Dialogic
Making Innovation Thrive™

Dialogic® NaturalAccess™ SIGTRAN Stack Developer's Reference Manual

July 2009 64-0451-01

www.dialogic.com

Copyright and legal notices

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 Corporation or its 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 medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

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 Corporation 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 9800 Cavendish Blvd., 5th Floor, Montreal, Quebec, Canada H4M 2V9. 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.

Any use case(s) shown and/or described herein represent one or more examples of the various ways, scenarios or environments in which Dialogic® products can be used. Such use case(s) are non-limiting and do not represent recommendations of Dialogic as to whether or how to use Dialogic products.

Dialogic, Dialogic Pro, Brooktrout, Diva, Cantata, SnowShore, Eicon, Eicon Networks, NMS Communications, NMS (stylized), Eiconcard, SIPcontrol, Diva ISDN, TruFax, Exnet, EXS, SwitchKit, N20, Making Innovation Thrive, Connecting to Growth, Video is the New Voice, Fusion, Vision, PacketMedia, NaturalAccess, NaturalCallControl, NaturalConference, NaturalFax and Shiva, among others as well as related logos, are either registered trademarks or trademarks of Dialogic Corporation or its 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 9800 Cavendish Blvd., 5th Floor, Montreal, Quebec, Canada H4M 2V9. 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 product 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.

Revision history

Revision	Release date	Notes
9000-62735-10	July 2008	DEH, SS7 5.0 Beta
9000-62735-11	October 2008	DEH, SS7 5.0
64-0451-01	July 2009	LBG, SS7 5.1
Last modified: July 7, 2009		

Refer to www.dialogic.com for product updates and for information about support policies, warranty information, and service offerings.

Table Of Contents

1 Introduction

The *Dialogic® NaturalAccess™ SIGTRAN Stack Developer's Reference Manual* explains how to use the Signaling Transport Protocol (SIGTRAN) to transport upper layer SS7 signaling packets over the IP network. This manual explains how to create applications using SIGTRAN and presents a detailed specification of its procedures and functions.

Note: The product to which this document pertains is part of the NMS Communications Platforms business that was sold by NMS Communications Corporation ("NMS") to Dialogic Corporation ("Dialogic") on December 8, 2008. Accordingly, certain terminology relating to the product has been changed. Below is a table indicating both terminology that was formerly associated with the product, as well as the new terminology by which the product is now known. This document is being published during a transition period; therefore, it may be that some of the former terminology will appear within the document, in which case the former terminology should be equated to the new terminology, and vice versa.

Former terminology	Current terminology
NMS SS7	Dialogic® NaturalAccess™ Signaling Software
Natural Access	Dialogic® NaturalAccess™ Software

2 SIGTRAN overview

Overview of Dialogic® NaturalAccess™ SIGTRAN Stack

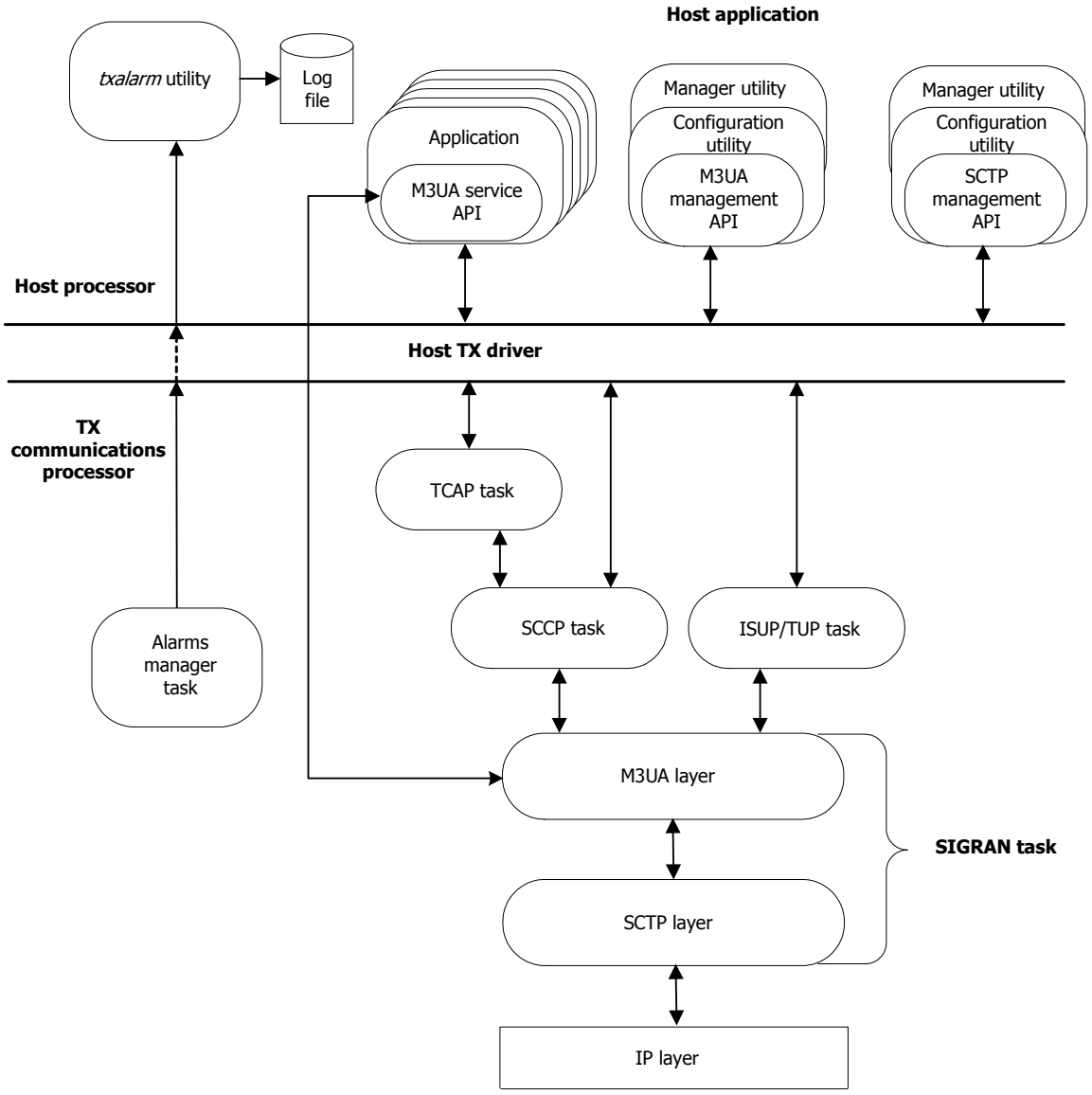
The Dialogic® NaturalAccess™ SIGTRAN Stack consists of the following SS7 layers:

SS7 layer	Description
M3UA	**MTP 3 user adaptation layer** An adaptation layer protocol that replaces the traditional SS7 MTP 3 layer in an IP network. It supports the transport of SS7 MTP 3 user signaling messages (such as ISUP and SCCP) over IP, using the services of SCTP. M3UA is used for communication between an application server process (ASP) and a signaling gateway process (SGP), or communication between two IP server processes (IPSPs). An ASP can serve as a media gateway controller (MGC) or IP-resident database. The M3UA implementation includes a data service API and a management API (MAPI). Host applications can use the service functions to transfer data, control flow, and obtain API statistics. They can use the management functions to configure and control M3UA entities, and to obtain status and statistical information from the M3UA layer. For more information, see *M3UA layer* on page 15.
SCTP	**Stream control transmission protocol** A reliable transport protocol that replaces the traditional SS7 MTP 2 layer in an IP network. It transports M3UA and higher layer SS7 signaling messages over IP networks. The SCTP implementation includes a management API (MAPI) that host applications can use to configure and control SCTP entities, and to obtain SCTP status and statistical information from the SCTP layer. For more information, see *SCTP layer* on page 18.

Note: The remainder of this manual refers to the Dialogic® NaturalAccess™ implementation of the SIGTRAN Stack as *SIGTRAN*.

SIGTRAN architecture

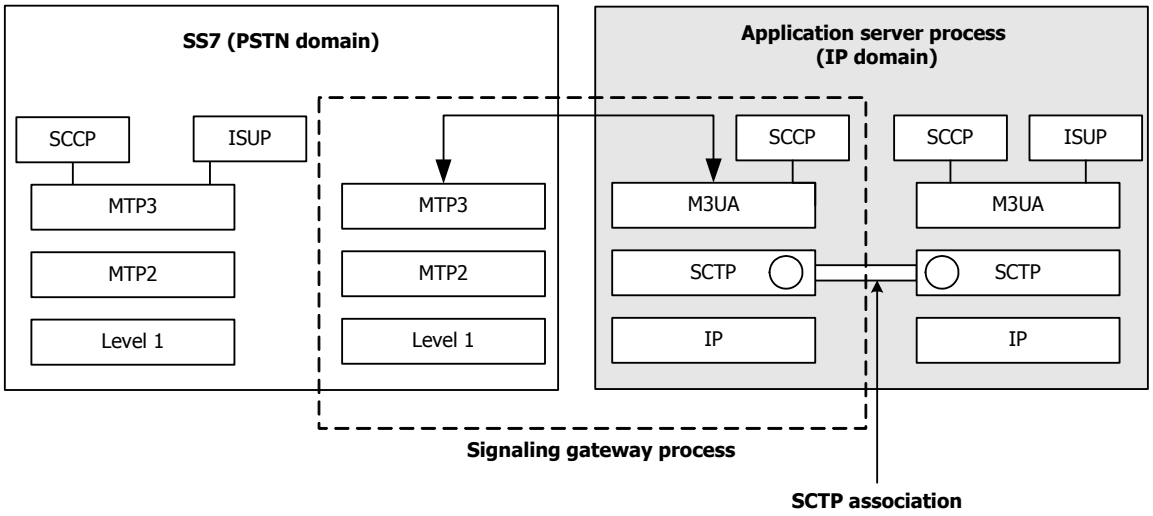
The following illustration shows the SIGTRAN high-level architecture:



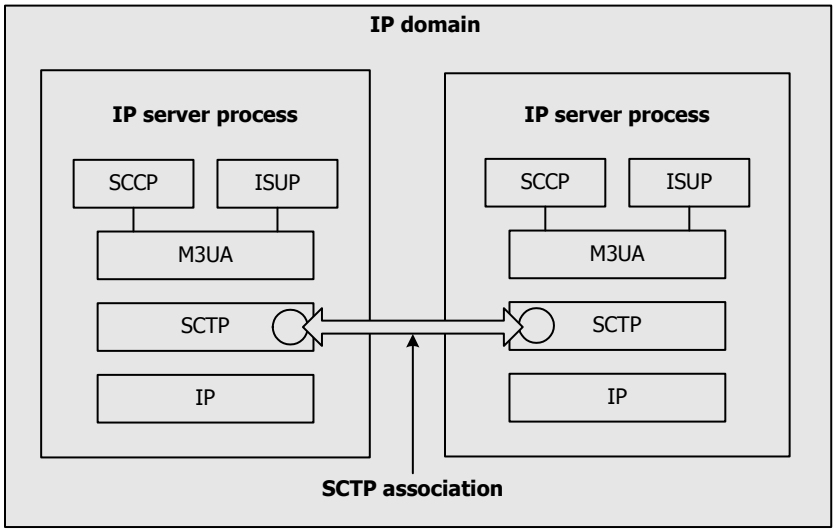
Two endpoints in a SIGTRAN network are connected through an SCTP association. The endpoints can be logically connected in the following ways:

- As nodes in a client-server relationship, where one node is an application server process (ASP) and the other is a signaling gateway process (SGP).
- As peer IP-only nodes, where each node is an IP server process (IPSP).

