



Dialogic[®] DSI Signaling Software
Sigtran Monitor Programmer's Manual

November 2011 U06STN05

www.dialogic.com

Copyright and Legal Notice

Copyright © 2007-2011 Dialogic Inc. All Rights Reserved. You may not reproduce this document in whole or in part without permission in writing from Dialogic Inc. at the address provided below.

All contents of this document are furnished for informational use only and are subject to change without notice and do not represent a commitment on the part of Dialogic Inc. and its affiliates or subsidiaries ("Dialogic"). Reasonable effort is made to ensure the accuracy of the information contained in the document. However, Dialogic does not warrant the accuracy of this information and cannot accept responsibility for errors, inaccuracies or omissions that may be contained in this document.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH DIALOGIC® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN A SIGNED AGREEMENT BETWEEN YOU AND DIALOGIC, DIALOGIC ASSUMES NO LIABILITY WHATSOEVER, AND DIALOGIC DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF DIALOGIC PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT OF A THIRD PARTY.

Dialogic products are not intended for use in certain safety-affecting situations. Please see <http://www.dialogic.com/company/terms-of-use.aspx> for more details.

Due to differing national regulations and approval requirements, certain Dialogic products may be suitable for use only in specific countries, and thus may not function properly in other countries. You are responsible for ensuring that your use of such products occurs only in the countries where such use is suitable. For information on specific products, contact Dialogic Inc. at the address indicated below or on the web at www.dialogic.com.

It is possible that the use or implementation of any one of the concepts, applications, or ideas described in this document, in marketing collateral produced by or on web pages maintained by Dialogic may infringe one or more patents or other intellectual property rights owned by third parties. Dialogic does not provide any intellectual property licenses with the sale of Dialogic products other than a license to use such product in accordance with intellectual property owned or validly licensed by Dialogic and no such licenses are provided except pursuant to a signed agreement with Dialogic. More detailed information about such intellectual property is available from Dialogic's legal department at 1504 McCarthy Boulevard, Milpitas, CA 95035-7405 USA. **Dialogic encourages all users of its products to procure all necessary intellectual property licenses required to implement any concepts or applications and does not condone or encourage any intellectual property infringement and disclaims any responsibility related thereto. These intellectual property licenses may differ from country to country and it is the responsibility of those who develop the concepts or applications to be aware of and comply with different national license requirements.**

Dialogic, Dialogic Pro, Dialogic Blue, Veraz, Brooktrout, Diva, Diva ISDN, Making Innovation Thrive, Video is the New Voice, VisionVideo, Diastar, Cantata, TruFax, SwitchKit, SnowShore, Eicon, Eiconcard, NMS Communications, NMS (stylized), SIPcontrol, Exnet, EXS, Vision, PowerMedia, PacketMedia, BorderNet, inCloud9, I-Gate, ControlSwitch, NaturalAccess, NaturalCallControl, NaturalConference, NaturalFax and Shiva, among others as well as related logos, are either registered trademarks or trademarks of Dialogic Inc. and its affiliates or subsidiaries. Dialogic's trademarks may be used publicly only with permission from Dialogic. Such permission may only be granted by Dialogic's legal department at 1504 McCarthy Boulevard, Milpitas, CA 95035-7405 USA. Any authorized use of Dialogic's trademarks will be subject to full respect of the trademark guidelines published by Dialogic from time to time and any use of Dialogic's trademarks requires proper acknowledgement.

The names of actual companies and products mentioned herein are the trademarks of their respective owners.

This document discusses one or more open source products, systems and/or releases. Dialogic is not responsible for your decision to use open source in connection with Dialogic products (including without limitation those referred to herein), nor is Dialogic responsible for any present or future effects such usage might have, including without limitation effects on your products, your business, or your intellectual property rights.

Publication Date: November 2011

Document Number: U06STN, Issue 5

Revision History

Issue	Date	Description
5	10-Nov-11	Support for Virtual Ports added allowing source messages to be delivered by MSG to MST. New module status and port statistics messages added. New diagnostic capabilities added for tracing messages and generation of selective tracing in the event of key protocol events.
4	27-Apr-09	Support for Windows operating system
3	17-Jul-08	Support for filtering of retransmitted SCTP DATA chunks. Support for in-sequence delivery of SCTP DATA chunks.
2	20-Jun-07	Swapped order of FSN and BSN in MST_MSG_M2PA_MSG. Changed units used in the Throughput Statistics message. Corrected PRTOPT_PROMISCUOUS definition. Other minor updates prior to first general release.
1	26-Mar-07	

Contents

1	Introduction	6
1.1	Applicability	6
1.2	Related Documentation	6
2	Functional Overview	7
2.1	Message Source	7
2.1.1	Using an Ethernet Hub	8
2.1.2	Using a Managed Switch.....	8
2.1.3	Using Virtual Ports.....	9
2.2	Monitoring Message Selection.....	9
2.2.1	Monitoring Message Presentation.....	10
2.2.2	Filtering of duplicated DATA chunks	10
2.2.3	Ordered delivery of DATA chunks	10
2.3	Throughput and Licensing	11
2.3.1	Flow Control	11
2.3.2	Throughput Capability	11
2.4	Operation, Administration and Maintenance	11
2.4.1	Statistics	11
2.4.2	Tracing	11
3	Installation and Configuration	12
3.1	Binary Installation	12
3.2	License File Installation	12
3.3	Configuration	12
4	Message Reference.....	17
4.1	Overview	17
4.1.1	Message Summary Table.....	17
4.1.2	Status Code Values.....	18
4.2	Configuration Messages.....	19
4.2.1	MST_MSG_CONFIG - MST Module Configuration	19
4.2.2	MST_MSG_CFG_PORT - MST Port Configuration	21
4.2.3	MST_MSG_CFG_TAP - MST TAP Configuration	23
4.2.4	MST_MSG_CFG_TRANSPORT_ADDR - Transport Address Configuration	26
4.2.5	MST_MSG_CFG_ASSOCIATION - Association Configuration.....	28
4.2.6	MGT_MSG_TRACE_MASK - Trace Mask Configuration.....	30
4.2.7	MST_MSG_SELTRACE_MASK - Selective Trace Mask Configuration	33
4.3	Payload Indication Messages	35
4.3.1	MST_MSG_IP_MSG - IP Payload Indication	35
4.3.2	MST_MSG_M3UA_MSG - M3UA Payload Indication.....	36
4.3.3	MST_MSG_M2PA_MSG - M2PA Payload Indication	37
4.4	State and Statistics Request Messages	39
4.4.1	MST_MSG_R_STATS - MST Module Statistics Request	39
4.4.2	MST_MSG_R_PORT_STATS - MST Port Statistics Request	41
4.4.3	MST_MSG_R_TAP_STATS - MST TAP Statistics Request.....	43
4.4.4	MST_MSG_R_ASSOCIATION_STATS - MST Association Statistics Request	44
4.4.5	MGT_MSG_R_THR_STATS - Throughput Statistics Request.....	47
4.4.6	MGT_MSG_R_LIC_STATUS - MST Licensing State Request.....	49
4.5	Event Indication Messages	51
4.5.1	MST_MSG_FLCT_EVENT_IND - MST Flow Control Event Indication.....	51
4.5.2	MST_MSG_EVENT_IND - MST Management Event Indication.....	52
4.5.3	MGT_MSG_LIC_EVENT - License Event Indication.....	54
4.5.4	MST_MSG_IP_TRACE - MST Input Packet Trace Message.....	55
4.6	Traffic Input Messages	56

4.6.1 MST_MSG_VPORT_PKT - Virtual Port Traffic Request 56

1 Introduction

The Dialogic® DSI Signaling Software – Sigtran Monitoring Binary (MST) allows live monitoring of SS7 traffic from Sigtran SCTP Associations running over Ethernet. SCTP Messages from one or more Ethernet ports can be selectively passed to a User Application in real time. Messages may be presented in a choice of formats depending on the needs of the application.

The Sigtran Monitoring software is available for Linux, Solaris (SPARC and x86) and Windows Operating Systems. It uses the same message-based inter process communication mechanism used by all other Dialogic® DSI Signaling Software.

This Programmer's Manual explains the capabilities of the MST module, how to install, configure and run the MST software on a server. It includes a reference of all messages used to interface to the module, and an example of the module's use.

1.1 Applicability

This manual is applicable to the the MST Sigtran monitoring software which is contained within the following software releases:

Dialogic® DSI Development Package for Linux – Release 6.3.8 or later

Dialogic® DSI Development Package for Solaris – Release 5.1.6 or later

Dialogic® DSI Development Package for Windows – Release 6.2.3 or later

1.2 Related Documentation

Current software and documentation supporting Dialogic® DSI products is available at:

<http://www.dialogic.com/support/helpweb/signaling>

The following user documentation is applicable:

[1] U10SSS: Software Environment Programmer's Manual

2 Functional Overview

The Sigtran Monitoring software (MST) captures packets from one or more Ethernet ports and applies various filters and post processing prior to passing selected packets to the User Application. In addition the module supports the concept of a 'Virtual Port' where instead of capturing packets from an Ethernet port, the source packets are sent by the user to the MST module in messages.

Received messages are optionally filtered to allow any duplicated packets (eg. retransmissions) to be discarded.

A further option allows the user to configure the module for in-sequence delivery of messages. In this mode messages are buffered on a per-association basis and delivered to the user application in the correct sequence (removing any mis-sequencing that may have been introduced by errors, path delays or retransmissions).

Finally the module applies filtering based on Service Indicator (ie User Part), Sigtran message type and source port to determine which messages get passed to each user application.

This chapter describes the functional characteristics of each part of the MST module and how it should be configured to realize a dedicated monitoring application.

