_MF_3`
: The MGS provider is to apply DTMF tones corresponding to digit 3. If the duration is not specified, the default duration is 200 milliseconds.

`MG_ACTION_MF_4`
: The MGS provider is to apply DTMF tones corresponding to digit 4. If the duration is not specified, the default duration is 200 milliseconds.

`MG_ACTION_MF_5`
: The MGS provider is to apply DTMF tones corresponding to digit 5. If the duration is not specified, the default duration is 200 milliseconds.

`MG_ACTION_MF_6`
: The MGS provider is to apply DTMF tones corresponding to digit 6. If the duration is not specified, the default duration is 200 milliseconds.

`MG_ACTION_MF_7`
: The MGS provider is to apply DTMF tones corresponding to digit 7. If the duration is not specified, the default duration is 200 milliseconds.

`MG_ACTION_MF_8`
: The MGS provider is to apply DTMF tones corresponding to digit 8. If the duration is not specified, the default duration is 200 milliseconds.

`MG_ACTION_MF_9`
: The MGS provider is to apply DTMF tones corresponding to digit 9. If the duration is not specified, the default duration is 200 milliseconds.

`MG_ACTION_MF_A`
: The MGS provider is to apply DTMF tones corresponding to digit A. If the duration is not specified, the default duration is 200 milliseconds.

`MG_ACTION_MF_B`
: The MGS provider is to apply DTMF tones corresponding to digit B. If the duration is not specified, the default duration is 200 milliseconds.

`MG_ACTION_MF_C`
: The MGS provider is to apply DTMF tones corresponding to digit C. If the duration is not specified, the default duration is 200 milliseconds.

`MG_ACTION_MF_D`
: The MGS provider is to apply DTMF tones corresponding to digit D. If the duration is not specified, the default duration is 200 milliseconds.

`MG_ACTION_MF_E`
: The MGS provider is to apply DTMF tones corresponding to digit E. If the duration is not specified, the default duration is 200 milliseconds.

`MG_ACTION_MF_F`
: The MGS provider is to apply DTMF tones corresponding to digit F. If the duration is not specified, the default duration is 200 milliseconds.

`MG_ACTION_WAIT`
: The MGS provider is to wait for the specified duration. If the duration is not specified, the default duration is 2 seconds.

State

This primitive is valid for existing sessions and termination points in any state.

New State

The state of the requesting Stream remains unchanged. The state of the session and contained termination points remain unchanged.

Response

The action request service primitive requires that the MGS provider acknowledge receipt of the primitive. The MGS user is not permitted to issue any MGI service primitives until it receives an acknowledgement to this primitive. Receipt acknowledgement is given by the MGS provider as follows:

– **Successful**: When successful, the MGS provider acknowledges successful completion of the action service with an `MG_OK_ACK` primitive. The new state is `MGS_WCON_AREQ`. When the MGS provider eventually initiates the action, it confirms the initiation of the action with a `MG_ACTION_CON` primitive and the new state is then `MGS_WIND_AREQ`.

– **Unsuccessful (non-fatal errors)**: When unsuccessful, the MGS provider acknowledges the failure of the connect service with a `MG_ERROR_ACK` primitive containing the error. The new state remains unchanged.

Reasons for Failure

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

`[MGSYSERR]`	UNIX system error.
`[MGBADADDR]`	Bad address format or content.
`[MGOUTSTATE]`	Interface out of state.
`[MGBADOPT]`	Bad options format or content.
`[MGBADPARM]`	Bad parameter format or content.
`[MGBADPARMTYPE]`	Bad paramater structure type.
`[MGBADFLAG]`	Bad flag.
`[MGBADPRIM]`	Bad primitive.
`[MGNOTSUPP]`	Primitive not supported.
`[MGBADSLOT]`	Bad multplex slot.

4.2.3.2 MG_ACTION_CON

Description

This MGS provider initiated service primitive is used by the MGS provider to inform the MGS user that a requested action has begun.

Format

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

```
typedef struct MG_action_con {
    mg_ulong mg_primitive;      /* always MG_ACTION_CON */
    mg_ulong mg_action;         /* confirmed action */
    mg_ulong mg_se_id;          /* session id */
    mg_ulong mg_tp_id;          /* termination id for action confirmed */
    mg_ulong mg_action_id;      /* action identifier */
} MG_action_con_t;
```

Parameters

The action confirmation service primitive contains the following parameters:

mg_primitive
: Indicates the service primitive type. Always `MG_ACTION_CON`.

mg_action

mg_se_id

mg_tp_id

mg_action_id

State

New State

Response

Reasons for Failure

4.2.3.3 MG_ACTION_IND

Description

This MGS provider initiated service primitive is used by the MGS provider to inform the MSG user that a requested action has terminated or has reached a repetition point.

Format

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

```
typedef struct MG_action_ind {
    mg_ulong mg_primitive;      /* always MG_ACTION_IND */
    mg_ulong mg_action;         /* completed action */
    mg_ulong mg_se_id;          /* session id */
    mg_ulong mg_tp_id;          /* termination id for action completed */
    mg_ulong mg_action_id;      /* action identifier */
} MG_event_ind_t;
```

Parameters

The action indication service primitive contains the following parameters:

mg_primitive
: Specifies the service primitive type. Always `MG_ACTION_IND`.

mg_action

mg_se_id

mg_tp_id

mg_action_id

State

New State

Response

Reasons for Failure

4.2.3.4 MG_ABORT_REQ

Description

This MGS user originated primitive is used by the MGS user to terminate an ongoing action requested on a session or termination point within a session.

Format

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

```
typedef struct MG_abort_req {
    mg_ulong mg_primitive;      /* always MG_ABORT_REQ */
    mg_ulong mg_se_id;          /* session id */
    mg_ulong mg_tp_id;          /* termination id for action to abort */
    mg_ulong mg_action_id;      /* identifier of action to abort */
} MG_abort_req_t;
```

Parameters

The abort request service primitive contains the following parameters:

mg_primitive
: Specifies the service primitive type. Always `MG_ABORT_REQ`.

mg_se_id

mg_tp_id

mg_action_id

State

New State

Response

Reasons for Failure

4.2.4 Disconnection Service Primitives

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

These service primitives implement the disconnection service (see Section 3.2.8 [Disconnection Service], page 26).

4.2.4.1 MG_DISCON_REQ

Description

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

Format

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

```
typedef struct MG_disconnect_req {
    mg_ulong mg_primitive;   /* always MG_DISCON_REQ */
    mg_ulong mg_conn_flags;  /* direction to disconnect */
    mg_ulong mg_slot;        /* slot within multiplex */
} MG_disconnect_req_t;
```

Parameters

The disconnect request service primitive contains the following parameters:

mg_primitive
: Specifies the service primitive type. Always `MG_DISCON_REQ`.

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

  `MGF_RX_DIR` Specifies that the MGS user Stream is to be disconnected from the medium in the receive direction.

  `MGF_TX_DIR` Specifies that the MGS user Stream is to be disconnected from the medium in the transmit direction.

  `MGF_MONITOR` Specifies that the MGS user Stream is to be discconnected from the medium in monitoring (tap) mode.

mg_slot
: Where the PPA is associated with a multiplexed medium, this parameter specifies the slots within the mutliplexed media that have been autonomouosly disconnected. The form of the slot specification is provider- and media-specific. See also [Multiplex Media], page 13.

  Where the PPA specifies a single channel for a medium, this parameter is set to zero (‘`0`’) by the MGS provider on MGS provider originated primitives and is ignored by the MGS provider on MGS user originated primitives.

State

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

New State

The state remains unchanged.

Response

The disconnect request service primitive requires that the MGS provider acknowledge receipt of the primitive as follows:

- **Successful**: When successful, the MGS provider acknowledges successful completion of the connect service with a `MG_OK_ACK` primitive. The new state is `MGS_WCON_DREQ`. When the MGS provider eventually completes the disconnection, it confirms the disconnect with a `MG_DISCON_CON` primitive and the new state is then `MGS_ENABLED`.
- **Unsuccessful (non-fatal errors)**: When unsuccessful, the MGS provider acknowledges the failure of the connect service with a `MG_ERROR_ACK` primitive containing the error. The new state remains unchanged.

Reasons for Failure

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

`[MGSYSERR]`	UNIX system error.
`[MGBADADDR]`	Bad address format or content.
`[MGOUTSTATE]`	Interface out of state.
`[MGBADOPT]`	Bad options format or content.
`[MGBADPARM]`	Bad parameter format or content.
`[MGBADPARMTYPE]`	Bad paramater structure type.
`[MGBADFLAG]`	Bad flag.
`[MGBADPRIM]`	Bad primitive.
`[MGNOTSUPP]`	Primitive not supported.
`[MGBADSLOT]`	Bad multplex slot.

4.2.4.2 MG_DISCON_CON

Description

This MGS provider originated primitive is issued by the MGS provider to confirm the successful completion of the disconnect service with the disconnection of the user Stream from the medium in the transmit, receive, or both, directions.

Format

```
typedef struct MG_disconnect_con {
    mg_ulong mg_primitive;
    mg_ulong mg_conn_flags;
    mg_ulong mg_slot;
} MG_disconnect_con_t;
```

Parameters

mg_primitive
: Indicates the service primitive type. Always `MG_DISCON_CON`.

mg_conn_flags
: Indicates the connect flags. This field is a bitwise OR of zero or more of the following flags:

`MGF_RX_DIR`	Confirms that the MGS user Stream was disconnected from the medium in the receive direction.
`MGF_TX_DIR`	Confirms that the MGS user Stream was disconnected from the medium in the transmit direction.
`MGF_MONITOR`	Confirms that the MGS user Stream was discconnected from the medium in monitoring (tap) mode.

mg_slot
: Where the PPA is associated with a multiplexed medium, this parameter indicates the slots within the mutliplexed media that are confirmed as disconnected. The form of the slot specification is provider- and media-specific. See also [Multiplex Media], page 13.

  Where the PPA specifies a single channel for a medium, this parameter is set to zero (‘`0`’) by the MGS provider on MGS provider originated primitives and is ignored by the MGS provider on MGS user originated primitives.

State

This primitive will only be issued by the MGS provider in the `MGS_WCON_DREQ` state.

New State

The new state of the interface is the `MGS_ENABLED` state.

4.2.4.3 MG_DISCON_IND

Description

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

Format

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

```
typedef struct MG_disconnect_ind {
    mg_ulong mg_primitive;   /* always MG_DISCON_IND */
    mg_ulong mg_conn_flags;  /* direction disconnected */
    mg_ulong mg_cause;       /* cause for disconnection */
    mg_ulong mg_slot;        /* slot within multiplex */
} MG_disconnect_ind_t;
```

Parameters

mg_primitive
: Indicates the service primitive type. Always `MG_DISCON_IND`.

mg_conn_flags
: Indicates the connect flags. This field is a bitwise OR of zero or more of the following flags:

  `MGF_RX_DIR` Indicates that the MGS user Stream disconnected from the medium in the receive direction.

  `MGF_TX_DIR` Indicates that the MGS user Stream disconnected from the medium in the transmit direction.