The following illustration shows an association that connects an ASP to a SGP:



The following illustration shows an association that connects endpoints from two IPSPs:



Associations are initiated automatically by SIGTRAN when the Ethernet interface becomes active. Dialogic® NaturalAccess™ SIGTRAN currently supports ASP and IPSP functionality. It cannot be used as an SGP.

SIGTRAN components

SIGTRAN consists of the following components:

Component	Purpose
M3UA layer	Provide a seamless replacement for MTP 3 to upper layer protocols, routes outgoing and incoming messages to specified destinations or applications, and reroutes traffic in the case of failure or congestion.
SCTP layer	Provides reliable transport for signaling.

Component	Purpose
TX alarms manager	Collects unsolicited alarms (status changes) generated by the SS7 tasks and forwards them to the host for application-specific alarm processing.
Operating system-independent TX device driver	Provides low-level access to the TX board from the host computer.
M3UA configuration program	Reads the M3UA configuration file and loads the configuration to the M3UA layer at system startup. For information, see the *Dialogic® NaturalAccess™ Signaling Software Configuration Manual*.
M3UA service API	Functions that get and send data, control data flow, and return API statistics. For information, see *M3UA service function summary* on page 41.
M3UA management API	Functions that initialize, set, and return configuration parameters for M3UA entities. These functions can also control M3UA entities and return status and statistical information for these entities. For information, see *M3UA management function summary* on page 57.
SCTP configuration program	Reads the SCTP configuration file and loads the configuration to the SCTP layer at system startup. For information, see the *Dialogic® NaturalAccess™ Signaling Software Configuration Manual*.
SCTP management API	Functions that initialize, set, and return configuration parameters for SCTP entities. These functions can also control SCTP entities and return status and statistical information for these entities. For information, see *SCTP management function summary* on page 153.
M3UA sample application	Sample application that uses the M3UA service API to send and receive data, provided in source and object code formats.
M3UA management utility (*m3uamgr*)	Utility that uses the M3UA management API, provided in source code and object code formats. For information, see *m3uamgr overview* on page 139.
SCTP management utility (*sctpmgr*)	Utility that uses the SCTP management API, provided in source code and object code formats. For information, see *sctpmgr overview* on page 197.
txalarm utility	Utility that captures alarms from the board and optionally writes them to a log file.
CPK/OS operating system	Proprietary operating system for the TX series boards. Includes utilities for troubleshooting and statistics gathering. For more information, see the *TX Utilities Manual*.

M3UA layer

The M3UA layer enables MTP 3-like functionality to be performed over IP. It performs these primary tasks:

Task	Description
Interface to the higher and lower SS7 layers	Provides an interface between the higher SS7 layers (ISUP, SCCP, and TUP), and the lower SCTP layer. This interface is implemented with a set of messages that the application can pass to the M3UA layer on the TX communications processor. For more information, see *M3UA service function summary* on page 41.
Routing	Routes messages to their IP destinations. M3UA uses a flexible configuration capable of supporting a wide variety of network routing and addressing requirements. For more information, see *M3UA message handling* on page 21.
Signaling network management	Maintains the availability status of all destinations through all routes. Automatically reroutes traffic in the case of failure or congestion.

In addition to general parameters, the M3UA layer consists of the following entities:

- Network definitions
- NSAPs
- SCT SAP
- Peer servers
- Peer signaling processes
- SCTP associations
- Routing keys

Network definitions

A network definition is a logical network that describes the signaling traffic between two IP signaling points (IPSPs) or an application server process (ASP) and a signaling gateway (SG) over a common SCTP association. Typically only one network definition is required, unless the local node supports both ANSI and ITU networks.

The M3UA management API lets you perform the following actions on a network definition:

- Initialize configuration parameters
- Configure a network for M3UA
- Obtain configuration information

For more information, see *M3UA management function summary* on page 57.

NSAPs

Network service access points (NSAPs) define the upper layer SS7 applications that use M3UA. Each NSAP is associated with a service indicator field and protocol variant combination.

If multiple protocol variants (network definitions) or multiple upper layers must be supported on the same M3UA instance (same board), an NSAP is required for each

SIO value and protocol variant. A single application can associate itself with multiple NSAPs, or a separate application can bind to each NSAP.

The M3UA management API lets you perform the following actions on an NSAP:

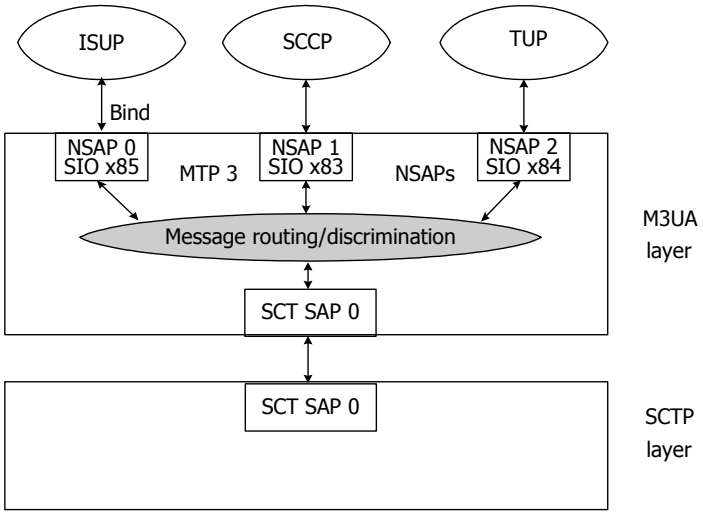
- Initialize configuration parameters
- Configure the NSAP for M3UA
- Obtain configuration information
- Obtain status information
- Obtain statistical information

For more information, see *M3UA management function summary* on page 57.

SCT SAP

An SCT service access point (SCT SAP) defines the interface between M3UA and SCTP. It is the lower SAP for M3UA and the upper SAP for SCTP. Only one SCT SAP is defined.

The following illustration shows three NSAPS and one SCT SAP defined for an M3UA layer:



The M3UA management API lets you perform the following actions on the SCT SAP:

- Initialize configuration parameters
- Configure the SCT SAP for M3UA
- Obtain configuration information
- Obtain status information
- Obtain statistical information

For more information, see *M3UA management function summary* on page 57.

Peer servers

A peer server (PS) is a logical entity on the IP network, such as a virtual switch or a database element, that is served by one or more peer signaling processes (PSPs). Each peer server serves a specific routing key. For example, a peer server can handle a signaling relation identified by a DPC/OPC combination or an SIO/DPC/OPC combination. There is a one-to-one relationship between a peer server and a routing key. Peer servers can be local or remote.

The M3UA API lets you perform the following actions on a peer server:

- Initialize configuration parameters
- Configure the peer server
- Obtain configuration information
- Obtain status information
- Obtain statistical information

For more information, see *M3UA management function summary* on page 57.

Peer signaling processes

A peer signaling process (PSP) is used to describe a remote SGP or IPSP that is, or will be, accessible from the local M3UA through an association. It is an executing process that handles signaling traffic for one or more peer servers/routing keys. A peer signaling process can be in an active or standby state.

The M3UA API lets you perform the following actions on a peer signaling process:

- Initialize configuration parameters
- Configure the peer signaling process
- Obtain configuration information
- Obtain status information
- Obtain statistical information
- Obtain the number of dynamic routing keys associated with the peer signaling process
- View the status of the SCTP association by which the remote peer signaling process is reachable.

Note: The local M3UA is a special case and is referred to as the local PSP. This local PSP is created automatically, not configured.

For more information, see *M3UA management function summary* on page 57.

SCTP associations

An SCTP association is a logical relationship between two SCTP endpoints that is used to transport M3UA user protocol messages between two peer servers. The transport addresses used by the endpoints in the association uniquely identify that association. Two SCTP endpoints cannot have more than one SCTP association between them at any given time. Each M3UA instance can support up to 254 associations.

The Dialogic® NaturalAccess™ M3UA API lets you perform the following actions on an association:

- Establish an association
- Terminate an association
- Inhibit, to temporarily remove the association from service
- Uninhibit, to restore the association to service
- Obtain the status of associations to determine which are active

For more information, see *M3UA management function summary* on page 57.

Routing keys

A routing key defines the range of signaling traffic to be handled by a particular peer server. Routing keys are defined using combinations of the following parameters:

- Destination point code (DPC) (the minimum requirement)
- Origination point code (OPC)
- Service indicator (SI)

The M3UA API lets you perform the following actions on a routing key:

- Initialize configuration parameters
- Configure the routing key
- Obtain configuration information
- Obtain status information

For more information, see *M3UA management function summary* on page 57.

SCTP layer

The SCTP layer provides transport functionality for higher protocol layers over IP. It is responsible for the reliable transfer of M3UA and higher level SS7 messages between two IP endpoints.

STCP message tasks

SCTP consists of messages that govern communications between SCTP functions and the lower layers on the TX board. The messages perform the following tasks:

- Binding
- Establishing an SCTP association
- Transferring messages

Binding

The binding phase establishes the higher layer application (M3UA) as the user of the SCTP interface, an SCT SAP. M3UA sends a single bind request message to SCTP, for which there is a bind confirmation response. Similarly, SCTP binds with its lower layer (IP) through a transport SAP (TSAP). This binding is performed automatically by the SCTP and M3UA layers at startup.

Establishing an SCTP association

The application establishes an SCTP association when directed to by M3UA. It initiates a four message handshake with the peer endpoint. When completed successfully, SCTP sends a connection up indication to M3UA.

Transferring messages

After the SCTP layer returns a connection up message to M3UA, M3UA initiates the sending of ASPUP (ASP UP) and ASPAC (ASP Active) messages to allow the transfer of higher level messages.

M3UA sends a data request message to SCTP to request transmission of an SS7 packet on a particular association. When the message is acknowledged by the far exchange, there is no corresponding notification sent to the application.

SCTP sends a data indication message to notify M3UA of an incoming data packet.

Configurable entities

In addition to general parameters, the SCTP layer consists of the following configurable entities:

- SCT service access point (SCT SAP)
- Transport service access point (TSAP)

SCT service access point (SCT SAP)

An SCT service access point (SCT SAP) defines the interface between SCTP and M3UA. It is the upper SAP for SCTP and the lower SAP for M3UA. Only one SCT SAP is defined for an SCTP layer.

The SCTP API lets you perform the following actions on the SCT SAP:

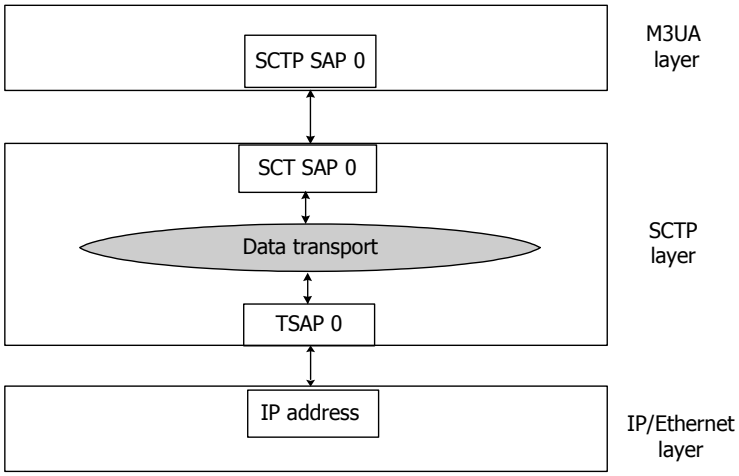
- Initialize configuration parameters
- Configure the SCT SAP for SCTP
- Obtain configuration information
- Obtain status information
- Obtain statistical information

For more information, see *SCTP management function summary* on page 153.