2.1 Message Source

There are three ways of providing traffic into the MST module:

- Directly through an Ethernet port connected to an Ethernet hub.
- Directly through an Ethernet port connected to a managed switch.
- Indirectly by sending packets using messages to a 'Virtual Port' within MST.

In each case it is necessary to configure the 'port' using the Port Configuration Message (MST_MSG_CFG_PORT). MST can monitor multiple ports simultaneously.

The Port Configuration Message is defined in Section 4.2.2. This should be sent once for each Port Configured. The 'port_id' is a local reference to the port and the 'device_name' identifies which Ethernet port it relates to, both parameters should be unique (although the device_name should be set to zero for virtual ports).

Ethernet Device names can be found on Linux and Solaris systems by using the 'ifconfig' command from a command prompt with Super User privileges. A typical device name for an Ethernet port is 'eth0' or 'eth1'.

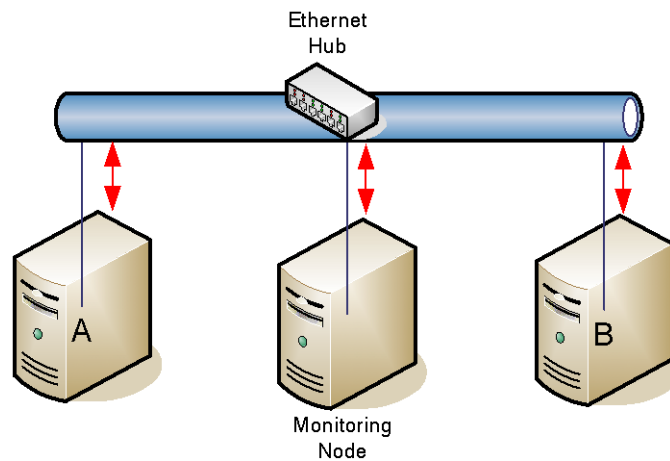
For the Windows operating system an IP address must be assigned to the Ethernet port to be used. The Ethernet port does not have to be in the same network as the monitored network. The device_name under Windows should be set to the hexadecimal representation of the assigned IP address. For example an IP address of 192.168.1.1 will have a device name of c0a80101.

The remainder of this section describes the three port topologies.

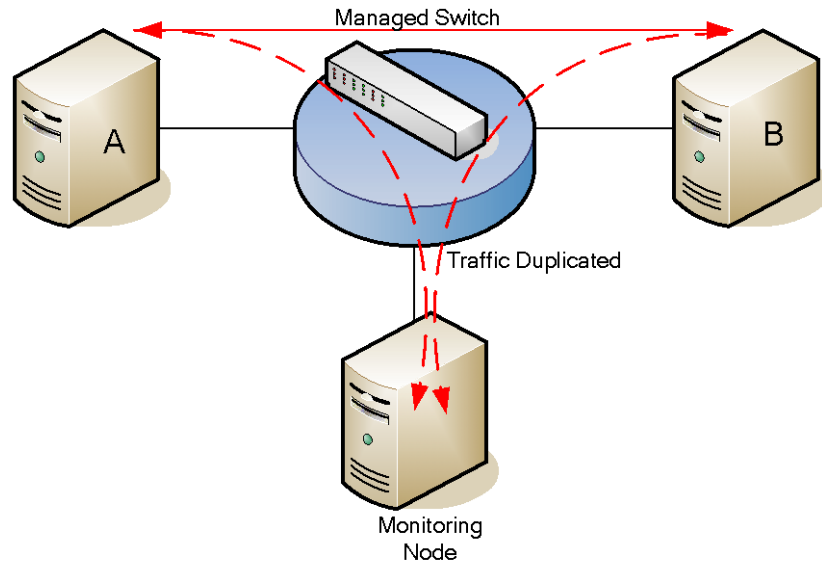
2.1.1 Using an Ethernet Hub

As illustrated below, an Ethernet Hub interconnects all of the Ethernet devices connecting into it so as to make it appear the devices are on the same Ethernet segment. This allows the Monitoring System to see directly the traffic to be monitored.

2.1.2 Using a Managed Switch



Many Managed IP Switches allow packets on one Ethernet Segment to be copied to another Segment. The traffic required to be monitored may be copied onto the Monitoring Systems Ethernet Connections. This allows the number of Ethernet connections brought to the monitoring device to be independent of the number of Ethernet connections on which the traffic actually is distributed across. For example, the traffic to be monitored may be distributed across eight Ethernet cables. These may be brought into a Managed Switch, which, in addition to forward routing the IP packets, may take a copy of the messages and forward them to the Monitoring System across two Ethernet Cables.



2.1.3 Using Virtual Ports

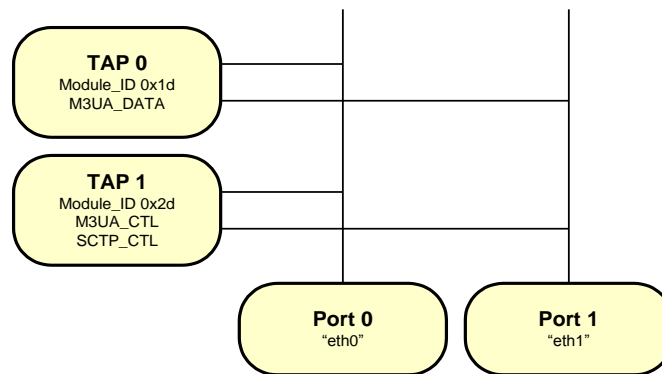
MST supports reception of IP packets from 'Virtual Ports' as opposed to Ethernet ports. IP packets may be received by the system externally to MST and then encapsulated in a `MST_MSG_VPORT_PKT` message and passed to MST for processing. Virtual ports are configured using the Port Configuration message and assigned a `port_id` in the same manner as Ethernet ports. The `port_id` value should also be used in the `MST_MSG_VPORT_PKT` message to identify the virtual port. For Virtual ports the `device_name` parameter is unused and should be set to zero.

2.2 Monitoring Message Selection

Once the Ethernet Ports have been configured, the user must select which types of message are to be sent to the user application. This is achieved by use of the 'Tap' configuration message [MST_MSG_CFG_TAP](#) which allows specific types of messages to be selected and other messages to be discarded. A number of message selections may be made, each as part of a separate 'Tap' configuration. A 'Tap' identifies a message selection and the module ID of the user application to which these messages should be sent. Each monitored message delivered to the user application will include the `tap_id` in the `id` field of the message header. A single system may have a number of configured Taps, each sending messages to a different module ID. Alternatively, each configured Tap can select a different type of message and send the selected messages to the same module ID.

The Tap configuration message allows the user to filter messages based on the Sigtran message type (eg M3UA Data, M2PA Data etc), or the SS7 service indicator (eg ISUP, SCCP etc) or the `port_id` on which the message was received.

Note that the SCTP protocol may combine multiple messages or 'chunks' together into a bundle for transmission over the network in a single Ethernet packet. MST will separate bundled SCTP messages or 'chunks' into their individual parts prior to passing them to the user application.



In addition to selection that is based on message type, 'TAP's can select the specific Ports from which they will receive messages. This is done using a bitmap based on the port_id. It allows messages received from different ports to be differentiated and is of particular use when connecting to different Sigtran networks.

2.2.1 Monitoring Message Presentation

A TAP also identifies the format in which selected messages should be presented to the User Application. This can be either with IP and SIGTRAN headers or without. If it is with IP headers, then the User Application is able to determine more detailed information about where the message has come from, where it is going, whether the message is a duplicate, etc. If it is without these headers, then the application has more easy access to the SS7 User Part data without having to parse these headers.

If a TAP is configured to present messages without IP and SIGTRAN headers, then this prohibits the monitoring of anything but M3UA and M2PA Data messages. Because of the different information contained in an M3UA Data message versus an M2PA Data message, each is presented with a different message type as shown in section 4.3.2 and 4.3.3 respectively.

2.2.2 Filtering of duplicated DATA chunks

SCTP is a reliable transport protocol supporting insequence guaranteed delivery of messages. This requires the retransmission of packets which may appear to the MST application as duplicated messages.

The MST application can be configured to filter out duplicated packets before the monitored messages are reported to a Tap. The filtering only applies to chunks of type DATA.

Configuration messages for these features are described in sections 4.2.4 and 4.2.5.

2.2.3 Ordered delivery of DATA chunks

In addition to filtering duplicate messages MST can also reorder monitored messages to provide in sequence delivery of messages before presentation to monitor applications.

Ordered delivery only applies to chunks of type DATA and is based on the SCTP TSN (Transmission Sequence Number).

Configuration messages for these features are described in sections 4.2.4 and 4.2.5.

2.3 Throughput and Licensing

2.3.1 Flow Control

The MST binary and the User Application use the DSI Software Environment for passing messages to communicate. The number of messages required will vary from system to system; however, the message passing environment should be configured with an adequate number of messages to ensure that there are always sufficient messages. If the message environment should become congested, then the MST process will detect this condition and suspend sending Monitor messages to the User Application until the congestion has abated. During this period messages will be buffered internally.

MST can buffer a limited number of messages; the exact number can be set when the module is first configured. This buffering prevents messages from being discarded if message congestion occurs. Once message congestion abates, the buffered messages will be sent to the User Applications. This buffer will provide a congestion indication when it is half full and a discard indication if it becomes completely full.

2.3.2 Throughput Capability

MST is licensed using throughput based licensing. Each licence supports up to a certain traffic throughput

The throughput is calculated using the length of the SCTP Chunks as identified in the SCTP Chunk header of messages that have been monitored (sent to User Applications).