  `MGF_MONITOR` Indicates that the MGS user Stream discconnected from the medium in monitoring (tap) mode.

mg_cause Indicates the cause of the autonomous disconnect.

mg_slot Where the PPA is associated with a multiplexed medium, this parameter indicates the slots within the mutliplexed media that have autonomously disconnected. The form of the slot specification is provider- and media-specific. See also [Multiplex Media], page 13.

Where the PPA specifies a single channel for a medium, this parameter is set to zero (‘`0`’) by the MGS provider on MGS provider originated primitives and is ignored by the MGS provider on MGS user originated primitives.

State

This primtiive will only be issued by the MGS provider in the `MGS_CONNECTED` state.

New State

The new state is `MGS_ENABLED`.

4.3 Diagnostics Requirements

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

4.3.1 Non-Fatal Error Handling Facility

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

4.3.2 Fatal Error Handling Facility

These errors are issued by the MG provider when it detects errors that are not correctable by the MG user, or if it is unable to report a correctible error to the MG 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 MGS 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.

5 MGI Input-Output Controls

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

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

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

5.1 MGI Configuration

These input-output controls can be used to interrogate or negotiate the configuration of a given channel or group of channels.

```
typedef struct mg_config {
    mg_ulong type;         /* unused */
    mg_ulong encoding;     /* encoding */
    mg_ulong block_size;   /* data block size (bits) */
    mg_ulong samples;      /* samples per block */
    mg_ulong sample_size;  /* sample size (bits) */
    mg_ulong rate;         /* clock rate (samples/second) */
    mg_ulong tx_channels;  /* number of tx channels */
    mg_ulong rx_channels;  /* number of rx channels */
    mg_ulong opt_flags;    /* options flags */
} mg_config_t;
```

The multiplex configuration structure, `mg_config_t`, contains the following members:

type This member is only to maintain alignment with the equivalient parameter structure as defined in the MGI and unused in the input-output control.

encoding Indicates or specifies the encoding associated with the multiplex. When the multiplex is used for any form of data, `MG_ENCODING_NONE` will be indicated and should be specified. *encoding* can be one of the following values:

`MG_ENCODING_NONE`	No encoding. Used for data or other clear channel information.
`MG_ENCODING_CN`	CN.
`MG_ENCODING_DVI4`	DVI4.
`MG_ENCODING_FS1015`	FIPS FS 1015 LPC.
`MG_ENCODING_FS1016`	FIPS FS 1016 LPC.
`MG_ENCODING_G711_PCM_A`	G.711 PCM A-law.

MG_ENCODING_G711_PCM_L	G.711 PCM Linear.
MG_ENCODING_G711_PCM_U	G.711 PCM Mu-law.
MG_ENCODING_G721	G.721.
MG_ENCODING_G722	G.722.
MG_ENCODING_G723	G.723.
MG_ENCODING_G726	G.726.
MG_ENCODING_G728	G.728.
MG_ENCODING_G729	G.729.
MG_ENCODING_GSM	GSM.
MG_ENCODING_GSM_EFR	GSM Extended Full-Rate.
MG_ENCODING_GSM_HR	GSM Half-Rate.
MG_ENCODING_LPC	LPC.
MG_ENCODING_MPA	MPA.
MG_ENCODING_QCELP	QCELP.
MG_ENCODING_RED	RED.
MG_ENCODING_S16_BE	Signed 16-bit Big-Endian.
MG_ENCODING_S16_LE	Signed 16-bit Little-Endian.
MG_ENCODING_S8	Sign 8-bit.
MG_ENCODING_U16_BE	Unsigned 16-bit Big-Endian.
MG_ENCODING_U16_LE	Unsigned 16-bit Little-Endian.
MG_ENCODING_U8	Unsigned 8-bit.
MG_ENCODING_VDVI	DVI.

block_size Specifies or indicates the block size associated with the multiplex. The block size is the number of samples that are written or read at one time. If this value is less than the size of a STREAMS fast buffer, `FASTBUF`, then a `FASTBUF` of samples will be read or written at once.

samples Specifies or indicates the number of samples (from the same timeslot) in a block.

sample_size

Specifies or indicates the sample size in bits. This can normally be 3, 4, 5, 7, 8, 12, 14 or 16.

rate Specifies or indicates the rate of the multiplex. This is the rate in samples per second. *rate* can be one of the following values:

MG_RATE_VARIABLE	The rate is variable.
MG_RATE_8000	56kbps or 64kbps.
MG_RATE_11025	11kHz Audio.
MG_RATE_16000	16kHz Audio.
MG_RATE_22050	22kHz Audio.
MG_RATE_44100	44kHz Audio.
MG_RATE_90000	90kHz Audio.
MG_RATE_184000	23B.
MG_RATE_192000	T1 (24B).
MG_RATE_240000	30B.
MG_RATE_248000	E1 (31B).

tx_channels

Specifies or indicates the number of transmit channels available. For the MG interface, this value is either 0 or 1.

rx_channels
: Specifies or indicates the number of receive channels available. For the MG interface, this value is either 0, 1, or 2. (The value of 2 is used for monitoring mode where two receive channels exists and zero transmit channels.)

opt_flags
: Specifies or indicates the options associated with the MG provider. MG provider options are provider specific and no generic options have yet been defined.

5.1.1 MGI Get Configuration

`MG_IOCGCONFIG`

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

5.1.2 MGI Set Configuration

`MG_IOCSCONFIG`

Set the media gateway configuration. Upon success, the media gateway configuration is read from the memory extent specified by the pointer argument to the `ioctl(2s)` call.

5.1.3 MGI Test Configuration

`MG_IOCTCONFIG`

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

5.1.4 MGI Commit Configuration

`MG_IOCCCONFIG`

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

Normally, the argument to the `MG_IOCCCONFIG` call is the same as to an immediately preceding `MG_IOCTCONFIG` call.

5.2 MGI Options

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

5.3 MGI State

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

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

```
typedef struct mg_statem {
    mg_ulong index;
    mg_ulong type;
    mg_ulong rate;
    mg_ulong mode;
    mg_ulong admin_state;
    mg_ulong usage_state;
    mg_ulong avail_status;
    mg_ulong ctrl_status;
} mg_statem_t;
```

The media gateway state structure, `mg_statem_t`, contains the following members:

index Provides time slot index for the channel. For T1 and J1 spans, the time slots '`1`' through '`24`' index the corresponding time slot in the span. For E1 spans, the time slot indices '`1`' throught '`31`' index the corresponding time slot in the span. For E1 operation, TS0 is unusable. For E1 CAS operation (where any channel in the span is configured for CAS), TS16 is not available to users for payload. For V.35 and other discrete synchronous channels, this index is '`1`'.

type Specifies or indicates whether the channel (or channels) has channel associated signalling or common channel signalling. This field can have one of the following values:

`MG_TYPE_NONE`
: For non-trunk channels, no type is necessary.

`MG_TYPE_CAS`
: For T1 and J1 span, channel associated signalling implies 56kbps DS0A operation for data within the channel.

`MG_TYPE_CCS`
: For E1, T1 or J1 spans, common channel signalling implies 64kbps DS0 oepration within the channel is indicated. For E1, CCS operation for the entire span implies that channel 17 (timeslot 16) is used for common channel signalling or is also available for payload. This is why it is typical on non-CAS E1 spans to place the signalling channel in timeslot 16 (e.g. the D-channel of a primary rate interface).

rate Specifies or indicates the bit rate of the channel in a single-rate channel, or of each channel in a multi-rate channel, or of each channel in a full-rate channel. Channels '`1`' through '`24`' for T1 and J1 can be 56kbps or 64kbps. Channels '`1`' through '`31`' for E1 are 64kbps but can be forced into 56kbps mode. The default is 64kbps for E1 CCS and CAS channels and T1 CCS channels; 56kbps for T1 CAS channels.

mode Specifies or indicates the channel mode. This is bitwise OR of zero or more of the following values:

`MG_MODE_REMLOOP`
: The receive data in the channel is looped back to replace the transmit data for the channel. This may either be accomplished within the host or using the per-channel loopback capability of some chip sets.

`MG_MODE_LOCLOOP`
: The transmit data for the channel is looped back to replace the receive data for the channel. This may be accomplished within the host.

`MG_MODE_TEST`
: The channel is marked for BERT testing. When BERT testing for the span is enabled on a channel basis, this channel will be included in the channels upon which the BERT test pattern is transmitted.

Because tests are disruptive, no value can be added to this set unless the channel has a control status of "subject to test" or "reserved for test".

admin_state
: Specifies or indicates the administrative state of the channel. The administrative state can be one of the following values:

`MG_ADMIN_LOCKED`
: The administrative state is "locked". The channel is administratively prohibited from providing service to users.

`MG_ADMIN_UNLOCKED`
: The administrative state is "unlocked". The channel is administratively permitted to provide service to users.

`MG_ADMIN_SHUTDOWN`
: The administrative state is "shutting down". The channel will continue to provide service to existing users but will reject new users: once there are no more users of the channel, the channel will move to the "locked" state.

usage_state Specifies or indicates the usage state of the channel. The usage state can be one of the following values:

`MG_USAGE_IDLE`
: The channel is "idle". The channel is not currently in use.

`MG_USAGE_ACTIVE`
: The channel is "active". The channel is in use and has sufficient operating capacity to provide for additional users simultaneously (e.g. a half-channel is used).

`MG_USAGE_BUSY`
: The channel is "busy". The channel is in use and has no spare capacity (i.e. the full channs is in use).

If partial channels are not supported, only the values "idle" and "busy" are allowed.

avail_status
: Specifies or indicates the availabiltiy status of the channel. The availablity status is a bitwise OR of zero or more of the following values:

`MG_AVAIL_INTEST`
: The channel is "in test". The channel is undergoing a test procedure. The administrative state is "locked" and the operational state is "disabled". This condition exists while the span is in test in a manner disruptive to the channel, or when the channel is in loopback or test modes.

`MG_AVAIL_FAILED`
: The channel has "failed". The channel has an internal fault that prevents it from operating. The operational state is "disabled". This value is present when the same value is present in the span availability status.

`MG_AVAIL_POWEROFF`
: The channel has "power off". The channl requires power to be applied and is not powered on. For example, power management may have removed power from the device. This value is present when the same value is present in the span availablity status.

`MG_AVAIL_OFFLINE`
: The channel is "off line". The channel requires a outing operation to be performed to place it online and make it available for use. The operation may be manul or automatic, or both. The operational state is "disabled". This value is present when the same value is present in the span availability status.

`MG_AVAIL_OFFDUTY`
: The channel is "off duty". The channel has been made inactive by an internal control process in accordance with a predetermined time schedule. Under normal conditions, the control process can be expected to reactivate the channel at some scheduled time.

`MG_AVAIL_DEPEND`
: The channel has a "dependency". The channel cannot operate because some other resource on which it depends is unavailable (e.g. the span).

`MG_AVAIL_DEGRADED`
: The channel is "degraded". The channel is operating with degraded peformance. This value is present when the same value is present in the span availability status.