Transport service access point (TSAP)

A transport service access point (TSAP) defines the interface between SCTP and the IP/Ethernet layer. It is the lower SAP for SCTP. Only one TSAP is defined for an SCTP layer.

The following illustration shows the SCT SAP and TSAP defined for SCTP:



The SCTP API lets you perform the following actions on the TSAP:

- Initialize configuration parameters
- Configure the TSAP for SCTP
- Obtain configuration information
- Obtain status information
- Obtain statistical information

For more information, see *SCTP management function summary* on page 153.

3 SIGTRAN programming model

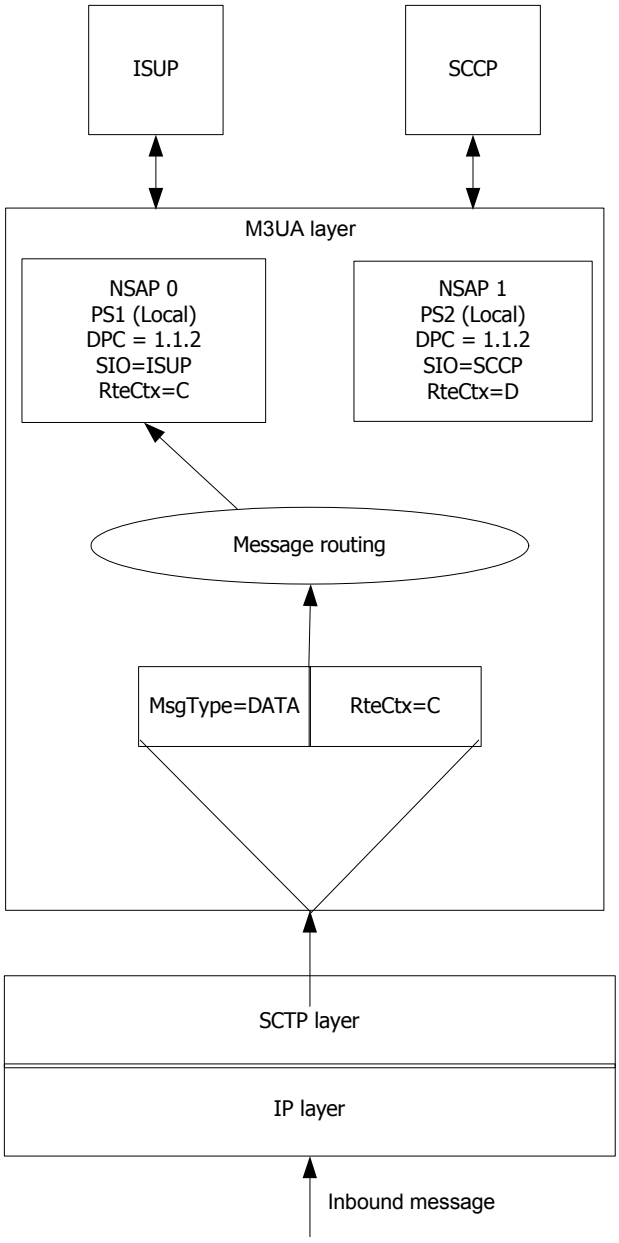
M3UA message handling

M3UA validates and routes both inbound and outbound messages. This topic describes each data flow direction.

Inbound messages

When an inbound message is received, M3UA finds a local peer server that matches the routing context in the received message. If the peer server is associated with an NSAP through a routing key, the message is routed over that NSAP to the appropriate M3UA user. In some error cases, such as a message received over an inactive association or with an invalid routing context, the message is discarded and an error message is returned to the sending side. In all other error cases, the message is discarded without generating a message.

The following illustration shows inbound message distribution for M3UA:



Outbound messages

Message routing for outbound messages originated by local user parts or applications is based on the following combinations of DPC, OPC, and SIO values:

- DPC
- DPC and OPC
- DPC, OPC, and SIO
- DPC and SIO

When an outbound message is ready for routing, M3UA searches its routing tables for a route that matches the DPC specified for the message (and optionally the OPC and SIO, depending on the values and masks configured). If a matching route is found for the message and the associated peer server is remote, the message is routed over the association for the peer signaling process associated with the peer server.

If a route is not found, an error status indication (NO_ROUTE_FOUND) is generated for the application**.** If the route is found but unavailable, a user part unavailable (UPU) status indication is generated for the application.

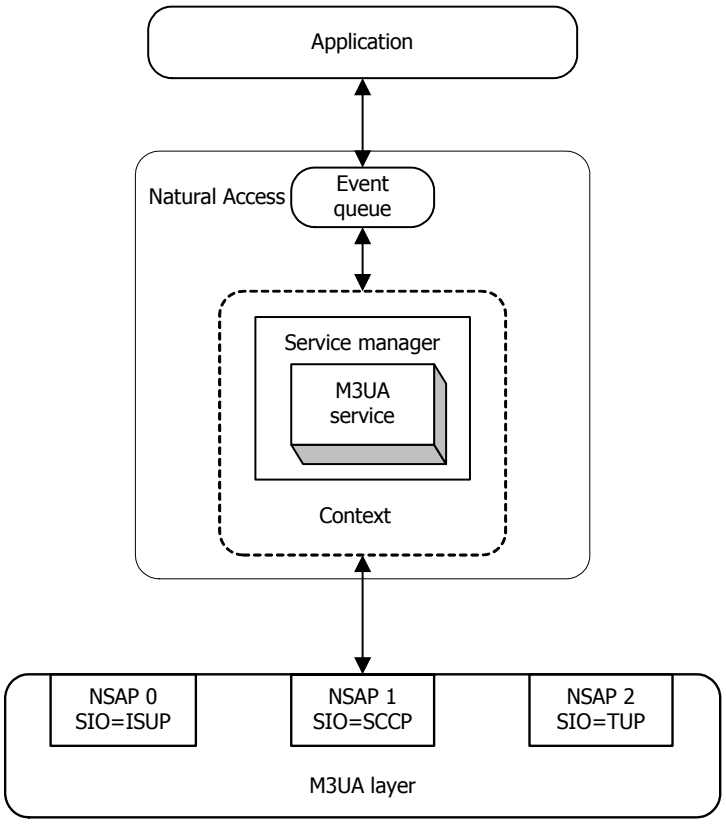
Contexts and queues

Natural Access organizes services and their associated resources around a processing object known as a context. Each instance of an application binding to an M3UA NSAP is a unique Natural Access context. Contexts are created with **ctaCreateContext**.

A Natural Access queue delivers all events and messages from the M3UA layer to the application. Queues are created with **ctaCreateQueue**. Each context is associated with a single queue through which all events and messages belonging to that context are distributed. More than one context can be assigned to the same queue.

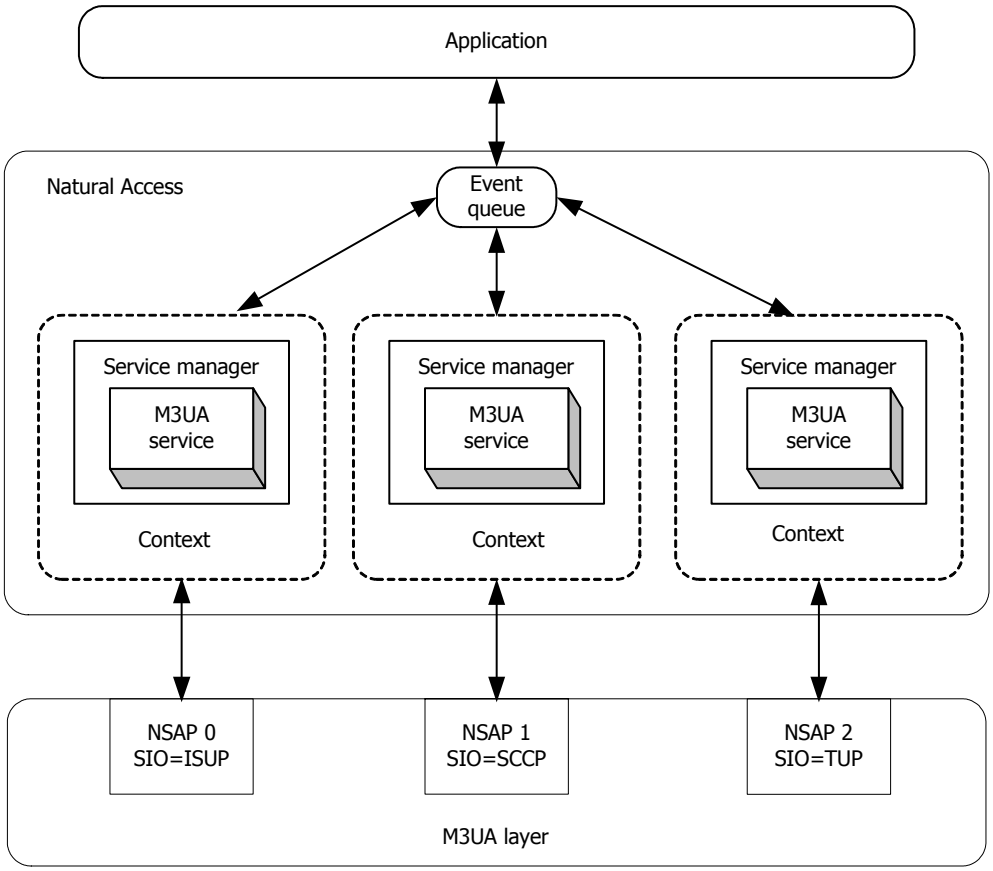
Different application programming models are possible depending on how many M3UA NSAPs (subsystems) are implemented by the application and how the application is organized.

An application that uses a single M3UA NSAP uses a single context, single queue model, as shown in the following illustration:



For a single threaded application that uses multiple M3UA NSAPs (multiple M3UA subsystems), a multiple context, single queue model is recommended. The application has a single event loop with events from all SAPs delivered through the same queue. The application determines which SAP a particular event is associated with from a service user ID (suID) value returned with each event.