2.4 Operation, Administration and Maintenance

2.4.1 Statistics

MST supports the gathering of different types of statistics to assist in the maintenance of the system. The statistics are gathered independently for each port_id and tap_id, and for the system as a whole. In addition to this functionality, the throughput licensing scheme used by MST maintains statistics useful for observing license utilisation.

2.4.2 Tracing

In common with other Dialogic® DSI Protocol Stack modules the MST module can trace messages as they are sent or received. It also supports the tracing of IP Packets as received by the module. See section 4.2.6 for the configuration message required to enable this tracing.

3 Installation and Configuration

3.1 Binary Installation

The MST module is supplied as part of the Dialogic® DSI Development Package for Linux, Solaris or Windows. This should be installed as documented in the *Software Environment Programmer's Manual*.

Note: In order to correctly access underlying operating system resources, it is essential that the MST binary has privileges to all it to run with Super User or Administrator privileges. Installation of the DSI Development Package will normally ensure this is the case.

For Linux or Solaris the following commands can be run from an account with root privileges to assign correct privileges:

```
chown root mst
chmod +s mst
```

For Windows the MST binary should be run by an account with administrator privileges.

3.2 License File Installation

When MST is started, it will look to find a valid license file. It will look in the working directory and optionally an additional directory specified as an argument to MST as it is started using the "-Lp" option. The license file should be copied into one of these two directories.

As with other DSI host binaries, MST allows limited operation without a license file when started in trial mode. Trial mode permits the binary to run for one hour after which it will terminate. To start the binary in trial mode spawn the module with the '-t' command line option.

3.3 Configuration

Configuration of an MST system requires the following steps:

- 1) Configuration of a system.txt with the following local tasks:
 - a) TIM (module id 0x00)
 - b) MST (by default module id 0xb1)
 - c) User Applications (Typically one or more of module ids 0x0d, 0x1d, 0x2d ...)
 - d) Optionally Layer Management (Typically module id 0xef)

Each of these requires a declaration of a LOCAL module id and may be automatically spawned by the use of the FORK_PROCESS command.

An example Linux system.txt file is shown in Figure 1 below. In this system there are two monitor applications and a layer management module running at 0xef. Both the User Applications and Layer Management simply have the **s7_log** application running to display any received messages to the screen.

```
* Local Module ID declarations
*
LOCAL    0x00    * TIM module
LOCAL    0xb1    * MST
LOCAL    0xef    * Mngt
LOCAL    0x1d    * User Monitor app 1
LOCAL    0x2d    * User Monitor app 2
*
* Processes to be started
FORK_PROCESS tim
FORK_PROCESS tick
FORK_PROCESS s7_log -m0xef -tt
FORK_PROCESS s7_log -m0x1d
FORK_PROCESS s7_log -m0x2d
FORK_PROCESS HSTBIN/mst -Lp./
```

Figure 1: Example system.txt file

- 1) Start the gct environment by running the gctload task. Reference the system.txt file on the command line.
 - a) gctload -csystem.txt
- 2) Configuration of the MST Module. The module configuration message (0x7e40) must be sent to the MST module to set certain important parameters. This message may be generated by a Management application or it may be formatted in text and sent using the **s7_play** utility. To ensure a response is sent, the response field should be set
 - a) The management ID should be set to that used in the system.txt file
 - b) The Congestion Buffer size may be set to zero to allow the default buffer size of 1000 messages to be allocated.
 - c) Both the Options and Throughput field should be set to zero.

An example configuration message is shown in Figure 2 (Solaris and Linux) or Figure 3 (Windows) below.

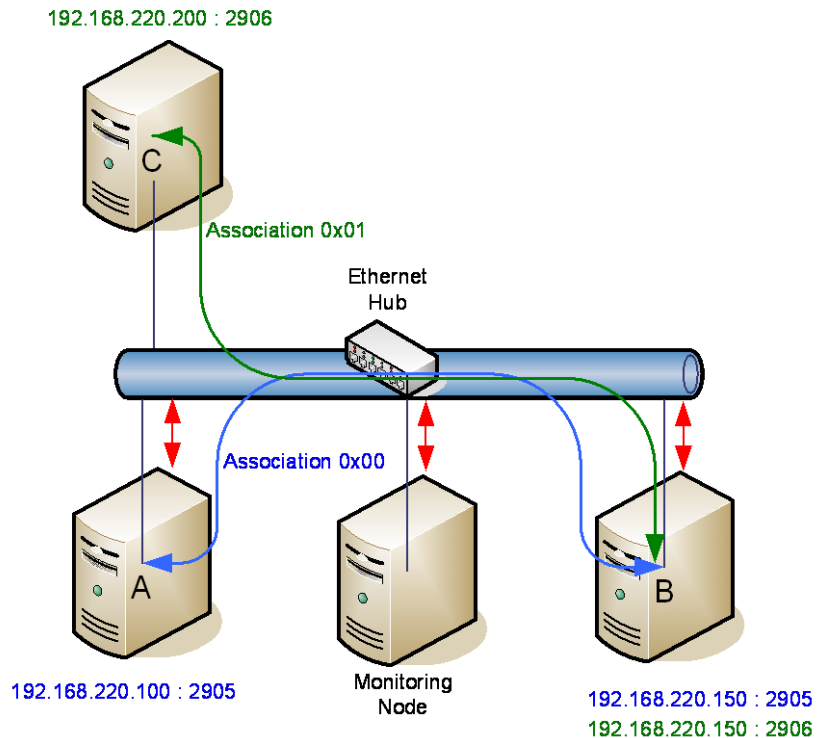
- c) Set the port to run the promiscuous mode.
- d) Copy the Ethernet Port device name as a NULL terminated ASCII string.

Example configuration messages are shown above in Figure 2.

- 2) Finally, each TAP required should be configured (0x7e42)
 - a) Set the TAP_ID
 - b) Set the bitmap of port_ids that this TAP can use.
 - c) Set the SIGTRAN events that this TAP should monitor.
 - d) Set the SS7 events that this TAP should monitor.
 - e) Set the format in which these messages should be presented.
 - f) Set the module_id to which these messages should be sent.

Example configuration messages are shown above in Figure 2.

- 3) (Optional) Additional configuration messages may also be sent to MST to enable duplicate filtering and ordered delivery of SCTP DATA chunks. For instance, the SCTP DATA chunks belonging to association 0x00 from A to B or association 0x01 from C to B are to be filtering so that no duplicated chunks are reported to the TAP.



The following configuration messages configure MST to filter duplicate DATA chunks on these two associations.

```

*
***** Duplicate Filtering setup for DATA chunks on Association 0x00 between A and B
*
* Define endpoint A of association between A and B(port 2905)
*   ----transport_addr_id
*           ----endpoint_id
*           ----sctp port num=2905
*           --ip type
*           -----unused
*           -----ipv4 address=192.168.220.100
M-t7e43-i0000-fef-db1-r8000-p00000b5901000000c0a8dc6400000000000000000000000000000000
*
* Define endpoint B of association between A and B
M-t7e43-i0001-fef-db1-r8000-p00010b5901000000c0a8dc9600000000000000000000000000000000
*
* Configure Association 0x00 between endpoint A & endpoint B with duplicate
* filtering enabled
*   ---- Association_id 0x00
*           ----endpointA
*           ----endpointB
*           -----option=duplicate filtering
*           -----storage number=1024 chunks
*           -----storage time=200ms
M-t7e44-i0000-fef-db1-r8000-p000000010000000100000400000000002
*
*
***** Duplicate Filtering setup for DATA chunks on Association 0x01 between C and B
*
* Define endpoint C of association between C and B(port 2906)
M-t7e43-i0002-fef-db1-r8000-p00020b5a01000000c0a8dcc800000000000000000000000000000000
*
* Define endpoint B of association between C and B(port 2906)
M-t7e43-i0003-fef-db1-r8000-p00030b5a01000000c0a8dc9600000000000000000000000000000000
*
* Configure Association 0x01 between endpoint C & endpoint B with duplicate
* filtering enabled
M-t7e44-i0001-fef-db1-r8000-p000200030000000100000400000000002

```

Figure 4: Example MST Configuration for filtering of duplicated DATA chunks

4 Message Reference

4.1 Overview

This section describes the individual messages and associated parameters that may be sent to or received from the MST module.

The interface is a message-based interface using messages of type MSG as defined in the *Software Environment Programmer's Manual*.

The following categories of messages are defined:

Configuration	Messages sent during initialisation to configure module.
Payload	Messages sent from MST to the monitoring application encapsulating a monitored packet.
State and Stats	Management messages sent to the MST to query it's state and request statistics on it's operation.
Event	Event indications proactively sent by MST to a management module.
Traffic	When operating with a Virtual Port these messages are passed to MST encapsulating a network packet.

4.1.1 Message Summary Table

The following table summarises the set of messages applicable to Sigtran Monitoring Binary (MST). The message format is fully defined in the following sections.

Message Mnemonic	Message Type	Message Description
MST_MSG_CONFIG	0x7e40	Module Configuration Message. Sent to MST once at startup.
MST_MSG_CFG_PORT	0x7e41	Port Configuration Message. Sent to MST once for each Ethernet port to be monitored.
MST_MSG_CFG_TAP	0x7e42	Tap Configuration Message. Sent to MST once for each monitor stream required.
MST_MSG_CFG_TRANSPORT_ADDR	0x7e43	Sent to the MST module to define an SCTP endpoint transport address. Used for Duplicate Filtering and Ordered Delivery of DATA chunks.
MST_MSG_CFG_ASSOCIATION	0x7e44	Sent to the MST module to define the two endpoints for an SCTP association. Used for Duplicate Filtering and Ordered Delivery of DATA chunks.
MGT_MSG_TRACE_MASK	0x5e4f	Configure module tracing for diagnostic purposes.
MST_MSG_SELTRACE_MASK	0x5e50	Enable / Disable selective trace events.
MST_MSG_IP_MSG	0x8e47	Indication of monitored message to User Application (format includes IP and SCTP headers).