`MG_AVAIL_MISSING`
: The channel is "not installed". The channel is not present in the system or is incomplete.

`MG_AVAIL_LOGFULL`
: Not used.

ctrl_status Specifies or indicates the control status of the channel. The control status is a bitwise OR of zero or more of the following values:

`MG_CTRL_CANTEST`
: The channel is "subject to test". The channel is available to normal users but tests may be conducted on it simultaneously at unpredicatable times, which may cause it to exhibit unusual characteristics to users.

`MG_CTRL_PARTLOCK`
: The channel is "part of services locked". A manager has adminstratively locked some part of the channel.

`MG_CTRL_RESERVED`
: The channel is "reserved for test". The channel is undergoing a test procedure and is unavailable to users.

`MG_CTRL_SUSPENDED`
: The channel is "suspended". The channel service has been administratively suspended to users.

5.3.1 MGI Get State

`MG_IOCGSTATEM`

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

5.3.2 MGI Reset State

`MG_IOCCMRESET`

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

5.4 MGI Statistics

These input-output controls can be used to collect statistics or set staticstics collection intervals associated with a channel or group of channels.

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

```
typedef struct mg_stats {
    mg_ulong header;
    mg_ulong rx_octets;
    mg_ulong tx_octets;
    mg_ulong rx_overruns;
    mg_ulong tx_underruns;
    mg_ulong rx_buffer_overflows;
    mg_ulong tx_buffer_overflows;
    mg_ulong lead_cts_lost;
    mg_ulong lead_dcd_lost;
    mg_ulong carrier_lost;
    mg_ulong errored_seconds;
    mg_ulong severely_errored_seconds;
    mg_ulong severely_errored_framing_seconds;
    mg_ulong unavailable_seconds;
    mg_ulong controlled_slip_seconds;
    mg_ulong path_coding_violations;
    mg_ulong line_errored_seconds;
    mg_ulong bursty_errored_seconds;
    mg_ulong degraded_minutes;
    mg_ulong line_coding_violations;
} mg_stats_t;
```

The media gateway statistics structure, `mg_stats_t`, contains the following members:

header Specifies or indicates the statistics period header associated with the media gateway. This header is a statistics collection period in milliseconds.

rx_octets Indicates the number of octets received during the collection interval. This does not include octets for which there was a receiver overrun condition.

tx_octets Indicates the number of octets transmitted during the collection interval. This does not include octets for which there was a transmitter underrun condition.

rx_overruns
Indicates the number of receive overrun conditions that occurred during the collection interval. When the overrun condition spans interval boundaries, the condition is counted in the interval during which the overrun condition began.

tx_underruns
Indicates the number of transmitter underrun conditions that occurred during the collection interval. When the underrun condition spans interval boundaries, the condition is counted in the interval during which the underrun condition began.

rx_buffer_overflows
Indicates the number of receive buffer overflows that occured during the collection interval. Receive buffer overflow conditions occur when the driver is unable to allocate a message block or buffer for received bits, resulting in the discard of the received bits.

tx_buffer_overflows
Indicates the number of transmit buffer overflows that occured during the collection interval. Transmit buffer overflow conditions occur when the driver is unable to allocate a message block or buffer for transmit bits, resulting in the discard of the bits to be transmitted.

lead_cts_lost
Indicates the number of Clear To Send leads lost. That is, the number of times that the Clear To Send lead transitioned from asserted to deasserted.

lead_dcd_lost
Indicates the number of Data Carrier Detect leads lost. That is, the number of times that the Data Carrier Detect lead trasitioned from asserted to deasserted.

carrier_lost Indicates the number of Carrier lost conditions. That is, the number of times that an alarm or lead indicated that the facility carrier was lost.

errored_seconds
The number of errored seconds (ESs) in the current interval. An errored second has one or more path code violations, one or more out of frame defects, one or more controlled slip events, or a detected alarm indication signal (AIS) defect.

severely_errored_seconds
The number of severely errored seconds (SESs) in the current interval.

severely_errored_framing_seconds
The number of severely errored framing seconds (SEFSs) in the current interval. A severely errored framing second has one or more out of frame defects or a detected AIS defect.

unavailable_seconds
The number of unavailable seconds in the current interval.

controlled_slip_seconds
The number of controlled slip seconds (CSSs) in the current interval. A controlled slip second has one or more controlled slip events.

path_coding_violations
: The number of path coding violations (PCVs) in the current interval. A path coding violation is a fram synchronization bit error in the D4 and E1 no-CRC4 formats, or a CRC or frame synchronization bit error in the ESF and E1 CRC4 formats.

line_errored_seconds
: The number of line errored seconds (LESs) in the current interval. A line errored second is a second in which one or more line code violation error events are detected.

bursty_errored_seconds
: The number of bursty errroed seconds (BESs) in the current interval. A bursty errored second has 2 to 319 path coding violation error events, no severely errored frame defects, and no detected inocming AIS defects.

degraded_minutes
: The number of degraded minutes (DMs) in the current interval.

line_coding_violations
: The number of line coding violations (LCVs) in the current interval. An LCV is the occurence of a bipolar violation (BPV) or excessive zeroes (EXZ) error event.

5.5 MGI Events

These input-output controls can be used to specify the events that will be reported by a channel or channels.

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

```
typedef struct mg_notify {
    mg_ulong events;
} mg_notify_t;
```

The media gateway events structure, `mg_notify_t`, contains the following members:

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

5.5.1 MGI Get Notify

`MG_IOCGNOTIFY`

Requests that the events associated with the media gateway be obtained and written to the `mg_notify_t` structure pointed to by the argument to the input-output control.

5.5.2 MGI Set Notify

`MG_IOCSNOTIFY`

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

5.5.3 MGI Clear Notify

`MG_IOCCNOTIFY`

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

5.6 MGI Commands

These input-output controls can be used to manage a channel or channels.

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

```
typedef struct mg_mgmt {
    mg_ulong cmd;
} mg_mgmt_t;
```

The media gateway management structure, `mg_mgmt_t`, contains the following members:

cmd Specifies the management command to be performed by the MGS provider. This member can have one of the following values:

`MG_CMD_REMLOOP`
: Place the multiplex in remote loopback. The administrative state of the multiplex must be "locked" for this command to be successfull. Once complete, the control status of the multiplex will contain "reserved for test" and the availability status of the multiplex will contain "in test".

`MG_CMD_LOCLOOP`
: Place the multiplex in local loopback. The administrative state of the multiplex must be "locked" for this command to be successfull. Once complete, the control status of the multiplex will contain "reserved for test" and the availabiltiy status of the multiplex will contain "in test".

`MG_CMD_FORTEST`
: Reserve the multiplex for BERT testing. The administrative state of the multiplex must be "locked" for this command to be successful. Once complete, the control status of the multiplex will contain "reserved for test" and the availability status of the multiplex will contain "in test" while BERT testing is actively being performed.

`MG_CMD_LOCK`
: Place the multiplex in the "locked" administrative state. If the multiplex is in the "unlocked" or "shutting down" states and the usage state is "busy", this will result in the removal from service of the multiplex while it is in use.

`MG_CMD_UNLOCK`
: Place the multiplex in the "unlocked" administrative state. This makes the multiplex adminstratively available for use.

`MG_CMD_SHUTDOWN`
: Place the multiplex in the "shutting down" administrative state. If the multiplex has a usage state of "idle" the multiplex will be placed immediately into the "locked" administrative state. If the usage state is "busy", then the administrative state will be set to "shutting down" and the driver

will wait until the multiplex is released before it is placed in the "locked" administrative state.

5.6.1 MGI Command

`MG_IOCCMGMT`

Request that the management command be read from the `mg_mgmt_t` structure pointed to by the argument to the input-output control and acted upon for the media gateway.

6 MGI Management

Mapping of MGI Primitives to ITU-T H.248

`Add` MG_JOIN_REQ, MG_JOIN_CON MG_CONN_REQ, MG_CONN_CON

`Modify` MG_OPTMGMT_REQ, MG_OPTMGMT_ACK MG_CONN_REQ, MG_CONN_CON MG_DISCON_REQ, MG_DISCON_IND, MG_DISCON_CON

`Subtract` MG_LEAVE_REQ, MG_LEAVE_IND, MG_LEAVE_CON

`Move` MG_LEAVE_REQ, MG_LEAVE_IND, MG_LEAVE_CON MG_DISCON_REQ, MG_DISCON_IND, MG_DISCON_CON MG_CONN_REQ, MG_CONN_CON MG_JOIN_REQ, MG_JOIN_CON