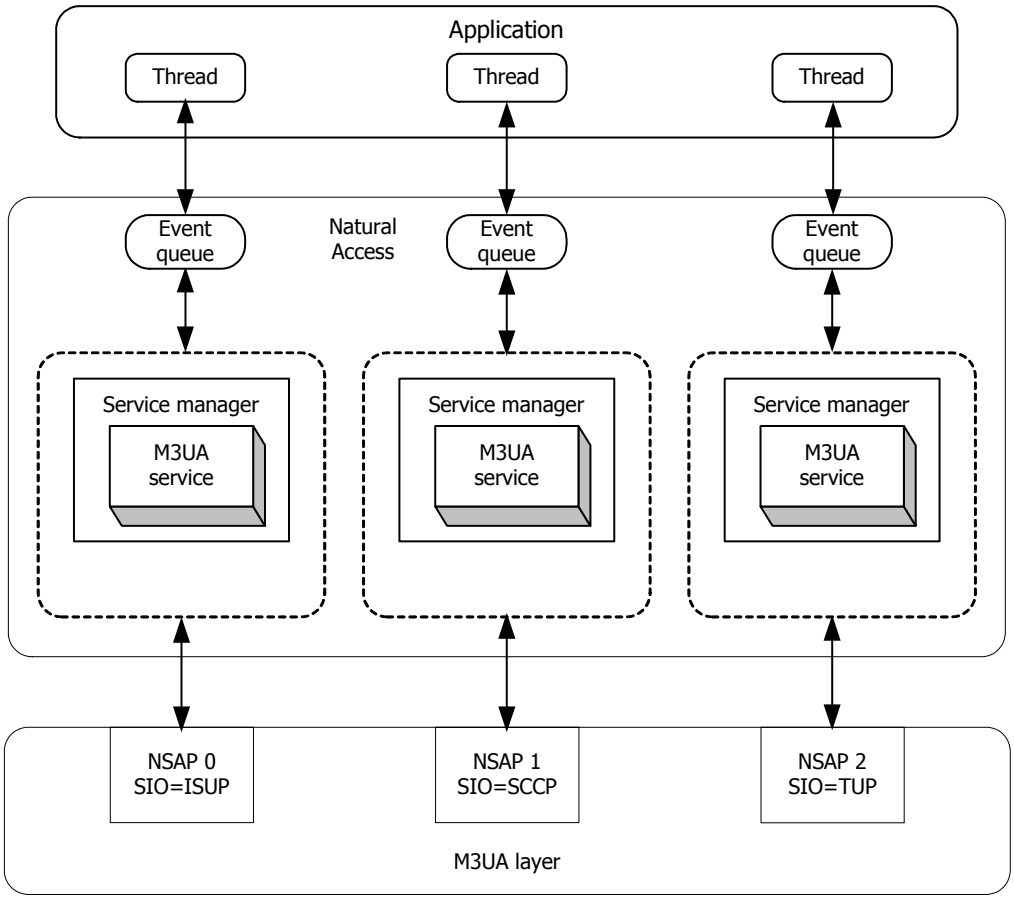
The following illustration shows a multiple context, single queue model:



For multi-threaded applications using multiple M3UA NSAPs (one per thread), a multiple context, multiple queue model is recommended. Each thread has its own independent event loop, receiving only the events associated with its M3UA NSAP on its Natural Access queue.

For this programming model, each thread and event queue must be assigned its own entity ID, unique among all applications on that host accessing any of the SS7 services.

The following illustration shows a multiple context, multiple queue model:



M3UA service users

The M3UA data interface supports one or more applications using network service access points (NSAPs). One NSAP is defined for each application that uses the M3UA service. An application binds to a particular NSAP at initialization time, specifying the NSAP ID to which it wants to bind. Each NSAP is associated with a service information octet (SIO) value, which in turn identifies the upper task protocol in use on that NSAP (for example ISUP, TUP, SCCP). Therefore, only one application process can handle incoming messages for a particular upper task protocol (only one process receiving incoming ISUP messages).

The M3UA configuration file specifies the number of upper NSAPs and the characteristics of each upper NSAP. For information, see *M3UANsapCfg* on page 112 and the *Dialogic® NaturalAccess™ Signaling Software Configuration Manual*.

Entity and instance IDs

Each application must have a unique entity and instance ID to route messages between the processes in the system. Entity IDs are single byte values in the range of 0x00 through 0xFF that represent a specific process. Allocate entity IDs as follows:

Range	Usage
0x00 through 0x1F 0x80 through 0xFF	Reserved for system utilities, configuration utilities, and management utilities.
0x20 through 0x7F	Reserved for applications.

Instance IDs identify the processor on which the entity executes. The host is always processor 0 (zero). All host-resident M3UA applications must be coded to zero. All tasks on TX board 1 receive an instance ID of 1, all tasks on TX board 2 receive an instance ID of 2, and so on.

Transferring data

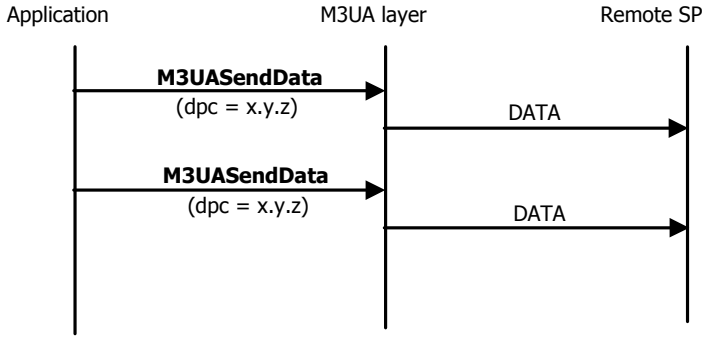
After the SCTP layer returns a connection up message to M3UA, M3UA sends ASPUP (ASP Up) and ASPAC (ASP Active) messages to allow the transfer of higher-level messages.

M3UA sends a data request message to SCTP to request transmission of an SS7 packet. When a message or messages are acknowledged by the far exchange, there is no corresponding notification sent to M3UA.

When it receives an M3UA message, SCTP sends a data indication message to notify M3UA of an incoming data packet.

After an application binds to the M3UA layer and receives a RESUME for a given destination, it can start sending data to that destination using **M3UASendData**. If the call succeeds, but later the message is undeliverable, the application receives a status indication describing the cause of failure. When the message is successfully delivered, no status indication is sent.

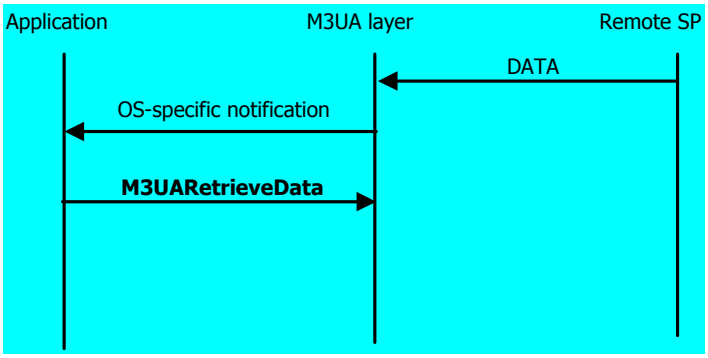
The following illustration shows how an application sends data:



Asynchronous notification and polling are two methods for an application to receive incoming data or status indications. Polling requires the application to call **M3UARetrieveMessage** to continually check for incoming messages. The application must call this function regularly to avoid excessive queuing of messages in the TX driver or the M3UA layer. **M3UARetrieveMessage** returns M3UA_NO_MSG until a message is available. M3UA returns M3UA_SUCCESS when a message is available.

An application can use an operating system-specific function, such as **WaitForMultipleObjects** in Windows, to set up asynchronous notifications that are triggered when a message is available. In this way, the application only calls **M3UARetrieveData** when a data or status indication is waiting to be read.

The following illustration shows the asynchronous notification method of receiving data:



Status indications

An application can receive status indications through asynchronous notification or polling. A status indication is sent to all applications when the corresponding network status changes. The following table describes the status indications:

Status indication	Description
StatPaused	Delivery to the specified destination point code is not currently possible. This can mean that there is no route configured for the destination point code or that all routes to that point code are currently unavailable.
StatResumed	Occurs initially or after a StatPaused indication when one or more routes to a particular destination point code become available.
StatCongested	Delivery to the destination SP failed due to network congestion at some point along the route. The user part or application must reduce traffic to that destination.
StatCongestionEnds	Occurs after a StatCongested indication when network congestion has abated.
StatUsrUnavail	Message was delivered to the remote M3UA layer, but no application was registered to receive data with the service information octet (SIO) contained in the message. For example, data was sent with an SIO indicating an ISUP message, but there was no ISUP application running at the destination. The remote M3UA layer discards the message.
StatPrimary	Local M3UA is the primary node in a redundant configuration.
StatBackup	Local M3UA is the backup node in a redundant configuration.
StatStandAlone	M3UA is not in a redundant configuration.

Controlling congestion

The NaturalAccess SIGTRAN Stack has queues and congestion thresholds at several points during the outbound flow of messages. There are internal thresholds at the IP layer and configurable thresholds at the SCTP and M3UA layers.

The following table describes how the IP, SCTP, and M3UA layers control congestion:

Layer	Trigger	Description
IP	Traffic exceeds the internal upper transmit threshold.	IP layer notifies the SCTP layer, which begins queuing messages.
	Transmit queue size is reduced to the lower transmit threshold.	IP layer notifies the SCTP layer, which resumes transmitting messages.
SCTP	Number of queued packets exceeds the configured value FLOW_START_THRESH.	SCTP sends a flow control start indication to M3UA.
	Number of queued packets is reduced to the configured value FLOW_STOP_THRESH.	SCTP sends a flow control stop indication to M3UA.
M3UA	Receipt of a flow control start indication from SCTP.	M3UA sets the congestion active flag for the association and begins polling SCTP periodically about its flow control status. While an association is in flow control, M3UA queues future data requests for that association until QUEUE_SIZE is reached. At that point, future messages for the association are discarded. M3UA propagates the association congestion information to the individual remote DPCs that are part of the association.
	Receipt of a flow control stop indication from SCTP.	M3UA clears the congestion flag for the association and stops polling SCTP. M3UA then transmits anything in its congestion queue for that association to SCTP.

In addition to internal congestion monitoring, M3UA maintains the congestion state of individual remote DPCs, as would an MTP3 layer. Congestion level indications for these DPCs are passed in a status indication to the bound upper layer, as with MTP3. It is up to the application to further reduce traffic as the congestion level increases in order to alleviate the congestion.

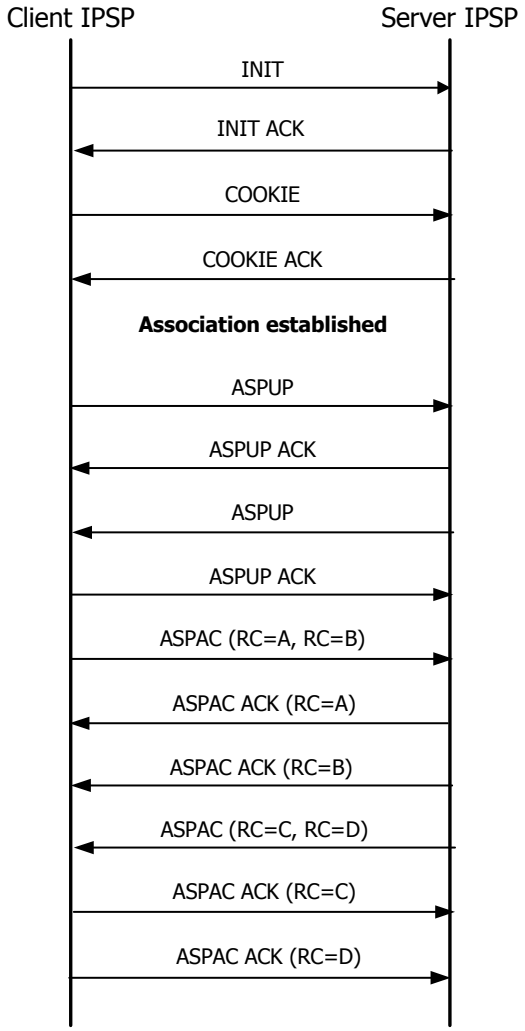
Establishing a connection

The M3UA layer creates a connection between a local and remote endpoint by establishing an SCTP association between the endpoints, and then exchanging M3UA management messages that specify the state of the peer server process (PSP) and the routing keys (PS). This message exchange occurs when the Ethernet port is activated.

The message exchange used to establish a connection differs depending on the endpoint configurations.