Message Mnemonic	Message Type	Message Description
MST_MSG_M3UA_MSG	0x8e48	Indication of monitored M3UA message to User Application (format excludes IP and SCTP headers).
MST_MSG_M2PA_MSG	0x8e49	Indication of monitored M2PA message to User Application (format excludes IP and SCTP headers).
MST_MSG_R_STATS	0x6e44	Module statistics request. Sent to MST to request per-module statistics.
MST_MSG_R_PORT_STATS	0x6e46	Port statistics request. Sent to MST to request statistics for a specific PORT.
MST_MSG_R_TAP_STATS	0x6e45	Tap statistics request. Sent to MST to request statistics for a specific TAP.
MST_MSG_R_ASSOCIATION_STATS	0x6e47	Association Statistics request. Sent to MST to request statistics for an association configured for Duplicate Filtering and Ordered Delivery.
MGT_MSG_R_THR_STATS	0x6f21	Message sent to MST to request the current licensing throughput statistics.
MGT_MSG_R_LIC_STATUS	0x6f22	Message sent to MST to request the current license state.
MST_MSG_FLCT_EVENT_IND	0x0e4a	Flow Control Event Indication. Sent by MST to layer management to indicate internal buffers have become congested or that messages have been discarded.
MST_MSG_EVENT_IND	0x0e4c	MST Event Indication. Sent from MST to management to indicate various events.
MGT_MSG_LIC_EVENT	0x0f23	Message sent to management to indicate a license-related event.
MST_MSG_IP_TRACE	0xce4e	Message sent by MST for diagnostic purposes to trace a frame received from the internal IP stack.
MST_MSG_VPORT_PKT	0xce4d	Used to encapsulate and send IP packets to MST. It requires the configuration of an MST Port of 'port type' Virtual.

Table 1: Message Type Summary

4.1.2 Status Code Values

Unless otherwise noted, when the MST module returns a confirmation message the status will be set as follows:

Mnemonic	Value	Description
MSTE_OK	0x00	Success.
MSTE_BAD_ID	0x01	Inappropriate or invalid id in request message.
MSTE_BAD_STATE	0x02	Message received in wrong state.
MSTE_BAD_MSG	0x05	Unsupported message received.
MSTE_BAD_PARAM	0x06	Invalid parameters contained in message.
MSTE_NO_RESOURCES	0x07	Insufficient internal message resources.

MSTE_BAD_MSG_LEN	0x0b	Invalid message length.
MSTE_LICENSE_ERR	0x0e	Command failed due to a licensing restriction.
MSTE_INTERNAL_ERR	0x0f	Command failed due to an internal error.

4.2 Configuration Messages

4.2.1 MST_MSG_CONFIG - MST Module Configuration

Synopsis

Message sent to MST to configure per-module parameters.

Format

MESSAGE HEADER		
Field Name	Meaning	
type	MST_MSG_CONFIG (0x7e40)	
id	0	
src	Sending module_id	
dst	MST module_id	
rsp_req	Used to request a confirmation	
hclass	0	
status	Non zero on error	
err_info	0	
len	20	
PARAMETER AREA		
Offset	Size	Name
0	1	mngt_mod_id
1	1	trace_mod_id
2	2	cong_buff_size
4	4	options
8	4	throughput
12	4	num_associations
16	4	association_buff_size

Description

This message should be the first message sent to MST to configure the module-wide settings and to validate that the required options and throughput are licensed.

Parameters

mngt_mod_id

Module_id of the Management Module to which MST will send event indications.

trace_mod_id

Module_id of the trace module to which MST will send any trace messages.

cong_buff_size

Number of MSGs that MST will allocate internally to buffer messages for periods of system congestion. Typically this could be set to the expected number of messages to be received over an 0.5 second interval. Users should take this into account when dimensioning the message pool size within system.txt.

options

32 bit mask for run-time configuration options for the module. Currently all bits are reserved for future use and should all be set to zero.

throughput

This parameter may be used to validate at startup that the MST module is adequately licensed to support a specified throughput level. If specified, throughput should be in terms of the maximum Bytes/s required from the MST module.

If the specified throughput is less than the amount supported by the run-time license then the configuration message will be rejected with error status. The throughput parameter does not modify the licensed capability in any way.

num_associations

Duplicate filtering and ordered delivery of DATA chunks requires MST to be configured with details of the SCTP associations which have these features enabled. This parameter defines the maximum number of associations that can be configured.

This value should not exceed 128. If set to zero a default value of 8 associations is supported.

association_buff_size

When the option to select in-sequence delivery of monitored messages is invoked it is necessary to allocate internal buffers to temporarily store out of sequence messages. This parameter defines the number of message buffers allocated per association which must be set to a power of 2 which is 16 or greater (eg 16, 32, 64, 128, 256 etc).

If set to zero a default value of 512 is used.

4.2.2 MST_MSG_CFG_PORT - MST Port Configuration

Synopsis

Sent to the MST module to configure a port.

Format

MESSAGE HEADER		
Field Name	Meaning	
type	MST_MSG_CFG_PORT (0x7e41)	
id	port_id	
src	Sending module_id	
dst	MST module_id	
rsp_req	Used to request a confirmation	
hclass	0	
status	Non zero on error	
err_info	0	
len	38	
PARAMETER AREA		
Offset	Size	Name
0	2	port_type
2	4	port_options
6	32	device_name

Description

This message is used to configure the source port from which data will be monitored. Multiple posts can be monitored and this message should be sent once for each port to be monitored.

Parameters

port_id

The local logical identifier for the port in the range 0 to one less than the maximum number of ports supported.

port_type

The port type indicates whether the data is to be read from a local Ethernet port (in which case the device_name must also be specified) or from a virtual port (where the messages to be monitored are sent to MST using the MST_MSG_VPORT_PKT message).

port_type	Mnemonic	Description
1	PRTTYPE_ETHERNET	Data to be monitored is read directly from the underlying Ethernet port.
2	PRTTYPE_VIRTUAL	Data to be monitored is passed to MST using the MST_MSG_VPORT_PKT message.

port_options

The port options are set by this 32 bit options field. Current options are defined in the table below and all other bits should be set to zero.

Bit	Mnemonic	Description
0	PRTOPT_PROMISCUOUS	If set to 1, causes the port to operate in promiscuous mode allowing monitoring of all packets including those not explicitly addressed to this node. If set to zero only packets addressed to this node are monitored.

device_name

The device name of the Ethernet port being configured in null-terminated ascii string format. This name will be used by MST when calling the low level operating system functions to open the port for reading. A common name for an Ethernet port device name is "eth0".

For Windows the device name should be set to the IP address of the Ethernet port formatted as a hexadecimal string - For example IP Address 192.168.1.10 would be represented as device name c0a8010a.

When port_type is set to PRTTYPE_VIRTUAL the device_name parameter should be set to zero (ie. a null string).

4.2.3 MST_MSG_CFG_TAP - MST TAP Configuration

Synopsis

Message sent to MST to configure the attributes of a Tap.

Format

MESSAGE HEADER		
Field Name	Meaning	
type	MST_MSG_CFG_TAP (0x7e42)	
id	tap_id	
src	Sending module_id	
dst	MST module_id	
rsp_req	Used to request a confirmation	
hclass	0	
status	Non zero on error	
err_info	0	
len	22	
PARAMETER AREA		
Offset	Size	Name
0	4	options
4	1	module_id
5	1	Reserved, set to 0.
6	2	instance
8	2	format
10	4	stn_mask
14	4	ss7_mask
18	4	port_bmp

Description

This message is used to configure the attributes for a specific set of messages to be sent to the application. The attributes include the list of ports that will supply messages for the Tap and filters to select which types of traffic are reported to the application. Multiple Taps can be established, each with its own tap_id and using a separate Tap configuration message.

Parameters

tap_id

The local logical identifier for the Tap in the range 0 to one less than the maximum number of Taps supported. The tap_id is included in all messages sent to the application.

options

The run-time options for the tap are set by this 32 bit options field. Current options are defined in the table below and all other bits should be set to zero.