`AuditValue`
MG_OPTMGMT_REQ, MG_OPTMGMT_ACK

`AuditCapabilities`
MG_INFO_REQ, MG_INFO_ACK

`Notify` MG_NOTIFY_REQ, MG_NOTIFY_IND

`ServiceChange`
MG_DISCON_IND, MG_LEAVE_IND, MG_NOTIFY_IND

Mode Property

SendOnly

RecvOnly

SendRecv

Inactive

LoopBack

Addendum for ITU-T H.248 Conformance

Appendix A State/Event Tables

Appendix B Primitive Precedence Tables

Appendix C MGI Header Files

C.1 MGI Header File Listing

```
#ifndef __SS7_MG_H__
#define __SS7_MG_H__

#define MG_INFO_REQ              1UL
#define MG_OPTMGMT_REQ           2UL
#define MG_ATTACH_REQ            3UL
#define MG_DETACH_REQ            4UL
#define MG_JOIN_REQ              5UL
#define MG_ACTION_REQ            6UL
#define MG_ABORT_REQ             7UL
#define MG_CONN_REQ              8UL
#define MG_DATA_REQ              9UL
#define MG_DISCON_REQ           10UL
#define MG_LEAVE_REQ            11UL
#define MG_NOTIFY_REQ           12UL

#define MG_INFO_ACK             13UL
#define MG_OPTMGMT_ACK          14UL
#define MG_OK_ACK               15UL
#define MG_ERROR_ACK            16UL
#define MG_ATTACH_ACK           17UL
#define MG_JOIN_CON             18UL
#define MG_ACTION_CON           19UL
#define MG_ACTION_IND           20UL
#define MG_CONN_CON             21UL
#define MG_DATA_IND             22UL
#define MG_DISCON_IND           23UL
#define MG_DISCON_CON           24UL
#define MG_LEAVE_IND            25UL
#define MG_LEAVE_CON            26UL
#define MG_NOTIFY_IND           27UL

/*
 *  MG STATES
 */
#define MGS_UNINIT              -2UL
#define MGS_UNUSABLE            -1UL
#define MGS_DETACHED             0UL
#define MGS_WACK_AREQ            1UL
#define MGS_WCON_AREQ            2UL
#define MGS_WACK_UREQ            3UL
#define MGS_WCON_UREQ            4UL
#define MGS_ATTACHED             5UL
#define MGS_WACK_JREQ            6UL
#define MGS_WCON_JREQ            7UL
#define MGS_WACK_LREQ            8UL
#define MGS_WCON_LREQ            9UL
#define MGS_JOINED              10UL
#define MGS_WACK_CREQ           11UL
#define MGS_WCON_CREQ           12UL
```

```
#define MGS_WACK_DREQ           13UL
#define MGS_WCON_DREQ           14UL
#define MGS_CONNECTED           15UL

#define MGSF_UNINIT             0
#define MGSF_UNUSABLE           0
#define MGSF_DETACHED           (1<<MGS_DETACHED)
#define MGSF_WACK_AREQ          (1<<MGS_WACK_AREQ)
#define MGSF_WCON_AREQ          (1<<MGS_WCON_AREQ)
#define MGSF_WACK_UREQ          (1<<MGS_WACK_UREQ)
#define MGSF_WCON_UREQ          (1<<MGS_WCON_UREQ)
#define MGSF_ATTACHED           (1<<MGS_ATTACHED)
#define MGSF_WACK_JREQ          (1<<MGS_WACK_JREQ)
#define MGSF_WCON_JREQ          (1<<MGS_WCON_JREQ)
#define MGSF_WACK_LREQ          (1<<MGS_WACK_LREQ)
#define MGSF_WCON_LREQ          (1<<MGS_WCON_LREQ)
#define MGSF_JOINED             (1<<MGS_JOINED)
#define MGSF_WACK_CREQ          (1<<MGS_WACK_CREQ)
#define MGSF_WCON_CREQ          (1<<MGS_WCON_CREQ)
#define MGSF_WACK_DREQ          (1<<MGS_WACK_DREQ)
#define MGSF_WCON_DREQ          (1<<MGS_WCON_DREQ)
#define MGSF_CONNECTED          (1<<MGS_CONNECTED)

/*
 *  MG PROTOCOL PRIMITIVES
 */

/*
 *  MG_OPTMGMT_REQ
 *  -------------------------------------------------------------------------
 */
typedef struct MG_optmgmt_req {
        mg_ulong mg_primitive;          /* always MG_INFO_REQ */
        mg_ulong mg_opt_length;         /* length of options */
        mg_ulong mg_opt_offset;         /* offset of options */
        mg_ulong mg_mgmt_flags;         /* management flags */
} MG_optmgmt_req_t;

/*
 *  MG_OPTMGMT_ACK
 *  -------------------------------------------------------------------------
 */
typedef struct MG_optmgmt_ack {
        mg_ulong mg_primitive;          /* always MG_INFO_ACK */
        mg_ulong mg_opt_length;         /* length of options */
        mg_ulong mg_opt_offset;         /* offset of options */
        mg_ulong mg_mgmt_flags;         /* management flags */
} MG_optmgmt_ack_t;

#define MG_SUCCESS      0x00    /* indicates successful operation */
#define MG_FAILURE      0x01    /* indicates unsuccessful operation */
#define MG_SET_OPT      0x02    /* set options */
#define MG_GET_OPT      0x04    /* get options */
#define MG_NEGOTIATE    0x08    /* negotiate options */
#define MG_DEFAULT      0x10    /* default options */
```

```
typedef struct MG_channel_opt {
        mg_ulong mg_obj_type;           /* always MG_OBJ_TYPE_CH */
        mg_ulong mg_obj_id;             /* channel id */
        mg_ulong ch_type;               /* channel media type */
        mg_ulong ch_flags;              /* channel media options flags */
        mg_ulong ch_block_size;         /* data block size (bits) */
        mg_ulong ch_encoding;           /* encoding */
        mg_ulong ch_sample_size;        /* sample size (bits) */
        mg_ulong ch_rate;               /* clock rate (Hz) */
        mg_ulong ch_tx_channels;        /* number of tx channels */
        mg_ulong ch_rx_channels;        /* number of rx channels */
        mg_ulong ch_opt_flags;          /* options flags */
} MG_channel_opt_t;

typedef struct MG_connleg_opt {
        mg_ulong mg_obj_type;           /* always MG_OBJ_TYPE_LG */
        mg_ulong mg_obj_id;             /* channel id */
        mg_ulong lg_type;               /* conn leg media type */
        mg_ulong lg_flags;              /* conn leg media options flags */
        mg_ulong lg_block_size;         /* data block size (bits) */
        mg_ulong lg_encoding;           /* encoding */
        mg_ulong lg_sample_size;        /* sample size (bits) */
        mg_ulong lg_rate;               /* clock rate (Hz) */
        mg_ulong lg_tx_channels;        /* number of tx channels */
        mg_ulong lg_rx_channels;        /* number of rx channels */
        mg_ulong lg_opt_flags;          /* options flags */
} MG_connleg_opt_t;

typedef struct MG_session_opt {
        mg_ulong mg_obj_type;           /* always MG_OBJ_TYPE_SE */
        mg_ulong mg_obj_id;             /* session id */
        mg_ulong se_type;               /* session media type */
        mg_ulong se_flags;              /* session media options flags */
        mg_ulong se_block_size;         /* data block size (bits) */
        mg_ulong se_encoding;           /* encoding */
        mg_ulong se_sample_size;        /* sample size (bits) */
        mg_ulong se_rate;               /* clock rate (Hz) */
        mg_ulong se_tx_channels;        /* number of tx channels */
        mg_ulong se_rx_channels;        /* number of rx channels */
        mg_ulong se_opt_flags;          /* options flags */
} MG_session_opt_t;

#define SEF_INTERWORKING        0x01    /* encoding interworking */
#define SEF_CONFERENCING        0x02    /* conferencing in effect */
#define SEF_CLEARCHANNEL        0x04    /* clear channel enforced */

typedef union MG_options {
        struct {
                mg_ulong mg_obj_type;   /* object type */
                mg_ulong mg_obj_id;     /* object id */
        } obj;
        struct MG_channel_opt ch;       /* channel options */
        struct MG_connleg_opt lg;       /* conn leg options */
        struct MG_session_opt se;       /* session options */
} MG_options_t;
```

```
#define MG_SE_OPT_AUTO_GAIN_CONTROL     0x01    /* perform Automatic Gain Control */
#define MG_SE_OPT_LIMITING              0x02    /* perform limiting */
#define MG_SE_OPT_COMPRESSION           0x04    /* perform compression */

/*
 *  MG_INFO_REQ
 *  -------------------------------------------------------------------------
 *  Requests information about the MG stream including its current channel
 *  configuration.
 */
typedef struct MG_info_req {
        mg_ulong mg_primitive;          /* always MG_INFO_REQ */
} MG_info_req_t;

/*
 *  MG_INFO_ACK
 *  -------------------------------------------------------------------------
 *  Provides information about the MG stream and provide including the current
 *  channel configuration.
 */
typedef struct MG_info_ack {
        mg_ulong mg_primitive;          /* always MG_INFO_ACK */
        mg_ulong mg_se_id;              /* session id */
        mg_ulong mg_opt_length;         /* channel options length */
        mg_ulong mg_opt_offset;         /* channel options offset */
        mg_ulong mg_prov_flags;         /* provider options flags */
        mg_ulong mg_style;              /* provider style */
        mg_ulong mg_version;            /* provider version */
} MG_info_ack_t;

#define MG_STYLE1       0x0     /* does not perform attach */
#define MG_STYLE2       0x1     /* does perform attach */

#define MG_VERSION_1_0  0x10    /* version 1.0 of interface */
#define MG_VERSION      MG_VERSION_1_0

/*
 *  MG_ATTACH_REQ
 *  -------------------------------------------------------------------------
 *  Requests that the specified slot on the requesting stream (mg_mx_id == 0)
 *  or specified multiplex (mg_mx_id != 0) be associated with a newly created
 *  channel with specified (mg_ch_id non-zero) or provider assigned (mg_ch_id
 *  zero) channel id.
 *
 *  This primitive is acknowledged with the MG_ATTACH_ACK.
 *
 *  If the requesting stream is closed, all channels attached to the
 *  requesting stream will be detached.    Only channels that are associated
 *  with the requesting stream need be assigned in this fashion.
 */
typedef struct MG_attach_req {
        mg_ulong mg_primitive;          /* always MG_ATTACH_REQ */
        mg_ulong mg_mx_id;              /* multiplex id (or 0 for requesting stream) */
        mg_ulong mg_mx_slot;            /* multiplex slot number */
        mg_ulong mg_ch_id;              /* channel id (or 0 for provider assignment) */
        mg_ulong mg_ch_type;            /* type of channel */
```

```
} MG_attach_req_t;

/*
 *  MG_ATTACH_ACK
 *  -------------------------------------------------------------------------
 *  Acknowledges an MG_ATTACH_REQ.  Returns the channel id.
 */
typedef struct MG_attach_ack {
        mg_ulong mg_primitive;          /* always MG_ATTACH_ACK */
        mg_ulong mg_mx_id;              /* multiplex id (or 0 for requesting stream) */
        mg_ulong mg_mx_slot;            /* multiplex slot number */
        mg_ulong mg_ch_id;              /* channel id assignment */
} MG_attach_ack_t;

/*
 *  MG_DETACH_REQ
 *  -------------------------------------------------------------------------
 *  Requests that the specified requesting stream channel be detached.    This
 *  primitive is acknowledged with the MG_OK_ACK.
 */
typedef struct MG_detach_req {
        mg_ulong mg_primitive;          /* always MG_DETACH_REQ */
        mg_ulong mg_ch_id;              /* channel id */
} MG_detach_req_t;

/*
 *  MG_JOIN_REQ
 *  -------------------------------------------------------------------------
 *  Requests that the specified channels be joined to the specified session
 *  with newly created termination point with id specified by the caller
 *  (mg_tp_id non-zero) or assigned by the provider (mg_tp_id zero).    If the
 *  specified session id is zero then a new session will be created.    If the
 *  specified session id exists, the existing session will be joined.    If the
 *  specified session id is non-zero but does not exist, one will be created
 *  wtih the specified id.    All channels must be successfully enabled before
 *  this primitive will be confirmed.    This primtive is confirmed with the
 *  MG_JOIN_CON primitive.    It is refused with the MG_ERROR_ACK or
 *  MG_LEAVE_IND primitive.
 */
typedef struct MG_join_req {
        mg_ulong mg_primitive;          /* always MG_JOIN_REQ */
        mg_ulong mg_se_id;              /* session to join (0 to create) */
        mg_ulong mg_tp_id;              /* joined termination (0 for new) */
        mg_ulong mg_channel_length;     /* channel ids that make up termination */
        mg_ulong mg_channel_offset;     /* channel ids that make up termination */
} MG_join_req_t;

/*
 *  MG_JOIN_CON
 *  -------------------------------------------------------------------------
 *  Confirms that the previous join request was successful and the session id
 *  and termination point id of the join.
 */
typedef struct MG_join_con {
        mg_ulong mg_primitive;          /* always MG_JOIN_CON */
        mg_ulong mg_se_id;              /* joined session */
```

```
        mg_ulong mg_tp_id;              /* joined termination */
} MG_join_con_t;

/*
 *  MG_ACTION_REQ, M_PROTO w/ 0 or more M_DATA
 *  -------------------------------------------------------------------------
 *  Requests that the action of the requested type be performed against the
 *  specified session and termination point for the requested duration and
 *  with the requested options.
 *
 *  When more data is indicated using the MG_MORE_DATA flag, it indicates that
 *  subsequent MG_ACTION_REQ primitives contain more data for the associated
 *  pattern.    The data is encoded according to the media format of the
 *  requesting stream.
 *
 *  When the requested duration is zero, the action will continue until its
 *  normal termination, or until subsequently aborted.
 *
 *  Actions on terminations which are currently connected in a communications
 *  session will be mixed with the media received from the communications
 *  session and any other actions which are currently being performed on the
 *  termination.
 *
 *  Actions on terminations which are currently disconnected from a
 *  communications session will be be mixed with the media from other actions
 *  on the termination point.
 *
 *  Some actions can only be performed on disconnected termination points
 *  (e.g., MG_ACTION_LOOPBACK, MG_ACTION_TEST_SILENT).
 *
 *  Some actions replace all other actions on the termination point (e.g.,
 *  MG_ACTION_LOOPBACK, MG_ACTION_TEST_SILENT).
 *
 *  Some actions performed on a termination point will be performed on
 *  individual channels that make up the termination point (e.g.
 *  MG_ACTION_LOOPBACK).
 */
typedef struct MG_action_req {
        mg_ulong mg_primitive;          /* always MG_ACTION_REQ */
        mg_ulong mg_action;             /* requested action */
        mg_ulong mg_se_id;              /* session id */
        mg_ulong mg_tp_id;              /* termination id to perform action */