IPSP-IPSP configuration in double-ended mode

The following illustration shows the message flow used to establish a connection between two endpoints configured as IPSPs in double-ended mode (DE), where each side supports two route contexts:



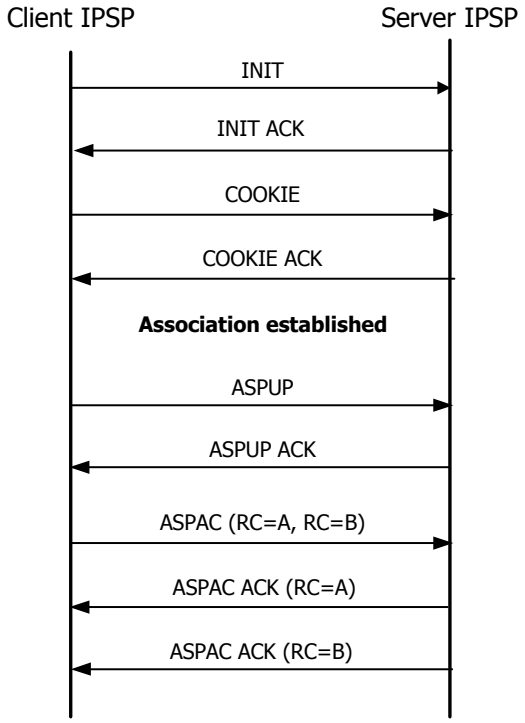
The following table describes the process for establishing a connection between two IPSP endpoints in DE mode:

Stage	Description
1	Client IPSP initiates an SCTP association by sending an INIT when the Ethernet port becomes active (which occurs when electrical connectivity is established).
2	M3UA layer establishes an SCTP association between the local and remote endpoints using INIT (Initiation), INIT ACK (Initiation Acknowledgement), COOKIE (State Cookie), and COOKIE ACK (Cookie Acknowledgment) messages. The association state changes from DOWN to ACTIVE (as displayed in the *m3uamgr* command: status psp <***n***>).
3	Each IPSP sends an ASPUP (ASP Up) message to notify the other side that the PSP is up. The ASP State on each side changes from DOWN to INACTIVE for each remote route context (as displayed in the *m3uamgr* command: status psp <***n***>).

Stage	Description
4	Each IPSP receiving an ASPUP responds with an ASPUP ACK message.
5	Each IPSP sends NOTIFY messages of type ASCHG (AS State Change) with new state INACTIVE for each route context. For readability, these messages are not shown in the message flow illustration.
6	Each IPSP sends an ASPAC (ASP Active) message for each active local route context. A local route is considered active when the following conditions are met: • An ASPUP message was sent and acknowledged. • A NOW-PRIMARY or NOW-STANDALONE event was received by the IPSP. • An M3UA-user has bound to the corresponding NSAP. The ASP State on the receiving side changes from INACTIVE to ACTIVE for each remote route context in the ASPAC message (as displayed in the *m3uamgr* command: status psp <***n***>). The sending of an ASPAC message indicates that the sending IPSP can now receive traffic related to the routing contexts identified in the message. The receipt of an ASPAC message indicates that the receiving IPSP can now send traffic related to the routing contexts identified in the message.
7	Each IPSP receiving an ASPAC responds with an ASPAC ACK message for each routing context.
8	Each IPSP sends NOTIFY messages of type ASCHG with new state ACTIVE for each route context. For readability, these messages are not shown in the flow illustration.

IPSP-IPSP configuration in single-ended mode

The following illustration shows the message flow for establishing a connection between two IPSPs in single-ended mode (SE):



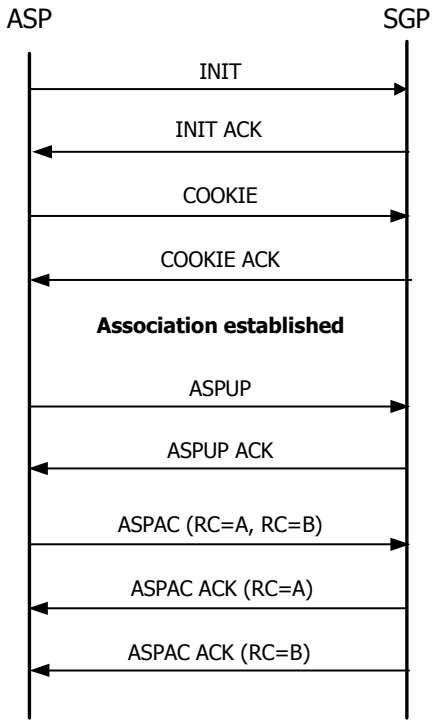
For endpoints configured as IPSPs in single-ended (SE) mode, only one side is required to send the ASPUP and ASPAC messages. In this configuration, the routing contexts must be configured identically on both sides.

The following table describes the process for establishing a connection between two IPSP endpoints in SE mode:

Stage	Description
1	Client IPSP initiates an SCTP association by sending an INIT when the Ethernet port becomes active (which occurs when electrical connectivity is established).
2	M3UA layer establishes an SCTP association between the local and remote endpoints using INT (Initiation), INT ACK (Initiation Acknowledgement), COOKIE (State Cookie), and COOKIE ACK (Cookie Acknowledgment) messages. The association State changes from DOWN to ACTIVE (as displayed in the *m3uamgr* command: status psp <***n***>).
3	Client IPSP sends an ASPUP (ASP Up) message to notify the other side that the PSP is up. The ASP State on each side changes from DOWN to INACTIVE (as displayed in the *m3uamgr* command: status psp <***n***>).
4	Each IPSP receiving an ASPUP responds with an ASPUP ACK message.
5	Each side sends NOTIFY messages of type ASCHG (AS State Change) with new state INACTIVE for each route context. For readability, these messages are not shown in the message flow illustration.
6	Client IPSP sends an ASPAC (ASP Active) message for each active local route context. A local route is considered active when the following conditions are met: • An ASPUP message was sent and acknowledged. • A NOW-PRIMARY or NOW-STANDALONE event was received by the IPSP. • An M3UA-user has bound to the corresponding NSAP. The ASP State on the each side changes from INACTIVE to ACTIVE for each remote route context in the ASPAC message (as displayed in the *m3uamgr* command: status psp <***n***>). The sending of an ASPAC message indicates that the sending IPSP can now receive traffic related to the routing contexts identified in the message. The receipt of an ASPAC message indicates that the receiving IPSP can now send traffic related to the routing contexts identified in the message.
7	Each IPSP receiving an ASPAC responds with an ASPAC ACK message for each routing context.
8	Each side sends NOTIFY messages of type ASCHG with new state ACTIVE for each route context. For readability, these messages are not shown in the flow illustration.

ASP-SGP configuration

The following illustration shows the message flow used to establish a connection between an endpoint configured as an ASP and an endpoint configured as an SGP:



The following table describes the process for establishing a connection between an ASP endpoint and an SGP endpoint:

Stage	Description
1	ASP initiates an SCTP association by sending an INIT when the Ethernet port becomes active (which occurs when electrical connectivity is established).
2	M3UA layer establishes an SCTP association between the local and remote endpoints using INT (Initiation), INT ACK (Initiation Acknowledgement), COOKIE (State Cookie), and COOKIE ACK (Cookie Acknowledgment) messages. The association State changes from DOWN to ACTIVE (as displayed in the *m3uamgr* command: status psp <***n***>).
3	ASP sends an ASPUP (ASP Up) message to notify the other side that the PSP is up. The ASP state on each side changes from DOWN to INACTIVE for each remote route context (as displayed in the *m3uamgr* command: status psp <***n***>).
4	Each IPSP receiving an ASPUP responds with an ASPUP ACK message.
5	SGP sends NOTIFY messages of type ASCHG (AS State Change) with new state INACTIVE for each route context. For readability, these messages are not shown in the message flow illustration.

Stage	Description
6	ASP sends one or more ASPAC (ASP Active) messages that identify its remote route contexts. These message are sent after the following conditions are met: • An ASPUP message was sent and acknowledged. • A NOW-PRIMARY or NOW-STANDALONE event was received by the ASP. • An M3UA-user has bound to the corresponding NSAP. The ASP State for each side changes from INACTIVE to ACTIVE for each remote route context in the ASPAC message (as displayed in the *m3uamgr* command: status psp <***n***>). The sending of an ASPAC message indicates that the ASP can now receive traffic related to the routing contexts identified in the message. The receipt of an ASPAC message indicates that the SGP can now send traffic related to the routing contexts identified in the message.
7	Each IPSP receiving an ASPAC responds with an ASPAC ACK message for each routing context.
8	SGP sends NOTIFY messages of type ASCHG with new state ACTIVE for each route context. For readability, these messages are not shown in the flow illustration.

Redundant configuration

The message flow for a redundant configuration differs from that for a non-redundant configuration in the following ways:

- Both the primary and backup PSP establish associations and send the ASPUP message.
- Only the primary PSP sends ASPAC messages.
- When a switchover occurs, the PSP that went from primary to backup sends ASPIA (ASP Inactive) messages for all local route contexts, indicating that it can no longer receive traffic related to those route contexts. The PSP that went from backup to primary sends ASPAC messages for all local route contexts, indicating that it will now receive traffic related to those route contexts. The receiver of these messages changes the association over which the related traffic is transferred.

As with the non-redundant configurations, NOTIFY (ASCHG) messages are sent by an IPSP or SGP upon receipt of ASPAC or ASPIA messages.

4 Using the M3UA service

Initializing the M3UA service under Natural Access

M3UA data functions are implemented as a Natural Access service. Natural Access is a development environment for telephony and signaling applications. It provides a standard application programming interface for services, such as signaling protocol stacks, independent of the underlying hardware.

Natural Access is described in detail in the *Natural Access Developer's Reference Manual*. Understanding the basic Natural Access programming concepts, including services, queues, contexts, and asynchronous events, is critical to developing applications that utilize the M3UA service.

Before calling any M3UA service functions, the application must:

1. Initialize the Natural Access run-time environment.
2. Create the desired queues and contexts.
3. Open the M3UA service to bind it to the desired M3UA service access points (SAPs).

Initializing the Natural Access environment

Initialize the Natural Access environment by calling **ctaInitialize** once per application, regardless of the number of queues and contexts to be created:

```
CTA_INIT_PARMS      M3UAInitparms       = {0};
CTA_SERVICE_NAME    M3UAServiceNames[]  = {{"M3UA", "M3UAMGR"}};
...
M3UAInitparms.size              = sizeof(CTA_INIT_PARMS);
M3UAInitparms.traceflags        = CTA_TRACE_ENABLE;
M3UAInitparms.parmflags         = CTA_PARM_MGMT_SHARED;
M3UAInitparms.ctacompatlevel    = CTA_COMPATLEVEL;

Ret = ctaInitialize(M3UAServiceNames, 1, &M3UAInitparms);
if (Ret != SUCCESS)
{
    printf("ERROR code 0x%08x initializing Natural Access.", Ret);
    exit( 1 );
}
```

Creating queues and contexts

The application creates the required Natural Access queues and contexts, as described in *Contexts and queues* on page 23. The queue must always be created before any context associated with it.

```
CTAHD       ctaHd;                  /* CTA context handle */
CTAQUEUEHD  ctaQueue;               /* Queue */
...

Ret = ctaCreateQueue( NULL, 0, &ctaQueue );
if ( Ret != SUCCESS )
{
     ctaGetText( NULL_CTAHD, Ret, sErr, sizeof( sErr ) );
     printf( "*ERROR : ctaCreateQueue failed( %s )\n", sErr );
     ...
}

sprintf( contextName, "M3UASAP-%d", spId ); /* context name is optional */

Ret = ctaCreateContext( ctaQueue, spId, contextName, &ctaHd );
if ( Ret != SUCCESS )
{
     ctaGetText( NULL_CTAHD, Ret, sErr, sizeof( sErr ) );
     printf( "ERROR : ctaCreateContext failed( %s )\n", sErr );
     ctaDestroyQueue( pSap->ctaQueue );
     ...
}
```

Using ctaOpenServices

After the queues and contexts are created, the application must bind itself to each desired M3UA NSAP by calling **ctaOpenServices** once for each binding. The binding operation specifies the following parameters:

Parameter	Description
board	TX board number.
srvInfo	Service information octet.
sapId	M3UA NSAP ID (defined in configuration) on which to bind. This parameter must be passed in all functions that make requests to the board.
srcEnt	Calling application entity ID.
srcInst	Calling application instance ID.
suId	Calling application service user ID. This parameter is passed up in future indications from the board.
poolsize	Number of messages allowed to be queued to the TX board. Default value is 256.