Bit	Mnemonic	Description
0	MST_TAP_OPT_COMBINE_MASKS	If set to 1, the traffic will only be reported to the application if it meets the criteria in both the SS7 and SIGTRAN event masks. If set to zero the message will be reported if it meets either one of the SS7 or SIGTRAN criteria.

module_id

The `module_id` of the user's application module to which all monitored messages for this Tap will be sent.

instance

The value that the instance field will be set to in all monitored messages issued to the application for this Tap. Typically this field should be set to zero.

format

MST allows monitored messages to be reported to the application in one of two different formats depending on the needs of the application. In one format all the low level IP and SCTP header information is included in the message whilst in the other format the message includes just the MTP routing label and user part payload. The format parameter should be set to the desired value from the table below:

format	Format Mnemonic	Description
1	MST_TAP_FMT_ETH_PAYLOAD	Monitored messages are sent to the application using the <code>MST_MSG_IP_MSG</code> message which includes the IP and SCTP headers in addition to the user part payload.
2	MST_TAP_FMT_STN_PAYLOAD	Monitored messages are sent to the application using the <code>MST_MSG_M3UA_MSG</code> message (for M3UA data) or the <code>MST_MSG_M2PA_MSG</code> message (for M2PA data). These messages remove the IP and SCTP headers making it easier for the application to access the payload of the message.

stn_mask

The `stn_mask` is used in conjunction with `ss7_mask` and bit zero of the options parameter to determine which received messages are passed to the user application. `stn_mask` is a 32 bit filter where each bit, when set, specifies a specific type of SIGTRAN traffic that will be a candidate for passing to the application. Multiple bits can be set. The meaning of each bit is detailed in the following table:

bit	stn_mask mnemonic	Value	Description
0	MST_TAP_STN_EVT_MASK_SCTP_DATA	0x0001	SCTP Data Messages
1	MST_TAP_STN_EVT_MASK_SCTP_CTL	0x0002	SCTP Control Messages
2	MST_TAP_STN_EVT_MASK_SCTP_SACK	0x0004	SCTP Acknowledgement Messages
3	MST_TAP_STN_EVT_MASK_M3UA_DATA	0x0008	M3UA Data Messages
4	MST_TAP_STN_EVT_MASK_M3UA_CTL	0x0010	M3UA Control Messages
5	MST_TAP_STN_EVT_MASK_M2PA_DATA	0x0020	M2PA Data Messages

6	MST_TAP_STN_EVT_MASK_M2PA_LS	0x0040	M2PA Link Status Messages
7	MST_TAP_STN_EVT_MASK_M2PA_ACK	0x0080	M2PA Data Acknowledgement Messages

ss7_mask

The `ss7_mask` is used in conjunction with `stn_mask` and bit zero of the options parameter to determine which received messages are passed to the user application. `ss7_mask` is a 32 bit filter where each bit, when set, specifies a specific type of SS7 traffic that will be a candidate for passing to the application. Multiple bits can be set. The meaning of each bit is detailed in the following table:

bit	ss7_mask mnemonic	Value	Description
0	MST_TAP_STN_EVT_MASK_ISUP	0x0001	ISUP Messages
1	MST_TAP_STN_EVT_MASK_TUP	0x0002	TUP Messages
2	MST_TAP_STN_EVT_MASK_NUP	0x0004	NUP Messages
3	MST_TAP_STN_EVT_MASK_SCCP	0x0008	SCCP Messages
4	MST_TAP_STN_EVT_MASK_BICC	0x0010	BICC Messages

port_bmp

The bit map of all the ports that MST should consider as candidates for this Tap. `port_bmp` is a 32 bit value with bit `n` set to 1 for each `port_id=n` that should be used. For example to monitor just from `port_id=0`, `port_bmp` should be set to the value 0x00000001, whilst to monitor from `port_id`'s 0, 1 and 2 `port_bmp` should be set to 0x00000007.

4.2.4 MST_MSG_CFG_TRANSPORT_ADDR - Transport Address Configuration

Synopsis

Message sent to MST to define an SCTP endpoint transport address.

Format

MESSAGE HEADER		
Field Name	Meaning	
type	MST_MSG_CFG_TRANSPORT_ADDR (0x7e43)	
id	transport_id	
src	Sending module_id	
dst	MST module_id	
rsp_req	Used to request a confirmation	
hclass	0	
status	Non zero on error	
err_info	0	
len	24	
PARAMETER AREA		
Offset	Size	Name
0	2	endpoint_id
2	2	port_num
4	1	ip_type
5	3	Reserved, should be set to 0
8	16	ip_addr

Description

This message is used in conjunction with the MST_MSG_CFG_ASSOCIATION message to configure the attributes of an SCTP association when advanced options such as filtering of duplicate frames and insequence message delivery are required.

To allow MST to determine which messages belong to the specific SCTP association it is necessary to configure (using one MST_MSG_CFG_TRANSPORT_ADDR message per IP address) all the IP addresses for one end of the association (Endpoint A) and all the IP addresses of the other end of the association (Endpoint B). Finally the MST_MSG_CFG_ASSOCIATION message should be used to bind the two endpoints to an association_id.

Each transport address is made up of IP address and SCTP port number.

Parameters

transport_id

Unique logical identifier for the transport address. This should be in the range from 0 up to one less than the maximum number of transport addresses supported. Each association may have up to eight transport addresses configured.

endpoint_id

Logical identifier of the SCTP endpoint. Multiple messages may refer to the same endpoint_id in order to correctly configure all the IP addresses relating to one endpoint of a multi-homed SCTP association.

port_num

The SCTP port number of the association endpoint.

ip_type

This field is used to specify the format of the IP address. Currently only one format is supported (IPv4) so this field should always be set to the value 1.

ip_addr

The IP address of this transport address. For an IPv4 IP address the first four bytes of this 16 byte field represent the IP address (Most significant byte first) and the remaining bytes must be set to zero.

status

The following values can be returned in the message header status field:

Value	Error Code	Description
0x00	MSTE_OK	Configuration message accepted
0x06	MSTE_BAD_PARAM	Endpoint already defined
0x06	MSTE_BAD_PARAM	SCTP port number must all be the same for a given endpoint
0x0B	MSTE_BAD_MSG_LEN	Invalid message length
0x0F	MSTE_INTERNAL_ERR	Too many endpoints already defined

4.2.5 MST_MSG_CFG_ASSOCIATION - Association Configuration

Synopsis

Message sent to MST to define an SCTP association in terms of two previously configured endpoints.

Format

MESSAGE HEADER		
Field Name		Meaning
type		MST_MSG_CFG_ASSOCIATION (0x7e44)
id		association_id
src		Sending module_id
dst		MST module_id
rsp_req		Used to request a confirmation
hclass		0
status		Non zero on error
err_info		0
len		16
PARAMETER AREA		
Offset	Size	Name
0	2	endpoint_a
2	2	endpoint_b
4	4	options
8	4	max_stored_chunks
12	4	max_storage_time

Description

This message is used in conjunction with the MST_MSG_CFG_TRANSPORT_ADDR message to configure the attributes of an SCTP association when advanced options such as filtering of duplicate frames and insequence message delivery are required.

Once all of the transport addresses for each end of the SCTP association have been configured (using the MST_MSG_CFG_TRANSPORT_ADDR message), the MST_MSG_CFG_ASSOCIATION message should be used to bind the two endpoints to an association_id.

Parameters

association_id

The unique logical identifier of the association in the range from 0 to one less than the maximum number of associations supported.

endpoint_a

Logical identifier of a previously configured association endpoint, as specified in the 'endpoint_id' field of one or more MST_MSG_CFG_TRANSPORT_ADDR messages, that will form Endpoint A for this association.

endpoint_b

Logical identifier of a previously configured association endpoint, as specified in the 'endpoint_id' field of one or more MST_MSG_CFG_TRANSPORT_ADDR messages, that will form Endpoint B for this association.

options

A 32 bit value containing run-time options for this association as defined in the following table:

Bit	Mnemonic	Description
0	MST_ASSOCIATION_OPTION_DUPLICATES	If set to 1, filtering of duplicate messages is enabled
1	MST_ASSOCIATION_OPTION_ORDERED_DELIV	If set to 1, in-sequence delivery of messages is enabled so messages arriving out of sequence will be buffered within MST until other missing messages have been received.

max_stored_chunks

The maximum number of data chunks that can be stored by the in-sequence delivery algorithm for this association at any one time which must be set to a power of 2 which is 16 or greater (eg 16, 32, 64, 128, 256 etc). Upon reaching this limit, any stored data chunks are sent to the user's application as a best effort attempt.

If set to zero, the default value configured in the module configuration message is used.

max_storage_time

The maximum amount of time in multiples of 100ms that a data chunk will be stored by the in-sequence delivery algorithm for this association. MST will wait that amount of time for a data chunk before deciding that it is missing. After this time any received messages will be passed to the user application.

If set to zero, the internal default value of 500ms is used.

status

The following values can be returned in the message header status field:

Value	Error Code	Brief Description
0x00	MSTE_OK	Configuration message accepted
0x06	MSTE_BAD_PARAM	Association_id already defined
0x06	MSTE_BAD_PARAM	Endpoint_id not yet defined
0x06	MSTE_BAD_PARAM	First and Second Endpoint_id can not be the same
0x06	MSTE_BAD_PARAM	Data Chunk Storage Size can not be more than the "Filter buffer size" value specified in MST_CFG_CONFIG message
0x0B	MSTE_BAD_MSG_LEN	Invalid message length
0x0F	MSTE_INTERNAL_ERR	Too many associations already defined

4.2.6 MGT_MSG_TRACE_MASK - Trace Mask Configuration

Synopsis

Message used to configure diagnostic run-time tracing for MST module.

Format

MESSAGE HEADER		
Field Name	Meaning	
type	MGT_MSG_TRACE_MASK (0x5e4f)	
id	0	
src	Sending module_id	
dst	MST module_id	
rsp_req	Used to request a confirmation	
hclass	0	
status	0	
err_info	0	
len	12	
PARAMETER AREA		
Offset	Size	Name
0	4	op_evt_mask
4	4	ip_evt_mask
8	4	non_prim_mask

op_evt_mask

The output event trace mask. This is a 32-bit value with bits set to 1 to cause a trace message to be sent to the trace module when MST sends the associated message.

Bit	31	30	29	28	27	26	25	24
Meaning	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Meaning	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Meaning	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Meaning	0	0	0	0	0	IP_MSG	M3UA_MSG	M2PA_MSG

IP_MSG - MST_MSG_IP_MSG

M3UA_MSG - MST_MSG_M3UA_MSG

M2PA_MSG - MST_MSG_M2PA_MSG

ip_evt_mask

The input event trace mask. This is a 32-bit value with bits set to 1 to cause a trace message to be sent to the trace module when MST receives the associated message.