        mg_ulong mg_duration;           /* duration in milliseconds */
        mg_ulong mg_flags;              /* option flags */
} MG_action_req_t;

#define MG_ACTION_FIRST                 1
#define MG_ACTION_SEND_PATTERN          1       /* send the provided pattern */
#define MG_ACTION_REPEAT_PATTERN        2       /* repeat the provided pattern */
#define MG_ACTION_LOOPBACK              3       /* apply loopback */
#define MG_ACTION_TEST_CONT             4       /* apply continuity test tone */
#define MG_ACTION_TEST_MILLIWATT        5       /* apply milliwatt */
#define MG_ACTION_TEST_SILENT           6       /* apply silent termination */
#define MG_ACTION_TEST_BALANCED         7       /* apply ballanced termination */
#define MG_ACTION_US_RINGBACK           8       /* apply US ringback */
#define MG_ACTION_US_BUSY               9       /* apply US busy */
```

```
#define MG_ACTION_US_REORDER            10      /* apply US reorder */
#define MG_ACTION_US_PERM_SIGNAL        11      /* apply US receiver of hook */
#define MG_ACTION_US_BONG               12      /* apply US bong tone */
#define MG_ACTION_EU_RINGBACK           13      /* apply EU ringback */
#define MG_ACTION_EU_BUSY               14      /* apply EU busy */
#define MG_ACTION_EU_REORDER            15      /* apply EU reorder */
#define MG_ACTION_EU_PERM_SIGNAL        16      /* apply EU receiver of hook */
#define MG_ACTION_EU_BONG               17      /* apply EU bong tone */
#define MG_ACTION_MF_0                  20      /* apply MF 0 */
#define MG_ACTION_MF_1                  21      /* apply MF 1 */
#define MG_ACTION_MF_2                  22      /* apply MF 2 */
#define MG_ACTION_MF_3                  23      /* apply MF 3 */
#define MG_ACTION_MF_4                  24      /* apply MF 4 */
#define MG_ACTION_MF_5                  25      /* apply MF 5 */
#define MG_ACTION_MF_6                  26      /* apply MF 6 */
#define MG_ACTION_MF_7                  27      /* apply MF 7 */
#define MG_ACTION_MF_8                  28      /* apply MF 8 */
#define MG_ACTION_MF_9                  29      /* apply MF 9 */
#define MG_ACTION_MF_A                  30      /* apply MF A */
#define MG_ACTION_MF_B                  31      /* apply MF B */
#define MG_ACTION_MF_C                  32      /* apply MF C */
#define MG_ACTION_MF_D                  33      /* apply MF D */
#define MG_ACTION_MF_E                  34      /* apply MF E */
#define MG_ACTION_MF_F                  35      /* apply MF F */
#define MG_ACTION_WAIT                  36      /* wait for specifie duration */
#define MG_ACTION_LAST                  36

#define MG_MORE_DATA                    0x01

/*
 *  MG_ACTION_CON, M_PROTO w/ 0 or more M_DATA
 *  -------------------------------------------------------------------------
 *  Confirms that the requested action has begun.    MG_ACTION_REQ which have
 *  the MG_MORE_DATA flag set will not be confirmed until the last
 *  MG_ACTION_REQ has been issued by the MG user.  The end of restricted
 *  duration actions will be indicated with MG_ACTION_IND.
 */
typedef struct MG_action_con {
        mg_ulong mg_primitive;          /* always MG_ACTION_CON */
        mg_ulong mg_action;             /* confirmed action */
        mg_ulong mg_se_id;              /* session id */
        mg_ulong mg_tp_id;              /* termination id for action confirmed */
        mg_ulong mg_action_id;          /* action identifier */
} MG_action_con_t;

/*
 *  MG_ACTION_IND, M_PROTO
 *  -------------------------------------------------------------------------
 *  Indicates that the action identified by the indicated action identifier
 *  has completed.
 */
typedef struct MG_action_ind {
        mg_ulong mg_primitive;          /* always MG_ACTION_CON */
        mg_ulong mg_action;             /* completed action */
        mg_ulong mg_se_id;              /* session id */
        mg_ulong mg_tp_id;              /* termination id for action completed */
```

```
        mg_ulong mg_action_id;          /* action identifier */
} MG_event_ind_t;

/*
 *  MG_ABORT_REQ, M_PROTO
 *  -------------------------------------------------------------------------
 *  Requests that the specified action be aborted.    This primitive is
 *  confirmed with the MG_OK_ACK primiitve.    If the action identifier is
 *  MG_ACTION_PREVIOUS, this primitive requests that the previously initiated
 *  (unconfirmed) action be aborted.    If the action identifier is zero, this
 *  primitive requests that all actions on the specified termination point be
 *  aborted.
 */
typedef struct MG_abort_req {
        mg_ulong mg_primitive;          /* always MG_ABORT_REQ */
        mg_ulong mg_se_id;              /* session id */
        mg_ulong mg_tp_id;              /* termination id for action to abort */
        mg_ulong mg_action_id;          /* identifier of action to abort */
} MG_abort_req_t;

#define MG_ACTION_PREVIOUS      (-1UL)

/*
 *  MG_CONN_REQ
 *  -------------------------------------------------------------------------
 *  Request that the requested termination point be connected into the
 *  communications session in the directions indicated by mg_conn_flags, with
 *  the digital padding specified and the optional topology description.
 *
 *  If the optional topology description is not included, it is assumed that
 *  the termination point is requested to be connected to all other
 *  participants in the communications session in the directions requested.
 *
 *  If the optional topology description is included, it contains the list of
 *  other termination points in the session to which the the specified
 *  termination point is to be connected in the directions requested.
 */
typedef struct MG_conn_req {
        mg_ulong mg_primitive;          /* always MG_CONN_REQ */
        mg_ulong mg_se_id;              /* session id */
        mg_ulong mg_tp_id;              /* termination point */
        mg_ulong mg_conn_flags;         /* connection flags */
        mg_ulong mg_padding;            /* digital padding */
        mg_ulong mg_topology_length;    /* length of topology to connect */
        mg_ulong mg_topology_offset;    /* offset of topology to connect */
} MG_conn_req_t;

/*
   connect flags
 */
#define MGF_IC_DIR      0x01
#define MGF_OG_DIR      0x02
#define MGF_BOTH_DIR    (MGF_IC_DIR|MGF_OG_DIR)

/*
 *  MG_CONN_CON
```

```
 *  -------------------------------------------------------------------------
 *  Confirms that the requested connection primitive has been successfully
 *  completed.
 */
typedef struct MG_conn_con {
        mg_ulong mg_primitive;          /* always MG_CONN_CON */
        mg_ulong mg_se_id;              /* session id */
        mg_ulong mg_tp_id;              /* connecting channel id */
} MG_conn_con_t;

/*
 *  MG_DATA_REQ, M_PROTO w/ M_DATA, prefferably just M_DATA.
 *  -------------------------------------------------------------------------
 *  Sends channel data to the session from the requesting MG stream.
 */
typedef struct MG_data_req {
        mg_ulong mg_primitive;          /* always MG_DATA_REQ */
        mg_ulong mg_flags;              /* data flags */
        mg_ulong mg_mx_slot;            /* multiplex slot number */
} MG_data_req_t;

/*
 *  MG_DATA_IND, M_PROTO w/ M_DATA, prefferably just M_DATA.
 *  -------------------------------------------------------------------------
 *  Receives channel data from the session on the MG stream.
 */
typedef struct MG_data_ind {
        mg_ulong mg_primitive;          /* always MG_DATA_IND */
        mg_ulong mg_flags;              /* data flags */
        mg_ulong mg_mx_slot;            /* multiplex slot number */
} MG_data_ind_t;

/*
 *  MG_DISCON_REQ
 *  -------------------------------------------------------------------------
 *  Requests that the termination point be disconnected from the cmmunications
 *  session in the directions indicated by mg_conn_flags, and the optional
 *  topology description.
 *
 *  If the optional topology description is not specified, it is assumed that
 *  the termination point is to be disconnected from all other participants in
 *  the communications session for the directions requested.
 *
 *  If the optional topology description is specified, it contains the list of
 *  other termination points in the session from which the specified
 *  termination point is to be disconnected in the directions requested.
 */
typedef struct MG_discon_req {
        mg_ulong mg_primitive;          /* always MG_DISCON_REQ */
        mg_ulong mg_se_id;              /* session id */
        mg_ulong mg_tp_id;              /* termination point */
        mg_ulong mg_conn_flags;         /* connection flags */
        mg_ulong mg_topology_length;    /* length of topology to disconnect */
        mg_ulong mg_topology_offset;    /* offset of topology to disconnect */
} MG_discon_req_t;
```

```
/*
 *  MG_DISCON_IND
 *  -------------------------------------------------------------------------
 *  Indicates that the termination point has been autonomously disconnected
 *  from the indicated communications session in the directions indicated by
 *  mg_conn_flags and with the optional topology description.
 *
 *  If the optional topology description is not indicated, it is assumed that
 *  the termination point has been autonomously disconnected from all other
 *  participants in the communications session for the directions indicated.
 */
typedef struct MG_discon_ind {
        mg_ulong mg_primitive;          /* always MG_DISCON_IND */
        mg_ulong mg_se_id;              /* session id */
        mg_ulong mg_tp_id;              /* disconnecting termination id */
        mg_ulong mg_conn_flags;         /* directions disconnected */
        mg_ulong mg_cause;              /* cause of disconnect */
} MG_discon_ind_t;

/*
 *  MG_DISCON_CON
 *  -------------------------------------------------------------------------
 *  Confirms that the requested disconnection primitive has been successfully
 *  completed.
 */
typedef struct MG_discon_con {
        mg_ulong mg_primitive;          /* always MG_DISCON_CON */
        mg_ulong mg_se_id;              /* session id */
        mg_ulong mg_tp_id;              /* disconnecting termination id */
} MG_discon_con_t;

/*
 *  MG_LEAVE_REQ
 *  -------------------------------------------------------------------------
 *  Requests that the specified termination point (mg_tp_id non-zero) or all
 *  termination points (mg_tp_id zero) leave the specified communication
 *  session.    Once all termination points leave a communications session, the
 *  communication session is destroyed.
 */
typedef struct MG_leave_req {
        mg_ulong mg_primitive;          /* always MG_LEAVE_REQ */
        mg_ulong mg_se_id;              /* session id */
        mg_ulong mg_tp_id;              /* leaving termination id */
} MG_leave_req_t;

/*
 *  MG_LEAVE_IND
 *  -------------------------------------------------------------------------
 *  Indicates that the termination point has autonomously left the
 *  communications session.
 */
typedef struct MG_leave_ind {
        mg_ulong mg_primitive;          /* always MG_LEAVE_IND */
        mg_ulong mg_se_id;              /* session id */
        mg_ulong mg_tp_id;              /* leaving termination id */
        mg_ulong mg_cause;              /* reason for leaving */
```

```
} MG_leave_ind_t;

/*
 *  MG_LEAVE_CON
 *  -------------------------------------------------------------------------
 *  Confirms that the termination point has left the session.    The termination
 *  point identifier is released.
 */
typedef struct MG_leave_con {
        mg_ulong mg_primitive;          /* always MG_LEAVE_CON */
        mg_ulong mg_se_id;              /* session id */
        mg_ulong mg_tp_id;              /* left termination id */
} MG_leave_con_t;

/*
 *  MG_OK_ACK
 *  -------------------------------------------------------------------------
 */
typedef struct MG_ok_ack {
        mg_ulong mg_primitive;          /* always MG_OK_ACK */
        mg_ulong mg_correct_prim;       /* correct primitive */
} MG_ok_ack_t;

/*
 *  MG_ERROR_ACK
 *  -------------------------------------------------------------------------
 */
typedef struct MG_error_ack {
        mg_ulong mg_primitive;          /* always MG_INFO_REQ */
        mg_ulong mg_error_primitive;    /* primitive in error */
        mg_ulong mg_error_type;         /* MG interface error type */
        mg_ulong mg_unix_error;         /* UNIX error */
} MG_error_ack_t;

#define MGSYSERR                0       /* UNIX system error */
#define MGOUTSTATE              1       /* Interface out of state */
#define MGBADPRIM               2       /* Bad primitive */
#define MGNOTSUPP               3       /* Primitive not supported */
#define MGBADID                 4       /* Bad identifier */
#define MGBADOPTTYPE            5       /* Bad options structure type */
#define MGBADOPT                6       /* Bad option format or content */
#define MGBADFLAG               7       /* Bad flag */
#define MGIDBUSY                8       /* Object busy */
#define MGBADACT                9       /* Bad action */

/*
 *  MG_NOTIFY_REQ
 *  -------------------------------------------------------------------------
 */
typedef struct MG_notify_req {
        mg_ulong mg_primitive;          /* always MG_NOTIFY_REQ */
        mg_ulong mg_events;             /* events to notify */
} MG_notify_req_t;

/*
 *  MG_NOTIFY_IND
```

```
 *  -------------------------------------------------------------------------
 */
typedef struct MG_notify_ind {
        mg_ulong mg_primitive;          /* always MG_NOTIFY_IND */
        mg_ulong mg_event;              /* event */
        mg_ulong mg_se_id;              /* session id */
        mg_ulong mg_tp_id;              /* termination id */
        mg_ulong mg_ch_id;              /* channel id */
        mg_ulong mg_diag_length;        /* diagnostic length */
        mg_ulong mg_diag_offset;        /* diagnostic offset */
} MG_notify_ind_t;

#endif                          /* __SS7_MG_H__ */
```