Under Natural Access, these parameters are specified in the CTA_SERVICE_ARGS structure, contained in the CTA_SERVICE_DESC structure. An example of the parameter specification is provided:

```
CTA_SERVICE_DESC M3UAOpenSvcLst[] = {{{"M3UA","M3UAMGR"}, {0}, {0}, {0}}};

M3UAOpenSvcLst[0].svcargs.args[0] = Board;     /* board number          */
M3UAOpenSvcLst[0].svcargs.args[1] = Sio;       /* Service Information
                                                * Octet                */
M3UAOpenSvcLst[0].svcargs.args[2] = NSapNmb;   /* network SAP number    */
M3UAOpenSvcLst[0].svcargs.args[3] = MyEnt;     /* application entity ID */
M3UAOpenSvcLst[0].svcargs.args[4] = DFLT_INST; /* Inst ID               */
M3UAOpenSvcLst[0].svcargs.args[5] = SuId;      /* Service user ID       */
M3UAOpenSvcLst[0].svcargs.args[6] = poolsize;  /* Pool size             */
```

ctaOpenServices is an asynchronous function. The return from the function indicates that the bind operation was initiated. Once completed, a CTAEVN_OPEN_SERVICES_DONE event is returned to the application.

If multiple contexts are assigned to the same queue, all of those contexts must use the same entity ID in the service arguments parameter. Conversely, contexts bound to different queues must specify unique entity IDs.

```
CTA_EVENT   event;    /* Event structure to wait for M3UA events */
...


Ret = ctaOpenServices( ctaHd, M3UAOpenSvcLst, 1 );
if ( Ret != SUCCESS )
{
    ctaGetText( NULL_CTAHD, Ret, sErr, sizeof( sErr ) );
    printf( "ERROR : ctaOpenServices failed( %s )\n", sErr );
    ctaDestroyQueue( ctaQueue );  /* destroys context too */
    return(...)
}

/* Wait for "open services" to complete; note: this loop
 * assumes no other contexts are already active on the queue
 * we're waiting on, so no other events will be received that
 * need handling
 */
event.id = CTAEVN_NULL_EVENT;
do
{
    ctaWaitEvent( ctaQueue, &event, 5000 );
}
while( (event.id != CTAEVN_OPEN_SERVICES_DONE) &&
       (event.id != CTAEVN_WAIT_TIMEOUT) );

/* check if binding succeeded */
if( (pSap->event.id != CTAEVN_OPEN_SERVICES_DONE) ||
    (pSap->event.value != CTA_REASON_FINISHED) )
{
    ctaGetText( event.ctahd, event.value, sErr, sizeof( sErr ) );
    printf( "ERROR opening M3UA service [%s]\n", sErr );
    ctaDestroyQueue( pSap->ctaQueue );    /* destroys context too */
    return( ... );
}
```

This example is correct only if the application uses a separate queue for each context and service instance. If the application opens multiple service instances against the same queue, with either multiple SAPs on the same board or on multiple boards (in a redundant configuration), it must process events (call **M3UARetrieveMessage**) for other contexts while waiting for the CTAEVN_OPEN_SERVICES_DONE event. Failure to do so can result in an infinite loop.

Handling redundancy events

After binding to an M3UA NSAP, the application receives a M3UARUNSTATEIND event indicating the redundancy state of the M3UA layer on the board. The event type associated with this event indicates one of the following states:

Event type	Description
SN_HAST_STANDALONE	Application is in a non-redundant configuration; normal operation can begin.
SN_HAST_PRIMARY	M3UA layer is configured on the primary board in a redundant board pair; normal operation is allowed as long as the board remains the primary.
SN_HAST_BACKUP	M3UA layer is configured on the backup board in a redundant board pair, monitoring the status of the primary board; no active traffic passes through this SAP until the board becomes the primary member of the pair.

The M3UARUNSTATEIND event is the first message posted to the application queue for each SAP after the binding is confirmed. No data traffic (unitdata or connections requests) should be directed to this SAP until this event is received.

See the *Dialogic® TX Series SS7 Boards Health Management Developer's Reference Manual* for details on writing redundant M3UA applications.

5 M3UA service function reference

M3UA service function summary

The M3UA service consists of the following asynchronous functions:

Function	Description
M3uaGetApiStats	Obtains statistics from the service about congestion level activity.
M3uaRetrieveMessage	Checks for and retrieves an incoming message from the M3UA layer.
M3uaSendData	Requests data to be transmitted to a specified signaling point.

Using the M3UA service function reference

This section provides an alphabetical reference to the M3UA service functions. A typical function includes:

Prototype	The prototype is followed by a list of the function arguments. Dialogic data types include: • U8 (8-bit unsigned) • U16 (16-bit unsigned) • S16 (16-bit signed) • U32 (32-bit unsigned) • Bool (8-bit unsigned)
Return values	The return value for a function is either M3UA_SUCCESS or an error code. A return value of M3UA_SUCCESS (zero) indicates the function was initiated; subsequent events indicate the status of the operation.

M3uaGetApiStats

Obtains statistics from the host driver about congestion level and data transfer activity.

Prototype

DWORD **M3uaGetApiStats** (CTAHD ***ctahd***, M3UAAPISTATS ****pStats***, U8 ***bReset***)

Argument	Description
ctahd	Natural Access handle.

pStats	Pointer to the following M3UAApiSt structure:
	`typedef struct`
	`{`
	`  U32 qCount;  /* number of API messages currently queued */`
	`  /* to M3UA task */`
	`  U32 qPeak;  /* max number of API messages ever queued to */`
	`  /* M3UA task */`
	`  U32 txPending;  /* current number of outstanding transmit */`
	`  /* rqsts to M3UA task */`
	`  U32 txPendPeak;  /* max number of transmit rqsts ever */`
	`  /* outstanding to M3UA task */`
	`  U32 txSuccess;  /* number of successful transmit requests */`
	`  /* completed */`
	`  U32 txFailed;  /* number of failed transmit requests */`
	`  U32 txLastErr;  /* error code from last failed transmit */`
	`  /* request */`
	`  U32 rxSuccess;  /* number of events received from M3UA task */`
	`  U32 rxFailed;  /* number of receive failure events from */`
	`  /* M3UA task */`
	`  U8  apiQCongLvl; /* current outbound queue congestion level */`
	`  /* [0..3] */`
	`  U8  spare1;  /* spare for alignment */`
	`  U16 spare2;  /* spare for alignment */`
	`} M3UAAPISTATS;`

bReset	If non-zero, statistics are reset after returning them to the application.

Return values

Return value	Description
M3UA_SUCCESS	
CTAERR_INVALID_CTAHD	Invalid handle.

M3uaRetrieveMessage

Checks for and retrieves an incoming message from the M3UA layer.

Prototype

DWORD NMSAPI **M3uaRetrieveMessage** (CTAHD ***ctahd,*** void ****pMsgInd***, short ****Length***)

Argument	Description
ctahd	Natural Access handle.
pMsgInd	Pointer to the buffer where the received message is returned. The size of the buffer must be large enough to contain a data indication. See the Details section for more information about data indications.
Length	Pointer to the buffer where the length of the incoming message is returned.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_INVBOARD	Invalid board number.
M3UA_NOT_BOUND	**ctaOpenServices** not previously called.
M3UA_NO_MSG	No message currently waiting.

Details

To process incoming messages, the application can perform one of the following tasks:

- Call **M3UARetrieveMessage** periodically within a polling process. The application must poll regularly to avoid excessive queuing of messages in the TX driver or the M3UA layer.
- Wait for an asynchronous notification that a message is available, and then call **M3UARetrieveMessage** to retrieve it.

Applications can receive the following types of messages:

- Data indications
- Status indications

Data indications

An application receives a DATA_IND structure when a remote signaling point sends data matching the application's service information octet:

```
typedef struct data_ind_s
{
  U8   code;           /* M3UA_DATA_IND (0x1A)              */
  U8   spare1;         /* Alignment                         */
  U16  sapId;          /* Service user Id from Open Services*/
  U32  opc;            /* Originating Point Code            */
  U32  dpc;            /* Destination Point Code            */
  U8   srvInfo;        /* Service information octet         */
  U8   lnkSel;         /* Link selector field               */
  U16  spare2;         /* Alignment                         */
  U8   data[MAXDATA]   /* Received data packet              */
} DATA_IND;
```

DATA_IND contains the following fields:

Field	Description
code	Always set to M3UA_DATA_IND, for data indications.
sapID	Service user identifier passed as suId in **ctaOpenServices**.
opc	Point code of the remote node that sent this data. Valid values are 0 through 0xFFFFFF.
dpc	Destination point code from the routing label of the incoming message. Valid values are 0 through 0xFFFFFF.
srvInfo	Service information octet from the incoming message specifying the upper task protocol and the network indicator.
lnkSel	Link selector field from the incoming message. Valid values are: 0 through 15 (ITU-T) 0 through 31 (ANSI) 0 through 255 (TUP or 8-bit ANSI)
data [MAXDATA]	Array of size MAXDATA where received data is stored.

Status indications

An application receives a STAT_IND structure when an important status change occurs. Status changes may be generated by the local M3UA layer or received from a remote MTP3 layer via a signaling gateway (SG). For more information, see *Status indications* on page 28.

```
typedef struct stat_ind_s
{
  U8   code;      /* M3UA_STAT_IND (0x7A)                */
  U8   spare1;    /* Alignment                           */
  U16  sapId;     /* Service user Id from Open Services  */
  U32  pc;        /* Point Code related to the status ind */
  U32  opc;       /* Originating point code related to the status ind */
  S16  status;    /* Status indicator. See defines below*/
  U8   priority;  /* Priority of this status indication  */
  U16  spare2;    /* Alignment                           */
} STAT_IND;
```

STAT_IND contains the following fields:

Field	Description
code	Whether an indication is data or status. Always M3UA_STAT_IND for status indications.
sapId	Service user identifier passed as suId in the **ctaOpenServices** call.
pc	Destination point code affected by this status indication.
opc	Originating point code affected by this status indication.
status	Status event that occurred: StatPaused = Data traffic paused StatResumed = Data traffic resumed StatCongested = Data congested StatUsrUnavail = User unavailable StatCongestionEnds = Congestion ended StatPrimary = M3UA is primary in a redundant configuration StatBackup = M3UA is backup in a redundant configuration StatStandAlone = M3UA is not in a redundant configuration
priority	Current congestion level for congestion-related indications. Valid range is 0 (lowest) through 3 (highest).

M3uaSendData

Requests data to be transmitted to a specified signaling point.