Bit	31	30	39	28	27	26	25	24
Meaning	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Meaning	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Meaning	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Meaning	0	0	0	0	0	0	IP_TRACE	VPORT_PKT

IP_TRACE - MST_MSG_IP_TRACE

VPORT_PKT - MST_MSG_VPORT_PKT

non_prim_mask

The non-primitive trace mask. This is a 32-bit value with bits set to 1 to cause a trace message to be sent to the trace module when MST receives or issues the associated non-primitive message.

Bit	31	30	39	28	27	26	25	24
Meaning	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Meaning	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Meaning	0	MOD_IDENT	LIC_EVENT	LIC_ST ATUS	LIC_STATS	EVENT	FLCT_EVENT	ASC_STATS

Bit	7	6	5	4	3	2	1	0
Meaning	PORT_STATS	TAP_STATS	STATS	TRACE_MASK	CFG_ASC	CFG_ADDR	CFG_TAP	CFG_PORT

CFG_PORT - MST_MSG_CFG_PORT

CFG_TAP - MST_MSG_CFG_TAP

CFG_ADDR – MST_MSG_CFG_TRANSPORT_ADDR
CFG_ASC – MST_MSG_CFG_ASSOCIATION
TRACE_MASK – MGT_MSG_TRACE_MASK
STATS – MST_MSG_R_STATS
TAP_STATS – MST_MSG_R_TAP_STATS
PORT_STATS – MST_MSG_R_PORT_STATS
ASC_STATS – MST_MSG_R_ASSOCIATION_STATS
FLCT_EVENT – MST_MSG_FLCT_EVENT_IND
EVENT – MST_MSG_EVENT_IND
LIC_STATS – MGT_MSG_R_THR_STATS
LIC_STATUS – MGT_MSG_R_LIC_STATUS
LIC_EVENT – MGT_MSG_LIC_EVENT
MOD_IDENT – GEN_MSG_MOD_IDENT

4.2.7 MST_MSG_SELTRACE_MASK - Selective Trace Mask Configuration

Synopsis

Message sent to MST to control which management events are traced on occurrence to management.

Format

MESSAGE HEADER		
Field Name	Meaning	
type	MST_MSG_SELTRACE_MASK (0x5e50)	
id	0	
src	Sending module_id	
dst	MST module_id	
rsp_req	Used to request a confirmation	
hclass	0	
status	0	
err_info	0	
len	9	
PARAMETER AREA		
Offset	Size	Name
0	9	Mask – Bits set to indicate selective trace events which should be active

Description

This message allows the user to control which protocol events cause generation of the Management Event Indication (MST_MSG_EVENT_IND) messages and the corresponding Selective Trace Event Indication (MGT_MSG_SEL_TRACE) messages containing the original message that caused the maintenance event.

By default selective tracing is enabled for all management events but this message can be used to disable some or all of the selective trace events (in which case generation of the corresponding Management Event Indication will also be disabled).

This message may be sent to MST at any time after the initial per-module configuration message and the selective trace mask can be changed at run-time as often as required.

The Selective Trace Event Indication message is similar in format to the Trace Event Indication MGT_MSG_TRACE_EV (0x0003) as defined in the Software Environment Programmer's Manual with the exception that the message type is set to MGT_MSG_SEL_TRACE (0x0f16) and the status field in the header is set to the module-specific event_id (as defined in the table below) which caused the event to be generated.

Parameters

Mask

An array of bytes representing a bit mask of the selective trace events which are active.

The least significant bit of the first byte represents event_id=0, the most significant bit of the first octet represents event_id=7, the least significant bit of the second byte represents event_id=8 and so on.

A '1' in the appropriate bit position indicates that an event is active and a '0' indicates that it is not active.

Selective Trace event_id's use the same set of values as the MST_MSG_EVENT_IND as defined in Section 4.5.2.

4.3 Payload Indication Messages

4.3.1 MST_MSG_IP_MSG - IP Payload Indication

Synopsis

Message sent by MST to convey monitored messages to the user application. This message type is used when the Tap is configured with format set to 1.

Format

MESSAGE HEADER		
Field Name	Meaning	
type	MST_MSG_IP_MSG (0x8e47)	
id	tap_id	
src	MST module_id	
dst	Per-tap user application module_id	
rsp_req	0	
hclass	0	
status	0 (1 if truncated)	
err_info	0	
len	Number of bytes in parameter area	
PARAMETER AREA		
Offset	Size	Name
0	len	Message data starting with the IP header

Description

This message conveys monitored message to the user preserving the IP and SCTP headers to allow the application to extract low level parameters as required. It is generated when the format parameter in the Tap configuration message is set to value 1 (MST_TAP_FMT_ETH_PAYLOAD).

The Parameter area commences with the IP header in network byte order. Each message will contain a single SCTP Chunk. If there were multiple chunks in the packet as it appeared on the link, then multiple message are sent to the application each containing a single chunk and the IP header. The length field in the IP header is adjusted to reflect any change in message length.

If the received packet is too large to fit into the 320 byte parameter of the message then it is truncated and the status field is set to 1. This will not affect any length fields present in the message.

Parameters

Message Data

Raw data as received from the link in Network byte order, starting with the IP header and continuing with the SCTP header, the SCTP chunk header and other data contained in the message as seen on the link according to RFC 4960.

4.3.2 MST_MSG_M3UA_MSG - M3UA Payload Indication

Synopsis

Message sent by MST to convey monitored M3UA messages to the user application. This message type is used when the Tap is configured with format set to 2.

Format

MESSAGE HEADER		
Field Name	Meaning	
type	MST_MSG_M3UA_MSG (0x8e48)	
id	tap_id	
src	MST module_id	
dst	Per-tap user application module_id	
rsp_req	0	
hclass	0	
status	0	
err_info	0	
len	12 + length of payload parameter	
PARAMETER AREA		
Offset	Size	Name
0	4	opc
4	4	dpc
8	1	si
9	1	ni
10	1	mp
11	1	sls
12	len - 12	payload

Description

This message conveys monitored M3UA message to the user preserving the M3UA headers and payload but without the low level IP and SCTP headers. It is generated when the format parameter in the Tap configuration message is set to value 2 (MST_TAP_FMT_STN_PAYLOAD).

Parameters

opc

The Originating Point Code contained in the message. It is a 32 bit value and should be extracted from the message using the `runpackbytes()` function.

dpc

The Destination Point Code contained in the message. It is a 32 bit value and should be extracted from the message using the `runpackbytes()` function.

si

The Service Indicator contained in the message which indicated the associated User Part.

ni

The Network Indicator contained in the message.

mp

The Message Priority contained in the message.

sls

The Signaling Link Selection field contained in the message.

payload

The MTP3 payload data from the message formatted exactly as received from the network in network byte order commencing with the first byte after the MTP routing label.

4.3.3**MST_MSG_M2PA_MSG - M2PA Payload Indication****Synopsis**

Message sent by MST to convey monitored M2PA messages to the user application. This message type is used when the Tap is configured with format set to 2.

Format

MESSAGE HEADER		
Field Name	Meaning	
type	MST_MSG_M2PA_MSG (0x8e49)	
ld	tap_id	
src	MST module_id	
dst	Per-tap user application module_id	
rsp_req	0	
hclass	0	
status	0	
err_info	0	
len	10 + length of sif parameter	
PARAMETER AREA		
Offset	Size	Name
0	4	bsn
4	4	fsn
8	1	pri
9	1	sio
10	len - 10	sif

Description

This message conveys monitored M2PA message to the user preserving the M2PA headers and payload but without the low level IP and SCTP headers. It is generated when the format parameter in the Tap configuration message is set to value 2 (MST_TAP_FMT_STN_PAYLOAD).

Parameters

bsn

The M2PA Backward Sequence Number of the message. It is a 32 bit value and should be extracted from the message using the `runpackbytes()` function.

fsn

The M2PA Forward Sequence Number of the message. It is a 32 bit value and should be extracted from the message using the `runpackbytes()` function.

pri

The M2PA Priority of the message.

sio

The Signaling Information Octet of the message. This holds the Service Indicator, Sub Service Field and Network Indicator.

sif

The Signaling Information Field of the received message. This holds the payload of the M2PA message formatted as received from the network commencing with with the dpc of the routing label.

4.4 State and Statistics Request Messages

4.4.1 MST_MSG_R_STATS - MST Module Statistics Request

Synopsis

Message sent to MST request per-module statistics.

Format

MESSAGE HEADER		
Field Name	Meaning	
type	MST_MSG_R_STATS (0x6e44)	
id	0	
src	Sending module_id	
dst	MST module_id	
rsp_req	Used to request a confirmation	
hclass	0	
status	0 – Read statistics 1 – Read and reset statistics	
err_info	0	
len	44	
PARAMETER AREA		
Offset	Size	Name
0	4	stats_time
4	4	msg_fwd_count
8	4	bytes_fwd_count
12	4	msg_rxd_count
16	4	bytes_rxd_count
20	4	bad_fmt_count
24	4	unknown_fmt_count
28	4	max_buffer_occupancy
32	4	congestion_count
36	4	discard_count
40	4	drop_count

Description

This message is used to request module wide statistics for the MST module. The sending module should allocate a message with the correct length parameter area which will be populated by MST and returned as a confirmation message.

The statistics can optionally be reset by setting the status field to 1.

Parameters

stats_time

The period in seconds over which the statistics have been gathered.

msg_fwd_count

The total number of messages forwarded to the user application.

bytes_fwd_count

The total number of bytes in messages that have been sent to the user application.

msg_rxd_count

The total number of messages (IP packets) received from network ports.

bytes_rxd_count

The total number of bytes (as appeared on the network link) received from network ports.

bad_fmt_count

The total number of badly formatted messages (IP packets) received from the network.

unknown_fmt_count

The total number of messages (IP packets) received from the network that were either of an unknown or unsupported format.

max_buffer_occupancy

The maximum number of internal buffers in use by the MST module at any one time.

congestion_count

The number times the MST module has gone into congestion.

discard_count

The number of messages discarded due to buffer overload.

drop_count

The number of messages that have been dropped by the underlying IP stack due to overload. This field is only supported under Solaris, for other operating systems it is set to zero.