C.2 MGI Input-Output Controls Header File Listing

```
#ifndef __SS7_MGI_IOCTL_H__
#define __SS7_MGI_IOCTL_H__

#include <linux/ioctl.h>

#define MG_IOC_MAGIC    'G'

#define MG_OBJ_TYPE_MG          1       /* media gateway control */
#define MG_OBJ_TYPE_SE          2       /* communications session */
#define MG_OBJ_TYPE_LG          3       /* connection leg */
#define MG_OBJ_TYPE_CH          4       /* channel */
#define MG_OBJ_TYPE_MX          5       /* multiplex */
#define MG_OBJ_TYPE_DF          6       /* default */

/*
 *  Media Gateway options
 *  -------------------------------------
 */
typedef struct mg_opt_conf_mg {
        mg_ulong type;                  /* media type */
        mg_ulong flags;                 /* options flags */
        mg_ulong encoding;              /* encoding */
        mg_ulong block_size;            /* data block size (bits) */
        mg_ulong sample_size;           /* sample size (bits) */
        mg_ulong samples;               /* samples per block */
        mg_ulong rate;                  /* clock rate (samples/second) */
        mg_ulong tx_channels;           /* number of tx channels */
        mg_ulong rx_channels;           /* number of rx channels */
} mg_opt_conf_mg_t;

/*
 *  Multiplex options
 *  ---------------------------------
 *  A multiplex is an upper or lower stream.
 *
 *  This specifies the media characteristics of a simple multiplex.    This is
 *  typically a read-only options structure.    To change media
 *  characteristics, change the characteristics of a channel a multiplex
 *  normally requires that the lower stream be unlinked an relinked or the
```

```
 *  pass-thru ioctls performed.
 */
typedef struct mg_opt_conf_mx {
        mg_ulong type;                  /* media type */
        mg_ulong flags;                 /* options flags */
        mg_ulong encoding;              /* encoding */
        mg_ulong block_size;            /* data block size (bits) */
        mg_ulong sample_size;           /* sample size (bits) */
        mg_ulong samples;               /* samples per block */
        mg_ulong rate;                  /* clock rate (samples/second) */
        mg_ulong tx_channels;           /* number of tx channels */
        mg_ulong rx_channels;           /* number of rx channels */
} mg_opt_conf_mx_t;

/*
 *  Channel options
 *  -----------------------------------
 *  A channel is a slot in a media stream.
 *
 *  This specifies the media characteristics of the channel.    The number of
 *  tx_channels and rx_channels must be 1 or 0.
 */
typedef struct mg_opt_conf_ch {
        mg_ulong type;                  /* media type */
        mg_ulong flags;                 /* options flags */
        mg_ulong encoding;              /* encoding */
        mg_ulong block_size;            /* data block size (bits) */
        mg_ulong sample_size;           /* sample size (bits) */
        mg_ulong samples;               /* samples per block */
        mg_ulong rate;                  /* clock rate (samples/second) */
        mg_ulong tx_channels;           /* number of tx channels */
        mg_ulong rx_channels;           /* number of rx channels */
} mg_opt_conf_ch_t;

/*
 *  Connection Leg options
 *  -----------------------------------
 *  A connection leg is a collection of channels that act together towards
 *  a termination that are normally controlled as a group.    An example would
 *  be multiple 64kbps channels making up a multi-rate call.    Channels must be
 *  added to connection legs before they can be used.    Channels that are
 *  not engaged in a call can remain attached to their connection legs for
 *  their entire life-cycle, or can be rearranged by the media gateway.
 *
 *  Connection legs have their own characteristics.    Only connection legs
 *  of like characteristics to a communications session can be associated with
 *  a communications session.
 *
 *  Connection legs do not necessarily have the same media characteristics
 *  as the channels that make up the termination point.    The connection leg
 *  object performs media conversion between the channel media type and the
 *  conenction leg.
 *
 *  Channels which have multiple embedded Tx and Rx channels can be associated
 *  with more than one Connection Leg.
 *
```

```
 *  When an channel is added to a communications session, connection leg
 *  objects may be dynamically  created that will adapt between the media
 *  type of the communications session and the media type of the channel.
 *  In that case, commnection leg characteristics are read-only.
 */
typedef struct mg_opt_conf_lg {
        mg_ulong type;                  /* media type */
        mg_ulong flags;                 /* options flags */
        mg_ulong encoding;              /* encoding */
        mg_ulong block_size;            /* data block size (bits) */
        mg_ulong sample_size;           /* sample size (bits) */
        mg_ulong samples;               /* samples per block */
        mg_ulong rate;                  /* clock rate (samples/second) */
        mg_ulong tx_channels;           /* number of tx channels */
        mg_ulong rx_channels;           /* number of rx channels */
} mg_opt_conf_lg_t;

/*
 *  Communications Session options
 *  ----------------------------------
 *  A communications session is a collection of connection legs engaged in a
 *  communications session.    A communications session has its own media
 *  characteristics.    Only terminations points of like characteristics can be
 *  associated in a communications session.    Comunications sessions do not do
 *  media conversion between connection legs.    Connection legs are responsible
 *  for performing any necessary media conversion between the communications
 *  session and the channel.
 *
 *  Not all communications sessions necessarily support all media types.    For
 *  example, communications sessions which support conferencing might only
 *  support 16-bit linear PCM coded voice.    The reason being that conferencing
 *  calculations need not perform conversion.
 *
 *  The media characteristics refer the to media characteristics of a
 *  participant in the communications session as supported by the
 *  participating connnection legs.
 *
 *  Communications session are by dynamically created by adding channels or
 *  connection legs to the NULL communications session.    These configuration
 *  options are normall read-only.
 */
typedef struct mg_opt_conf_se {
        mg_ulong type;                  /* media type */
        mg_ulong flags;                 /* options flags */
        mg_ulong encoding;              /* encoding */
        mg_ulong block_size;            /* data block size (bits) */
        mg_ulong sample_size;           /* sample size (bits) */
        mg_ulong samples;               /* samples per block */
        mg_ulong rate;                  /* clock rate (samples/second) */
        mg_ulong tx_channels;           /* number of tx channels */
        mg_ulong rx_channels;           /* number of rx channels */
} mg_opt_conf_se_t;

/*
 *  Default options
 *  ----------------------------------
```

```
 *  When channels, connection legs and communications sessions are newly
 *  created and can support different media types, the default media type will
 *  be initially assigned.
 */
typedef struct mg_opt_conf_df {
        mg_ulong type;                  /* media type */
        mg_ulong flags;                 /* options flags */
        mg_ulong encoding;              /* encoding */
        mg_ulong block_size;            /* data block size (bits) */
        mg_ulong sample_size;           /* sample size (bits) */
        mg_ulong samples;               /* samples per block */
        mg_ulong rate;                  /* clock rate (samples/second) */
        mg_ulong tx_channels;           /* number of tx channels */
        mg_ulong rx_channels;           /* number of rx channels */
} mg_opt_conf_df_t;

/*
 *  OPTIONS
 */
typedef struct mg_option {
        mg_ulong type;
        mg_ulong id;
        /* followed by specific object options structure */
} mg_option_t;

#define MG_IOCGOPTIONS  _IOWR(  MG_IOC_MAGIC,    0,     mg_option_t     )
#define MG_IOCSOPTIONS  _IOWR(  MG_IOC_MAGIC,    1,     mg_option_t     )

/*
 *  Media Gateway configuration
 *  -----------------------------------
 */
typedef struct mg_conf_mg {
} mg_conf_mg_t;

/*
 *  Multiplex configuration
 *  -----------------------------------
 */
typedef struct mg_conf_mx {
        mg_ulong muxid;                 /* lower multiplexing driver id */
} mg_conf_mx_t;

/*
 *  Channel configuration
 *  -----------------------------------
 */
typedef struct mg_conf_ch {
        mg_ulong tpid;                  /* termination point id */
        mg_ulong mxid;                  /* multiplex id */
        mg_ulong slot;                  /* slot in multiplex */
        mg_ulong encoding;              /* channel encoding */
} mg_conf_ch_t;

/*
 *  Connection leg configuration
```

```
 *  ---------------------------------
 *  Connection leg object are not created statically.
 */
typedef struct mg_conf_lg {
        mg_ulong seid;                  /* session id */
} mg_conf_lg_t;

/*
 *  Communications Session configuration
 *  ---------------------------------
 *  Communications session objects are not created statically.
 */
typedef struct mg_conf_se {
} mg_conf_se_t;

/*
 *  Default configuration
 *  ---------------------------------
 *  The default object is not created statically.
 */
typedef struct mg_conf_df {
} mg_conf_df_t;

/*
 *  CONFIGURATION
 */
typedef struct mg_config {
        mg_ulong type;                  /* object type */
        mg_ulong id;                    /* object id */
        mg_ulong cmd;                   /* configuration command */
        /* followed by specific configuration structure */
} mg_config_t;

#define MG_GET          0       /* get configuration */
#define MG_ADD          1       /* add configuration */
#define MG_CHA          2       /* cha configuration */
#define MG_DEL          3       /* del configuration */

#define MG_IOCGCONFIG   _IOWR(  MG_IOC_MAGIC,   2,      mg_config_t     )
#define MG_IOCSCONFIG   _IOWR(  MG_IOC_MAGIC,   3,      mg_config_t     )
#define MG_IOCTCONFIG   _IOWR(  MG_IOC_MAGIC,   4,      mg_config_t     )
#define MG_IOCCCONFIG   _IOWR(  MG_IOC_MAGIC,   5,      mg_config_t     )

/*
 *  Multiplex state
 *  ---------------------------------
 */
typedef struct mg_timers_mx {
} mg_timers_mx_t;
typedef struct mg_statem_mx {
        struct mg_timers_mx timers;
        /* followed by the channel associations */
        struct {
                mg_ulong slot;          /* slot number */
                mg_ulong chid;          /* channel id */
        } slot[0];
```

```
} mg_statem_mx_t;

/*
 *  Channel state
 *  -----------------------------------
 */
typedef struct mg_timers_ch {
} mg_timers_ch_t;
typedef struct mg_statem_ch {
        struct mg_timers_ch timers;
        mg_ulong mxid;
} mg_statem_ch_t;

/*
 *  Termination Point state
 *  -----------------------------------
 */
typedef struct mg_timers_lg {
} mg_timers_lg_t;
typedef struct mg_statem_lg {
        struct mg_timers_lg timers;
        mg_ulong mxid;
} mg_statem_lg_t;

/*
 *  Communications Session state
 *  -----------------------------------
 */
typedef struct mg_timers_se {
} mg_timers_se_t;
typedef struct mg_statem_se {
        struct mg_timers_se timers;
        /* followed by the connection leg participation */
        struct {
                mg_ulong role;          /* participant role */
                mg_ulong flags;         /* topology flags */
                mg_ulong lgid;          /* connection leg id */
        } leg[0];
} mg_statem_se_t;

/*
 *  Media Gateway state
 *  -----------------------------------
 */
typedef struct mg_timers_mg {
} mg_timers_mg_t;
typedef struct mg_statem_mg {
        struct mg_timers_mg timers;
} mg_statem_mg_t;

/*
 *  Default state
 *  -----------------------------------
 */
typedef struct mg_timers_df {
} mg_timers_df_t;
```

```
typedef struct mg_statem_df {
        struct mg_timers_df timers;
        /* followed by a list of connection sessions */
        struct {
                mg_ulong seid;          /* session identifier */
        } sessions[0];
} mg_statem_df_t;

/*
 *  STATE
 */
typedef struct mg_statem {
        mg_ulong type;                  /* object type */
        mg_ulong id;                    /* object id */
        mg_ulong flags;                 /* object flags */
        mg_ulong state;                 /* object state */
        /* followed by object-specific state structure */
} mg_statem_t;

#define MG_IOCGSTATEM   _IOR(   MG_IOC_MAGIC,   6,      mg_statem_t     )
#define MG_IOCCMRESET   _IOR(   MG_IOC_MAGIC,   7,      mg_statem_t     )

/*
 *  Media Gateway statistics
 *  -----------------------------------
 */
typedef struct mg_stats_mg {
} mg_stats_mg_t;

/*
 *  Multiplex statistics
 *  -----------------------------------
 */
typedef struct mg_stats_mx {
} mg_stats_mx_t;

/*
 *  Channel statistics
 *  -----------------------------------
 */
typedef struct mg_stats_ch {
} mg_stats_ch_t;

/*
 *  Connection Leg statistics
 *  -----------------------------------
 */
typedef struct mg_stats_lg {
} mg_stats_lg_t;

/*
 *  Communications Session statistics
 *  -----------------------------------
 */
typedef struct mg_stats_se {
} mg_stats_se_t;
```

```
/*
 *  Default statistics
 *  ----------------------------------
 */
typedef struct mg_stats_df {
} mg_stats_df_t;

/*
 *  STATISTICS
 */
typedef struct mg_stats {
        mg_ulong type;                  /* object type */
        mg_ulong id;                    /* object id */
        /* followed by object-specific statistics type */
} mg_stats_t;

#define MG_IOCGSTATSP   _IOR(   MG_IOC_MAGIC,    8,     mg_stats_t      )
#define MG_IOCSSTATSP   _IOWR(  MG_IOC_MAGIC,    9,     mg_stats_t      )
#define MG_IOCGSTATS    _IOR(   MG_IOC_MAGIC,   10,     mg_stats_t      )
#define MG_IOCCSTATS    _IOW(   MG_IOC_MAGIC,   11,     mg_stats_t      )

/*
 *  Media Gateway notifications
 *  ----------------------------------
 */
typedef struct mg_notify_mg {
        mg_ulong events;
} mg_notify_mg_t;

/*
 *  Multiplex notifications
 *  ----------------------------------
 */
typedef struct mg_notify_mx {
        mg_ulong events;
} mg_notify_mx_t;

/*
 *  Channel notifications
 *  ----------------------------------
 */
typedef struct mg_notify_ch {
        mg_ulong events;
} mg_notify_ch_t;

/*
 *  Connection Leg notifications
 *  ----------------------------------
 */
typedef struct mg_notify_lg {
        mg_ulong events;
} mg_notify_lg_t;

/*
 *  Communications Session notifications
```

```
 *  ---------------------------------
 */
typedef struct mg_notify_se {
        mg_ulong events;
} mg_notify_se_t;

/*
 *  Default notifications
 *  ---------------------------------
 */
typedef struct mg_notify_df {
        mg_ulong events;
} mg_notify_df_t;

/*
 *  EVENTS
 */
typedef struct mg_notify {
        mg_ulong type;                  /* object type */
        mg_ulong id;                    /* object id */
        /* followed by object-specific notification type */
} mg_notify_t;

#define MG_IOCGNOTIFY   _IOR(   MG_IOC_MAGIC,   12,     mg_notify_t     )
#define MG_IOCSNOTIFY   _IOW(   MG_IOC_MAGIC,   13,     mg_notify_t     )
#define MG_IOCCNOTIFY   _IOW(   MG_IOC_MAGIC,   14,     mg_notify_t     )

/*
 *  MG MANAGEMENT
 */
typedef struct mg_mgmt {
        mg_ulong type;                  /* object type */
        mg_ulong id;                    /* object id */
        mg_ulong cmd;                   /* mgmt command */
} mg_mgmt_t;

#define MG_MGMT_BLOCK       1
#define MG_MGMT_UNBLOCK     2
#define MG_MGMT_RESET       3
#define MG_MGMT_QUERY       4

#define MG_IOCCMGMT     _IOWR(  MG_IOC_MAGIC,   15,     mg_mgmt_t       )

/*
 *  PASS LOWER
 */
typedef struct mg_pass {
        mg_ulong muxid;                 /* mux index of lower CH structure to pass message to */
        mg_ulong type;                  /* type of message block */
        mg_ulong band;                  /* band of message block */
        mg_ulong ctl_length;            /* length of cntl part */
        mg_ulong dat_length;            /* length of data part */
        /* followed by cntl and data part of message to pass to to channel */
} mg_pass_t;

#define MG_IOCCPASS     _IOWR(  MG_IOC_MAGIC,   16,     mg_pass_t       )
```

```
#define MG_IOC_FIRST     0
#define MG_IOC_LAST     16
#define MG_IOC_PRIVATE  32

#endif                          /* __SS7_MGI_IOCTL_H__ */
```