Prototype

DWORD NMSAPI **M3uaSendData** (CTAHD ***ctahd***, S16 ***sapId***, U32 ***opc***, U32 ***dpc***, U8 ***lnkSel***, U8 ***priority***, U8 ****data***, S16 ***length***, U8 ***srvInfo***)

Argument	Description
ctahd	Natural Access handle.
sapId	Service access point ID. Specify the same value used for **ctaOpenServices**.
opc	24-bit, 14-bit, or 16-bit originating point code to be inserted in the outgoing message.
dpc	24-bit, 14-bit, or 16-bit destination point code of the remote system.
lnkSel	Link selector passed to an SGP to choose the link to send data over. Value is masked based on the configured slsLen. Masking results in ranges from 0 through 15 for slsLen=4, 0 through 31 for slsLen=5, or 0 through 255 for slsLen=8.
priority	Priority of the message. Valid range is 0 (lowest) through 3 (highest).
data	Pointer to the address of a buffer of data to transmit.
length	Length (in octets) of the data in the ***data*** field.
srvInfo	Service information octet (SIO) associated with this message.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_INVBOARD	Invalid board number.
M3UA_NOT_BOUND	**ctaOpenServices** not previously called.
M3UA_OSERROR	Lower-level drivers or task returned an error.
M3UA_RESOURCE	Host running out of buffers to send to the board. See the Details section for more information.

Details

Both ***opc*** and ***dpc*** are passed as 32-bit values. For example, the 24-bit point code 5.49.7 is passed as 0x053107.

lnkSel is an 8-bit value that is masked by M3UA based on the configured slsLen. The lower four bits of lnkSel are used as the sls value if slsLen is configured as 4, the lower five bits of lnkSel are used as the sls value if slsLen is configured as 5, and no masking is performed and the entire 8-bit lnkSel is used as the sls value if slsLen is configured as 8. For more information, see *M3UANwkCfg* on page 115.

The user data consists of the upper layer data. For example, when constructing an ISUP message, the first byte of user data is the first byte of the circuit identification code (CIC). The user application is responsible for any byte-order translation necessary for all data in the data field.

The SIO must be unique for each application. The service information field is composed of service indicator and network indicator fields.

If you receive an M3UA_RESOURCE error, perform one or more of the following tasks:

- Increase the number of host buffers in **ctaOpenServices**. See *Using ctaOpenServices* on page 38.
- Monitor M3UAEVN_CONGEST indications to determine when the number of available buffers is running low. Traffic could then be throttled before the buffer pool is exhausted. See *Controlling congestion* on page 29.
- Reduce other host to board traffic, including management function traffic.

For more information, see *Transferring data* on page 27.

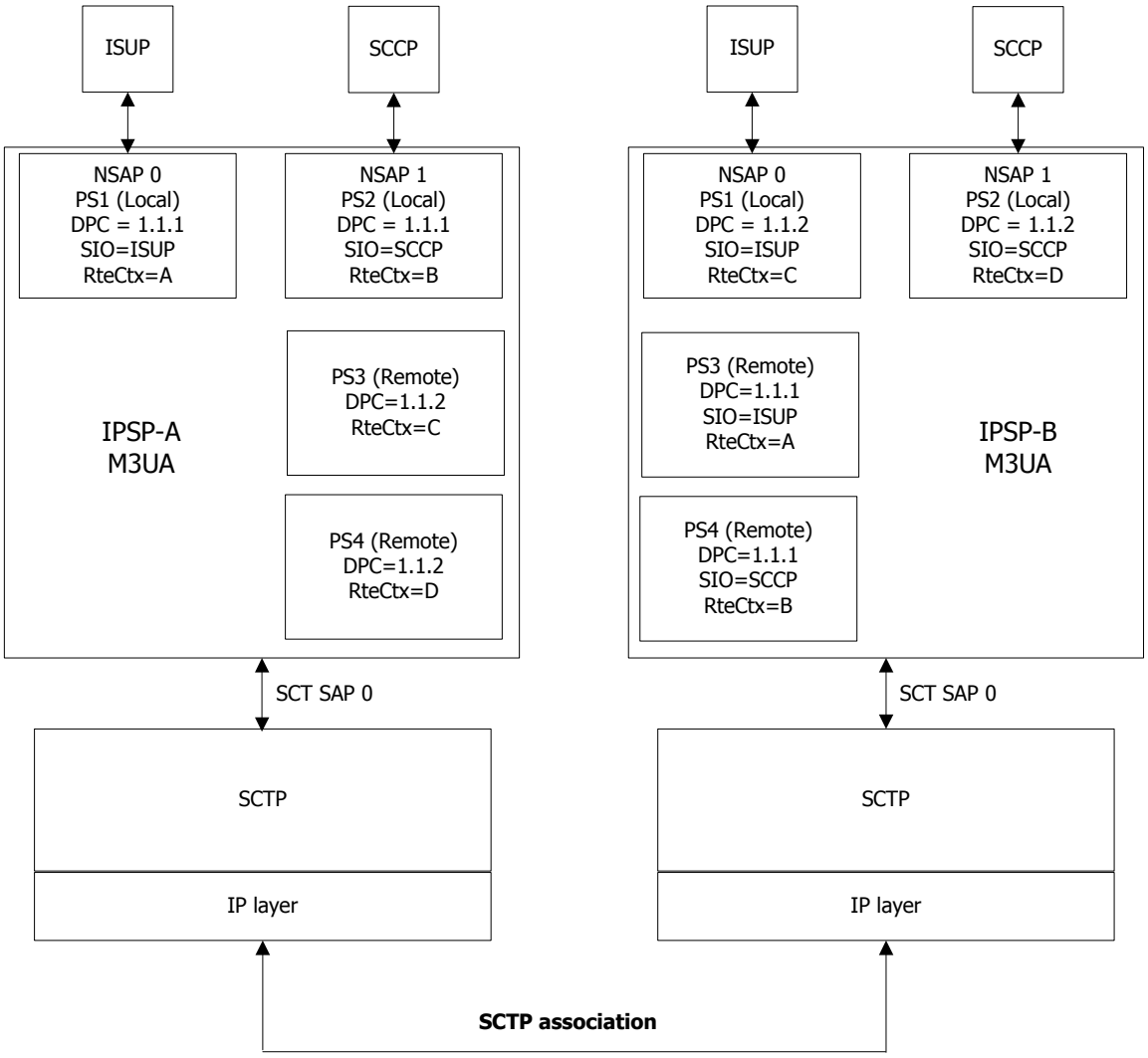
6 Managing the M3UA layer

Configuring M3UA entities

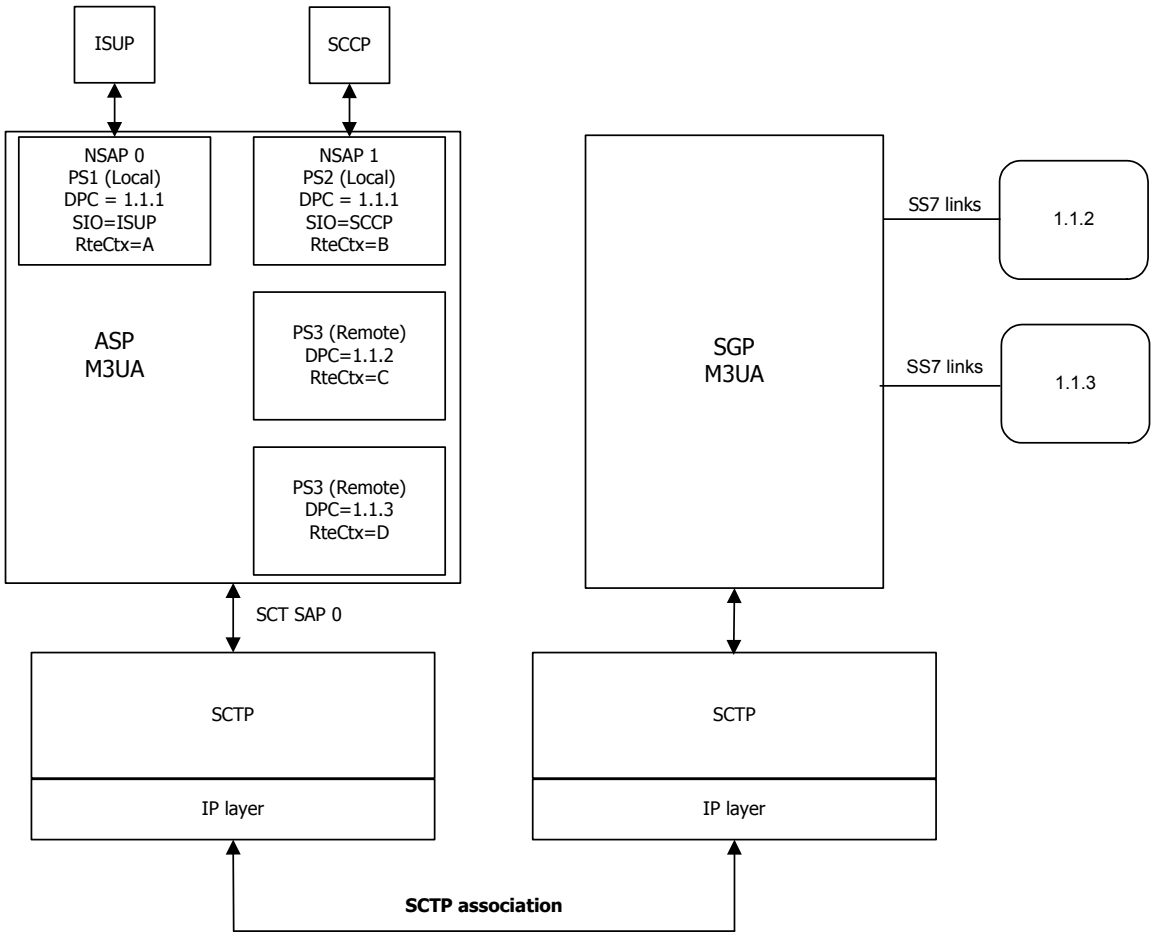
Configure M3UA entities after calling **M3UAMgmtInit**. You must configure M3UA entities in the following order:

- General M3UA configuration
- Networks
- SCT SAPs
- NSAPs
- Peer signaling processes
- Peer servers
- Routes

The following illustration shows the local and remote configuration for a SIGTRAN architecture consisting of two IP server processes (IPSPs):



The following configuration shows the local and remote configuration for a SIGTRAN architecture consisting of an application server process (ASP) and a signaling gateway process (SGP):



For information about the initial configuration of M3UA, see the *Dialogic® NaturalAccess™ Signaling Software Configuration Manual*.

General M3UA configuration

General configuration parameters define and control the general operation of the signaling point (SP) implemented by the SS7 software. The configurable attributes for the general M3UA configuration include the:

- Maximum number of other configurable elements (such as NSAPs) to control memory allocation
- Congestion level attributes
- Timer values

Define the general parameters for M3UA once at board download time, before you configure the other M3UA entities. To define the general parameters, follow these steps:

Step	Action
1	Call **M3uaInitGenCfg** to initialize the general configuration parameter structure (M3UAGenCfg) to default values.
2	Change the parameter values, as appropriate.
3	Call **M3uaSetCfg** to set the configuration.

After the general configuration is defined, you can optionally change the values for the congestion level and timer parameters. To change these parameter values, follow these steps:

Step	Action
1	Call **M3uaGetGenCfg** to obtain the current general configuration values.
2	Change the parameter values, as appropriate.
3	Call **M3uaSetGenCfg** to set the configuration with the specified values.

You must download the board again to change any of the other general configuration parameters. For more information, see *M3uaInitGenCfg* on page 71, *M3uaSetGenCfg* on page 93, and *M3UAGenCfg* on page 108.

M3UA network configuration

Network configuration parameters define the types of networks in which the M3UA layer is used. The configurable attributes of a network include the:

- Network identifier
- Network appearance
- Subservice field (SSF)
- DPC length
- SLS length
- Service user variants

Define network types after you define the general M3UA configuration parameters. In most applications, a single network configuration is sufficient. However, if you use multiple network types (such as ANSI and ITU) in the same node, define one network configuration for each network type. You can define multiple network configurations up to the maximum number of networks (MAX_NETWORK) allowed by the general configuration parameters definition.