4.4.2 MST_MSG_R_PORT_STATS - MST Port Statistics Request

Synopsis

Message sent to MST to request statistics for a specific Port.

Format

MESSAGE HEADER		
Field Name	Meaning	
type	MST_MSG_R_PORT_STATS (0x6e46)	
id	port_id	
src	Sending module_id	
dst	MST module_id	
rsp_req	Used to request a confirmation	
hclass	0	
status	0 – Read statistics 1 – Read and reset statistics	
err_info	0	
len	20	
PARAMETER AREA		
Offset	Size	Name
0	4	stats_time
4	4	msg_rxd_count
8	4	bytes_rxd_count
12	4	error_count
16	4	drop_count

Description

This message is used to request statistics for a specific MST port_id. The sending module should allocate a message with the correct length parameter area which will be populated by MST and returned as a confirmation message.

The statistics can optionally be reset by setting the status field to 1.

Parameters

stats_time

The period in seconds over which the statistics have been gathered.

msg_rxd_count

The number of messages (IP packets) received from this network port.

bytes_rxd_count

The number of Kbytes (as appeared on the network link) received from this network port.

error_count

The number messages from this network port dropped by MST due to format errors.

drop_count

The number of messages from this network port that have been dropped by the underlying IP stack due to overload. This field is only supported under Solaris, for other operating systems it is be set to zero.

4.4.3 MST_MSG_R_TAP_STATS - MST TAP Statistics Request

Synopsis

Message sent to MST to request statistics for a specific Tap.

Format

MESSAGE HEADER		
Field Name	Meaning	
type	MST_MSG_R_TAP_STATS (0x6e45)	
id	tap_id	
src	Sending module_id	
dst	MST module_id	
rsp_req	Used to request a confirmation	
hclass	0	
status	0 – Read statistics 1 – Read and reset statistics	
err_info	0	
len	12	
PARAMETER AREA		
Offset	Size	Name
0	4	stats_time
4	4	msg_fwd_count
8	4	bytes_fwd_count

Description

This message is used to request statistics for a specific MST tap_id. The sending module should allocate a message with the correct length parameter area which will be populated by MST and returned as a confirmation message.

The statistics can optionally be reset by setting the status field to 1.

Parameters

stats_time

The period in seconds over which the statistics have been gathered.

msg_fwd_count

The number of messages forwarded to the user application for this tap_id.

bytes_fwd_count

The total number of bytes (measured in Kbytes) in messages that have been sent to the user application for this tap_id.

4.4.4 MST_MSG_R_ASSOCIATION_STATS - MST Association Statistics Request

Synopsis

Message sent to MST to request statistics for a specific association.

Format

MESSAGE HEADER		
Field Name	Meaning	
type	MST_MSG_R_ASSOCIATION_STATS (0x6e47)	
id	association_id	
src	Sending module_id	
dst	MST module_id	
rsp_req	Used to request a confirmation	
hclass	0	
status	0 – Read statistics 1 – Read and reset statistics	
err_info	0	
len	120	
PARAMETER AREA		
Offset	Size	Name
0	4	stats_time
4	4	inactive_count
8	4	association_uptime
12	28	reserved
40	4	rx_a_count
44	4	rx_b_count
48	4	tx_a_count
52	4	tx_b_count
56	4	data_rx_a_count
60	4	data_rx_b_count
64	4	data_tx_a_count
68	4	data_tx_b_count
72	4	data_duplicate_a_count
76	4	data_duplicate_b_count
80	4	data_missing_a_count
84	4	data_missing_b_count
88	4	data_oos_a_count
92	4	data_oos_b_count
96	4	buffer_overflow_a_count
100	4	buffer_overflow_b_count
104	4	data_max_stored_a_count
108	4	data_max_stored_b_count
112	4	restart_a_count
116	4	restart_b_count

Description

This message is used to request statistics for a specific MST association_id. The sending module should allocate a message with the correct length parameter area which will be populated by MST and returned as a confirmation message.

The statistics can optionally be reset by setting the status field to 1.

Parameters**stats_time**

The period in seconds over which the statistics have been gathered.

inactive_count

The number of times the association has gone inactive. MST considers an association to be inactive if no SCTP chunks have been received for a period of 50 times the max_storage_time specified in the per-association configuration message MST_MSG_CFG_ASSOCIATION.

association_uptime

The duration for which the association has been established.

rx_a_count, rx_b_count

The Received Chunks Count which is the number of SCTP chunks (of all types including DATA type) received by Endpoint A or Endpoint B of the specified association.

tx_a_count, tx_b_count

The Transmitted Chunks Count which is the number of SCTP chunks (of all types including DATA type) transmitted by Endpoint A or Endpoint B of the specified association.

data_rx_a_count, data_rx_b_count

The number of SCTP DATA chunks received by Endpoint A or Endpoint B.

data_tx_a_count, data_tx_b_count

The number of SCTP DATA chunks transmitted by Endpoint A or Endpoint B.

data_duplicate_a_count, data_duplicate_b_count

The number of duplicated DATA chunks that have been detected by MST on Endpoint A or Endpoint B of the specified association.

data_missing_a_count, data_missing_b_count

The number of missing DATA chunks detected by MST from Endpoint A or Endpoint B of the specified association.

data_oos_a_count, data_oos_b_count

The number of Out of Sequence DATA chunks detected by MST received by Endpoint A or Endpoint B of specified association.

buffer_overflow_a_count, buffer_overflow_b_count

The Algorithm Overflow Count which is the number of time that MST has had to flush its memory and send received chunks to the user's application with the consequence that the some chunks may be missing or out of sequence.

data_max_stored_a_count, data_max_stored_b_count

The maximum number of chunks that have been stored by MST at any time (during the measurement period) as part of the in sequence delivery algorithm.

restart_a_count, restart_b_count

The Association Restart Count which is the number of times that a change of Verification Tag has been detected by Endpoint A or Endpoint B of the specified association.

status

The following values can be returned in the message header status field:

Value	Error Code	Brief Description
0x00	MSTE_OK	Request accepted.
0x01	MSTE_BAD_ID	Invalid association_id
0x0B	MSTE_BAD_MSG_LEN	Invalid message length

4.4.5 MGT_MSG_R_THR_STATS - Throughput Statistics Request

Synopsis

Message sent to MST to request the current licensing throughput statistics.

Format

MESSAGE HEADER		
Field Name	Meaning	
type	MGT_MSG_R_THR_STATS (0x6f21)	
id	0	
src	Sending module_id	
dst	MST module_id	
rsp_req	Used to request a confirmation	
hclass	0	
status	0 – Read statistics 1 – Read and reset statistics	
err_info	0	
len	36	
PARAMETER AREA		
Offset	Size	Name
0	4	version
4	4	protocol
8	4	period
12	4	rx_data
16	4	tx_data
20	4	rx_rate_peak
24	4	tx_rate_peak
28	4	rate_peak
32	2	cong_count
34	2	enfmt_count

Description

This message is used to request throughput statistics for the MST module. The sending module should allocate a message with the correct length parameter area and set the 'version' parameter accordingly, the remaining parameters will be populated by MST and returned as a confirmation message.

The statistics can optionally be reset by setting the status field to 1.

Parameters

version

The version parameter tells the receiving module which format parameter area the sender expects to receive back. For the format described above the user should set version to zero.

protocol

The protocol parameter is set by MST to the value LICMOD_MST (5).

period

The period in seconds over which the statistics have been gathered.

rx_data

Unused by MST, set to zero.

tx_data

The amount of measured data, in Kbytes, sent by MST to the user application since the last statistics reset.

rx_peak_rate

Unused by MST, set to zero.

tx_peak_rate

The peak measured transmit data rate (averaged over a rolling thirty second time window) in units of bytes per second.

rate_peak

The peak measured data rate for both transmit and receive data combined (averaged over a rolling thirty second time window) in units of bytes per second.

cong_count

The number of times the congestion state has been entered since the last statistics reset.

enfnt_count

The enforcement count which is the number of times license enforcement has cut in to throttle the traffic rate since the last statistics reset.

4.4.6 MGT_MSG_R_LIC_STATUS - MST Licensing State Request

Synopsis

Message sent to MST to request the current license state.

Format

MESSAGE HEADER		
Field Name	Meaning	
type	MGT_MSG_R_LIC_STATUS (0x6f22)	
id	0	
src	Sending module_id	
dst	MST module_id	
rsp_req	Used to request a confirmation	
hclass	0	
status	Set to zero, unless module is enforcing throughput limiting in which case it is set to 1.	
err_info	0	
len	56	
PARAMETER AREA		
Offset	Size	Name
0	4	version
4	4	protocol
8	4	lic_thrp_rate
12	4	lic_links – reserved for future use
16	4	lic_sessions – reserved for future use
20	4	lic_options – reserved for future use
24	4	cfg_thrp_rate
28	4	cfg_max_links – reserved for future use
32	4	cfg_max_sessions – reserved for future use
36	4	cfg_options – reserved for future use
40	4	thrp_credit
44	4	cfg_links – reserved for future use
48	4	active_sessions – reserved for future use
52	4	thrp_cong_state

Description

This message is used to request the current license enforcement state for the MST module. The sending module should allocate a message with the correct length parameter area and set the 'version' and 'protocol' parameters accordingly, the remaining parameters will be populated by MST and returned as a confirmation message.