Appendix D MGI Drivers and Modules

There are a number of standard drivers and modules provided by the *OpenSS7 Project* the provide capabilities uilizing the Media Gateway Interface.

D.1 MGI Drivers

Drivers that provide the MGI interace fall into two categories:

D.1.1 MGI Pseudo-device Drivers

Pseudo-device drivers that accept or provide the MGI interface for the purpose of providing or controlling access the multiplexed facilities available on a system.

D.1.1.1 Media Gateway Driver—`mg`

The `mg` driver is a pseudo-device multiplexing driver that provides the *Media Gateway Interface* (MGI) at its upper service interface and accepts the *Channel Interface* (CHI) at its lower service interface. This driver provides the MGI service interface to directly control *Media Gateway* (MG) functions. Its purpose is to control the *Switch Matrix Multiplexing Driver* (`matrix`) beneath it and connected using CHI streams, and to perform media marshalling, conversion, tones, announcements and termination functions of a *Media Gateway*.

Due to the intensive nature of media handling and conversion and desire for wide scalability, the functions such as those performed by this driver (that are in the media stream) are best performed by STREAMS modules or drivers.[1]

The *Media Gateway Controller* for this MG function can either be local to the same host, or remote, using the services of the *H.248 Media Gateway (MG) Driver*, `h248-mg`.

D.1.1.2 H.248 Media Gateway Controller (MGC) Driver–`h248-mgc`

The `h248-mgc` driver is peudo-device multiplexing driver that provides the MGI interface at its upper service interface and accepts the NPI or TPI interface at its lower service interface. This driver performs the conversion of the MGI service interface to the H.248 protocol carried on the lower transport streams. Its purpose is to implement the *Media Gateway Controller* (MGC) side of the MGC-MG communications as described in the *ITU-T Recommendation H.248* and equivalent IETF RFCs. It could support both H.248 (MEGACO) and MGCP.

The `h248-mgc` driver can be used to control a remote MG. It would normally be used by a *Media Gateway Controller* (MGC) or softswitch.

In general, it is not necessary for the conversion between the H.248 or MGCP protocol and the services of the MGI to be performed by a STREAMS module. This is because the protocol is only involved in the setup and tear-down of media connections, and is not directly involved in the media path. Therefore, a user-space application is quite capable of performing the same functions using the normal user-space services: XTI or Sockets for accessing the transport streams.

D.1.1.3 H.248 Media Gateway (MG) Driver—`h248-mg`

The `h248-mg` driver is a pseudo-device multiplexing driver that provides the MGI interface at its lower service interface and also accepts the NPI or TPI interface at its lower service interface. This driver performs the conversion of the MGI service interface to and from the H.248 protocol carried on

[1] This is due to the superior scheduling and throughput performance exhibited by STREAMS modules and drivers.

the transport streams. Its purpose is to implement the *Media Gateway* (MG) side of the MGC-MG communications as described in the *ITU-T Recommendation H.248* and equivalent IETF RFCs. It could support both H.248 (MEGACO) and MGCP.

In general, it is not necessary for the conversion between H.248 and MGCP protocol and the services of the MGI to be performed by a STREAMS module. This is because the protocol is only involved in the setup and tear-down of media connections, and is not directly involved in the media path. Therefore, a user-space application is quite capable of performing the same functions using the normal user-space services: XTI or Sockets for accessing the transport streams, and the MGI for accessing the MG driver (which is kernel resident and directly involved in handling media).

D.1.2 MGI Device Drivers

There are currently no real device drivers that provide the MGI interface.

D.2 MGI Modules

STREAMS pushable modules are an excellent way of adapting a MGS user Stream that conforms to the general concept of a communications media gateway into a complex communications protocol.

There are currently no pushable modules implementing the MGI.

Appendix E MGI Applications

The media gateway interface is a rather important upper layer component of a number of *OpenSS7 Project* protocol and media stacks.

E.1 MGI in MGC Stack

As illustrated in Figure E.1, the MGI interface provides support for the remote control of a *Media Gateway* (MG) using several related protocols used for that purpose. The protocols are *MGCP* and *MEGACO*[1].

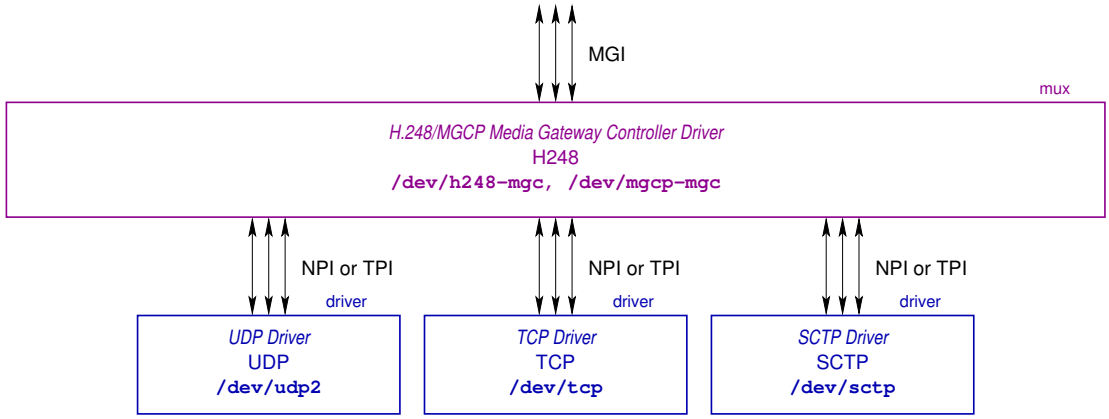

Figure E.1: *Media Gateway Controller Stack*

The MGI interface is responsible for providing access to media gateway services necessary for the *Media Gateway Controller* to control the functions of the *Media Gateway*. Use of the MGI and the *OpenSS7* media gateway controller MGCP/MEGACO component provides a mechanism whereby any media gateway, whether local or remote, can be controlled by an MGC as a MGS user of the MGI.

The stack also provides the capability of providing for redundancy of remote media gateways, or redundant configuration with local and remote media gateways.

Note that, because the H.248/MEGACO and MGCP protocols are only used for setup and tear-down of media connections, the traffic is not intensive under normal operation. This makes H.248/MEGACO and MGCP possible to be implemented in user-space applications using regular application approaches. However, there is one instance where H.248 requires extensive messaging: when performing audits on large media gateways. In these circumstances audits on 10's to 100's of thousand circuits might be required in an extremely short interval. For ISUP applications, this may particularly be true when adjacent switches with trunks groups with 10's of thousands of circuits restart.

Also note that, in an integrated MGC-MG product such as the *Optranex 248*, use of H.248 might not be necessary.

[1] *ITU-T Recommendation H.248.*

E.2 MGI in MG Stack

As illustrated in Figure E.2, the MGI interface provides support for both the remote control by a *Media Gateway Controller* (MGC) of the local MG functions, as well as providing an interface for interaction using IP transport and the *MGCP* or *MEGACO* protocols.

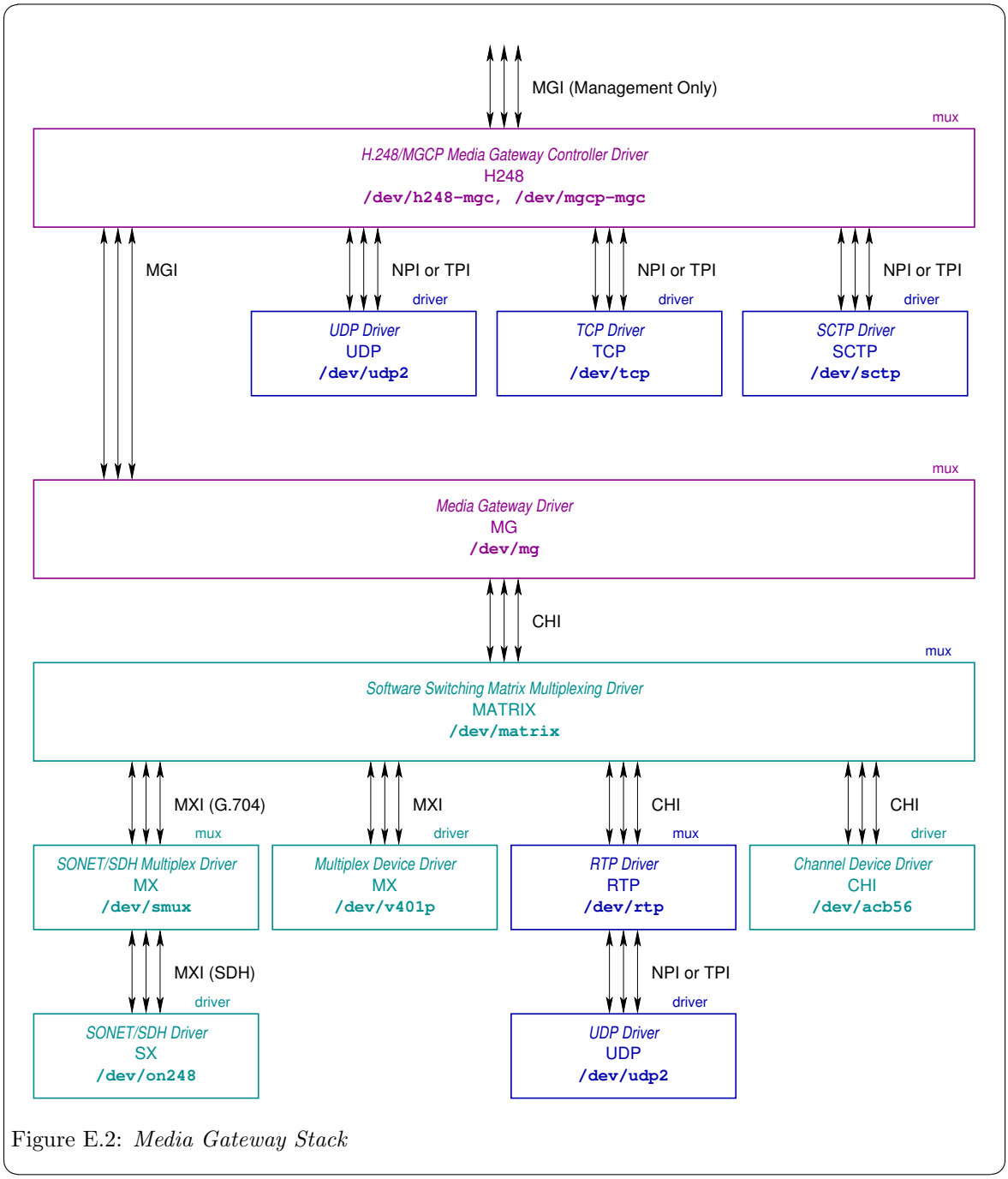

Figure E.2: *Media Gateway Stack*

The MGI interface is responsible both for providing a service primitive interface that is capable of performing the functions of the H.248/MEGACO and MGCP protocols, as well as providing for the control of the media functions of the media gatway itself. This is illustrated by the two multiplexing drivers, `h248-mgc` and `mg`, illustrated in Figure E.2.

Appendix F MGI Utilities

No MGI-specific utilities are currently provided.

Appendix G MGI File Formats

No MGI-specific file formats are currently defined.

Appendix H MGI Compatibility and Porting

The *Media Gateway Interface* (MGI) is a service interface and API that is defined by the *OpenSS7 Project* for use with *OpenSS7* modules, drivers and applications programs and has not (to our knowledge) been implemented by others. The interface uses standard STREAMS and *POSIX/SUSv3* facilities. As such, there are no compatibility or porting issues associated with the interface.

Glossary

Media Gateway Service Data Unit
: A grouping of MGS 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 media gateway service users.

MGS provider
: The media gateway control layer protocol that provides the services of the media gateway interface.

MGS user
: The user-level application or user-level or kernel-level protocol that accesses the services of the media gateway layer.

Local management
: The phase in connection and connectionless modes in which a MGS 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

ANSI	American National Standards Institute
ETSI	European Telecommunications Standards Institute
GCP	Gateway Control Protocol
H.248	ITU-T Recommendation H.248
ITU-T	International Telecommunications Union - Telecom Sector
LMS Provider	A provider of Local Management Services
LMS	Local Management Service
LMS User	A user of Local Management Services
LM	Local Management
LSC	Link State Control
MEGACO	Media Gateway Control Protocol
MGCP	Media Gateway Control Protocol
MGC	Media Gateway Controller
MGSP	MGS Provider
MGS	Media Gateway Service
MGSU	MGS User
MG	Media Gateway
PPA	Physical Point of Attachment
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 137. The text of this manual is licensed under the [GNU Free Documentation License], page 147, 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

O

P

R

S

T

U