Parameters

version

The version parameter tells the receiving module which format parameter area the sender expects to receive back. For the format described above the user should set version to zero.

protocol

The protocol parameter is set by MST to the value LICMOD_MST (5) for MST.

lic_thrp_rate

The licensed throughput rate in Kbytes/s that the module is licensed to support.

cfg_thrp_rate

The maximum throughput rate requested when the module was configured using the MST_MSG_CONFIG message.

thrp_credit

The current Throughput Credit which is the number of bytes that can be received from the network or that can be sent to a User by MST before enforcement mechanisms are triggered.

thrp_cong_state

The current Throughput Congestion State for the module which is set to zero if there is no congestion or 1 if the module is in congestion.

4.5 Event Indication Messages

4.5.1 MST_MSG_FLCT_EVENT_IND - MST Flow Control Event Indication

Synopsis

Message sent by MST to notify management of a flow control event relating to the capacity of the internal MST message buffers.

Format

MESSAGE HEADER	
Field Name	Meaning
type	MST_MSG_FLCT_EVENT_IND (0x0e4a)
ld	0
src	MST Module
dst	Management Entity
rsp_req	0
hclass	0
status	event_id (see below)
err_info	0
len	0

Description

In order to provide in sequence delivery and also to limit the rate at which messages are issued to the application during periods of system congestion MST has the ability to buffer messages internally. In the case of the onset and abatement of congestion within the internal message buffers and in the event that messages need to be discarded, MST will issue this event notification message to the management module.

Parameters

event_id

The meaning of the event is conveyed as the event_id from the following table contained within the status field of the message header

event_id	Event Mnemonic	Description
1	MST_FLCT_CONGESTION_ABATE	Congestion has abated
2	MST_FLCT_CONGESTION_ONSET	Congestion has occurred
3	MST_FLCT_DISCARD	Message discard has occurred

4.5.2 MST_MSG_EVENT_IND - MST Management Event Indication

Synopsis

Message sent by MST to notify management of unexpected software events or formatting errors within received messages.

Format

MESSAGE HEADER		
Field Name	Meaning	
type	MST_MSG_EVENT_IND (0x0e4c)	
id	See below	
src	MST Module	
dst	Management Entity	
rsp_req	0	
hclass	0	
status	event_id	
err_info	0	
len	2	
PARAMETER AREA		
Offset	Size	Name
0	2	param1

Description

This message notifies management of unexpected events detected by MST. Unexpected events may be due to incorrect configuration of the module or receipt of invalid or badly formatted messages received on the monitoring port.

By default, whenever a management event indication is generated the message that caused the event is traced to the trace_mod_id configured in the module configuration message using the selective trace mechanism. If required the selective trace mechanism can be disabled on a per-event basis using the MST_MSG_SELTRACE_MASK message.

The Selective Trace Event Indication message is similar in format to the Trace Event Indication MGT_MSG_TRACE_EV (0x0003) as defined in the Software Environment Programmer's Manual with the exception that the message type is set to MGT_MSG_SEL_TRACE (0x0f16) and the status field in the header is set to the module-specific event_id (as defined in the table below) which caused the event to be generated.

Parameters

id, event_id, param1

The event_id in the status field indicates the type of event. The id field and the param1 parameter are then coded in an event-specific manner as detailed in the following table:

event_id	Event Mnemonic	Description	id	param1
0x01	MST_ASCFTR_BUFFER_OVERFLOW	Overflow of the internal buffer used for filtering an association.	association_id	endpoint_id
0x02	MST_IP_PKT_TRUNC	The amount of data received for an IP packet differs from that indicated in the IP header	port_id	0
0x03	MST_INVALID_PORT_ID	Virtual IP Message received for invalid port	port_id	0
0x04	MST_OVERLENGTH_PACKET	Overlength Virtual IP Message received.	port_id	len
0x05	MST_SHORT_PKT	A packet has been received which is too short	port_id	
0x06	MST_BAD_FMT	Badly formatted packet received	port_id	
0x07	MST_IP_FRAGMENT	Fragmented IP packet received	port_id	
0x08	MST_SCTP_FRAGMENT	Fragmented SCTP packet received	port_id	
0x09	MST_ETH_HDR_FMT	Malformed Ethernet header received	port_id	
0x0a	MST_IP_HDR_FMT	Malformed IP header received	port_id	
0x0b	MST_SCTP_HDR_FMT	Malformed SCTP header received	port_id	
0x0c	MST_SCTP_CHUNK_FMT	Malformed chunk header received	port_id	
0x0d	MST_SCTP_CHUNK_TYPE	Unknown chunk type received	port_id	
0x0e	MST_SCTP_DATA_CHUNK_FMT	Malformed data chunk header received	port_id	
0x0f	MST_COMMON_MSG_HDR_FMT	Malformed common message header received	port_id	
0x10	MST_M3UA_DATA_MSG_FMT	Malformed M3UA chunk received	port_id	
0x11	MST_M2PA_DATA_MSG_FMT	Malformed M2PA chunk received	port_id	

4.5.3 MGT_MSG_LIC_EVENT - License Event Indication

Synopsis

Message sent to management to indicate a license-related event.

Format

MESSAGE HEADER		
Field Name	Meaning	
type	MGT_MSG_LIC_EVENT (0x0f23)	
id	0	
src	MST module	
dst	Management module	
rsp_req	0	
hclass	0	
status	0	
err_info	0	
len	8	
PARAMETER AREA		
Offset	Size	Name
0	4	protocol_type – set to 5 for MST
4	2	event_type – set to 1 indicating 'Throughput' event
6	2	event_ind

Description

This message is used by modules, such as MST, that are licensed based on throughput to indicate when the throughput is getting close to the maximum permitted throughput (ie. congestion onset), recovered to a normal level (ie. Congestion abatement) and exceeded the permitted amount in which case enforcement commences.

Parameters

protocol_type, event_type

All indications sent by MST have protocol_type set to 5 and event_type set to 1 indicating 'Throughput Event'

event_ind

The possible event_ind values issued by MST are detailed in the following table:

event_ind	Description
0	ABATE – The traffic rate has reduced to be within the permitted licenced throughput.
1	CONGESTION – The traffic rate currently exceeds 60% of the permitted licensed throughput. If this condition persists then users should consider upgrading the licensed capacity to avoid chances of license enforcement.
2	ENFORCEMENT – The rate is consistently exceeding the permitted licensed throughput so the MST module has activated the enforcement mechanism which will limit the rate at which messages are delivered to the application and may result in message discard.

4.5.4 MST_MSG_IP_TRACE - MST Input Packet Trace Message

Synopsis

Message sent by MST for diagnostic purposes to trace a frame received from the internal IP stack.

Format

MESSAGE HEADER		
Field Name	Meaning	
type	MST_MSG_IP_TRACE (0xce4e)	
id	port_id	
src	MST module	
dst	Management module	
rsp_req	0	
hclass	0	
status	0 (1 if truncated)	
err_info	0	
len	Len	
PARAMETER AREA		
Offset	Size	Name
0	len	Message data starting with the IP header

Description

This messages generates a raw copy of the IP frame received from the underlying IP stack and sends it to the management module. It is intended for diagnostic purposes and is activated by sending a Trace Mask Configuration message (MGT_MSG_TRACE_MASK) to MST with the MST_MSG_IP_TRACE bit set in the ip_evt_mask. Care should be taken when activating tracing as all received packets will be traced which will cause additional load on the system.

Parameters

Message Data

Raw data as received from the link in Network byte order, starting with the IP header and continuing with the SCTP header, the SCTP chunk header and other data contained in the message as seen on the link.

4.6 Traffic Input Messages

4.6.1 MST_MSG_VPORT_PKT - Virtual Port Traffic Request

Synopsis

Message sent to MST to emulate messages usually received by MST from the underlying IP stack.

Format

MESSAGE HEADER		
Field Name	Meaning	
type	MST_MSG_VPORT_PKT (0xce4d)	
id	port_id	
src	Sending module_id	
dst	MST module_id	
rsp_req	0	
hclass	0	
status	0	
err_info	0	
len	Packet Length - Number of bytes in parameter area	
PARAMETER AREA		
Offset	Size	Name
0	len	Data - IP packet starting with the IP header

Description

The normal mode of operation for MST is to receive messages directly from the underlying IP stack. However MST is also capable of operating in a mode where the source packets are passed to MST within normal messages. To operate in this mode, first an MST port must be configured with port_type set to 'Virtual' then this message is used to supply MST with the incoming packets with the id field in the header indicating the port_id.

The parameter area starts with the IP header in network byte order unmodified as seen on the network. Any header in front of the IP header must be removed and not included in this message.

For messages with more than 320 bytes in the parameter area, MST supports the use of Long Messages as detailed in the *Software Environment Programmer's Manual*.

Parameters

port_id

The id field of the message identifies the MST port_id which must match a that of a port that has been configured with the port_type set to Virtual (PRTYPE_VIRTUAL, value=2). If port_id is invalid the message will be rejected and an event indication message with status set to MST_INVALID_PORT_ID will be generated.

Packet Length

The maximum permitted length of the IP packet is 1500 bytes; this corresponds to the maximum MTU size defined for Ethernet in IEEE 802.3. Messages in excess of this will be rejected and an event indication message with status set to `MST_OVERLENGTH_PACKET` will be generated.

If an `MST_MSG_VPORT_PKT` is received with the length field in the message header less than that in the IP header then the message is rejected and an event indication with status `MST_IP_PKT_TRUNC` is generated.

Data

Data as seen on the link in Network byte order, starting with the IP header and continuing with the SCTP header, the SCTP chunks (including the SCTP chunk header) and other data contained in the message as seen on the link.