To define a network configuration, follow these steps:

Step	Action
1	Call **M3uaInitNwkCfg** to initialize the network parameter structure (M3UANwkCfg) to default values.
2	Change the default values, as appropriate.
3	Call **M3uaSetNwkCfg** to set the configuration with the specified values.

You must download the board again to change the network configuration parameters. For more information, see *M3uaInitNwkCfg* on page 73, *M3uaSetNwkCfg* on page 95,and *M3UANwkCfg* on page 115.

M3UA SCT SAP configuration

The M3UA SCT service access point (SCT SAP) configuration parameters define the interface between the M3UA and SCTP layers. The configurable attributes in an SCT SAP include the:

- M3UA identifier for the SCT SAP
- Source port
- SCTP identifier for the SCT SAP

Define the SCT SAP for the M3UA layer right after you define the M3UA network configuration parameters. You can define only one SCT SAP, whose identifier must be 0.

To define an SCT SAP configuration, follow these steps:

Step	Action
1	Call **M3uaInitSctSapCfg** to initialize the SCT SAP parameter structure (M3UASCTSapCfg) to default values.
2	Change the default values, as appropriate.
3	Call **M3uaSetSctSapCfg** to set the configuration with the specified values.

You must download the board again to change the SCT SAP configuration parameters. For more information, see *M3uaInitSctSapCfg* on page 77, *M3uaSetSctSapCfg* on page 99, and *M3UASctSapCfg* on page 129.

M3UA NSAP configuration

M3UA network service access point (NSAP) configuration parameters define the SS7 applications, such as ISUP, SCCP, and TCAP, that use M3UA. The configurable attributes of NSAPs include the:

- NSAP identifier
- Logical network identifier
- Type of NSAP service user

Define NSAPs for the M3UA layer after you define the SCT SAP configuration parameters. You can define up to the maximum number of NSAPs (MAX_NSAP) allowed by the general configuration parameters definition.

To define an NSAP configuration, follow these steps:

Step	Action
1	Call **M3uaInitNSapCfg** to initialize the NSAP parameter structure (M3UANSapCfg) to default values.
2	Change the default values, as appropriate.
3	Call **M3uaSetNSapCfg** to set the configuration with the specified values.

You must download the board again to change the network configuration parameters. For more information, see *M3uaInitGenCfg* on page 71, *M3uaSetNSapCfg* on page 94,and *M3UANSapCfg* on page 112.

M3UA peer signaling process configuration

M3UA peer signaling process configuration parameters define a peer signaling process for the M3UA layer. The configurable attributes in a peer signaling process include the:

- Peer signaling process identifier
- Remote peer signaling process type (signaling gateway process or IP signaling process)
- Optional message field requirements
- Remote association attributes

Define peer signaling processes for the M3UA layer after you define the M3UA NSAP configuration parameters. You can define up to the maximum number of peer signaling processes (MAX_PSP) allowed by the general configuration parameters definition.

To define a peer signaling process configuration, follow these steps:

Step	Action
1	Call **M3uaInitPspCfg** to initialize the peer signaling process parameter structure (M3UAPSPCfg) to default values.
2	Change the default values, as appropriate.
3	Call **M3uaSetPspCfg** to set the configuration with the specified values.

After a peer signaling process configuration is defined, you can optionally change the values for some of the peer signaling process parameters, as described in *M3UAPspCfg* on page 121. To change these parameter values, follow these steps:

Step	Action
1	Call **M3uaGetPspCfg** to obtain the current peer signaling process configuration values.
2	Change the parameter values, as appropriate.
3	Call **M3uaSetPspCfg** to set the configuration with the specified values.

You must download the board again to change any of the other peer signaling process configuration parameters. For more information, see *M3uaInitPspCfg* on page 75, *M3uaSetPspCfg* on page 97, and *M3UAPspCfg* on page 121.

M3UA peer server configuration

The M3UA peer server configuration parameters define a peer server for the M3UA layer. The configurable attributes in a peer server include:

- Peer server identifier
- Peer server network identifier
- Whether the peer server is local or remote
- Routing context
- List of peer signaling processes defined for this PS

Define peer servers for the M3UA layer after you define peer server processes. You can define up to the maximum number of peer servers (MAX_PS) allowed by the general configuration parameters definition.

To define a peer server configuration, follow these steps:

Step	Action
1	Call **M3uaInitPsCfg** to initialize the peer server parameter structure (M3UAPSCfg) to default values.
2	Change the default values, as appropriate.
3	Call **M3uaSetPsCfg** to set the configuration with the specified values.

After a peer server configuration is defined, you can optionally change the values for some parameters, as described in **M3UAPsCfg**. To change these parameter values, follow these steps:

Step	Action
1	Call **M3uaGetPsCfg** to obtain the current PS configuration values.
2	Change the parameter values as appropriate.
3	Call **M3uaSetPsCfg** to set the configuration with the specified values.

You must download the board again to change the other peer server configuration parameters. For more information, see *M3uaInitPsCfg* on page 74, *M3uaSetPsCfg* on page 96, and *M3UAPsCfg* on page 117.

M3UA route configuration

The route configuration parameters define a route for the M3UA layer. The configurable attributes in a route include the:

- Route identifier
- Route type (local user or peer server)
- NSAP identifier associated with the route, for local (inbound) routes
- Route filter configuration

Define routes for the M3UA layer after you define the peer servers.

To define a route configuration, follow these steps:

Step	Action
1	Call **M3uaInitRteCfg** to initialize the route parameter structure (M3UARteCfg) to default values.
2	Change the default values, as appropriate.
3	Call **M3uaSetRteCfg** to set the configuration with the specified values.

You must download the board again to change the route configuration parameters. For more information, see *M3uaInitRteCfg* on page 76, *M3uaSetRteCfg* on page 98, *M3UARteCfg* on page 127.

Controlling M3UA entities

Use **M3uaMgmtCtrl** to enable the application to control M3UA entities. The following table describes the available control requests by entity:

Entity	Control request
None specified	One of the following general control requests: • Enable or disable alarms. • Start or stop debug logging. • Start flow control and disable the transmission of all M3UA messages except critical messages. • End flow control and enable the transmission of all M3UA messages. • Shut down the M3UA layer. • Start or stop trace data
ASP	Send any of the following messages: • ASP Up • ASP Down • ASP Active • ASP Inactive **Note:** Use these messages for debugging purposes only. They are sent automatically by the M3UA layer and should not usually be sent through the management API.
Association	Any of the following: • Inhibit or uninhibit the association. • Establish or terminate the association. • Dynamically change the Type Of Service octet for all IP messages across this association. **Note:** Establish or terminate an association for debugging purposes only. This is performed automatically by the M3UA layer and should not usually be performed through the management API.
Routing key	Register or de-register a dynamic routing key. **Note:** Dynamic routing keys are not currently supported.
SCT SAP	Enable or disable the SCT SAP. **Note:** Enable or disable the SCT SAP for debugging purposes only. These actions are performed automatically by the M3UA layer and should not usually be performed through the management interface.

Retrieving M3UA statistics

Use the following M3UA statistics functions to retrieve and optionally reset the following statistics:

Entity	Function	Statistics returned include...
None specified	**M3uaGenStatistics**	Transmit and receive counts for M3UA management messages, lower and upper interface messages, and error messages.
NSAP	**M3uaNsapStatistics**	M3UA transmit and receive message statistics over an NSAP.
Peer signaling process	**M3uaPspStatistics**	M3UA transmit and receive message statistics to and from the peer signaling process.
SCT SAP	**M3uaSctSapStatistics**	Transmit and receive counts for messages over the SCT SAP.

Retrieving M3UA status information

Use the following M3UA status functions to retrieve status information from the M3UA layer:

Entity	Function	Status information returned includes
Address translation	**M3uaAddrTransStatus**	Number of destination point codes and routes.
Destination point	**M3uaDpcStatus**	Destination point code state and congestion level.
General M3UA layer	**M3uaGenStatus**	Size of the memory reserved for and allocated for M3UA.
Network service access point (NSAP)	**M3uaNSapStatus**	NSAP state and remote identifier.
Peer server	**M3uaPsStatus**	Peer server's application server state and endpoint status.
Peer signaling process	**M3uaPspStatus**	Peer signaling process state and endpoint status.
Routing key	**M3uaRkStatus**	Number of dynamically registered routing keys.
SCT SAP	**M3uaSctSapStatus**	SCT SAP state and list of IP addresses used in associations over the SAP.

7 M3UA management function reference

M3UA management function summary

NaturalAccess M3UA consists of the following synchronous management functions, in which the action is completed before control is returned to the application:

- Control
- Configuration
- Statistics
- Status

Control functions

Function	Description
M3uaMgmtCtrl	Sends a control request to the M3UA layer.
M3uaMgmtInit	Initializes internal structures and opens communication with the M3UA process on the TX board. This function must be called before any other management function.
M3uaMgmtTerm	Terminates access to the M3UA management API for this application.

Configuration functions

Function	Description
M3uaGetGenCfg	Obtains general configuration parameter values from the M3UA layer.
M3uaGetNwkCfg	Obtains network configuration parameter values from the M3UA layer.
M3uaGetNSapCfg	Obtains NSAP configuration parameter values from the M3UA layer.
M3uaGetPsCfg	Obtains peer server configuration parameter values from the M3UA layer.
M3uaGetPspCfg	Obtains peer signaling process configuration parameter values from the M3UA layer.
M3uaGetRteCfg	Obtains route configuration parameter values from the M3UA layer.
M3uaGetSctSapCfg	Obtains SCT SAP configuration parameter values from the M3UA layer.
M3uaInitGenCfg	Initializes the provided general configuration structure with default values.
M3uaInitNSapCfg	Initializes the provided NSAP configuration structure with default values and the NSAP identifier.
M3uaInitNwkCfg	Initializes the provided network configuration structure with default values and the network identifier.
M3uaInitPsCfg	Initializes the provided peer server configuration structure with default values and the peer server identifier.
M3uaInitPspCfg	Initializes the provided peer signaling process configuration structure with default values and the peer signaling process identifier.
M3uaInitRteCfg	Initializes the provided route configuration structure with default values and the route number, DPC, and switch type.
M3uaInitSctSapCfg	Initializes the provided SCT SAP configuration structure with default values and the SCT SAP identifier.
M3uaSetGenCfg	Configures the M3UA layer with the values contained in the general configuration structure.
M3uaSetNSapCfg	Configures the M3UA layer with the values contained in the NSAP configuration structure.
M3uaSetNwkCfg	Configures the M3UA layer with the values contained in the network configuration structure.
M3uaSetPsCfg	Configures the M3UA layer with the values contained in the peer server configuration structure.
M3uaSetPspCfg	Configures the M3UA layer with the values contained in the peer signaling process structure.
M3uaSetRteCfg	Configures the M3UA layer with the values contained in the route configuration structure.
M3uaSetSctSapCfg	Configures the M3UA layer with the values contained in the SCTP SAP configuration structure.

Statistics functions

Function	Description
M3uaGenStatistics	Obtains and optionally resets the general statistics for the M3UA layer.
M3uaNSapStatistics	Obtains and optionally resets the statistics for the specified M3UA NSAP.
M3uaPspStatistics	Obtains and optionally resets the statistics for the specified M3UA peer signaling process.
M3uaSctSapStatistics	Obtains and optionally resets the statistics for the specified M3UA SCT SAP.

Status functions

Function	Description
M3uaAddrTransStatus	Obtains address translation status information from the M3UA layer.
M3uaDpcStatus	Obtains destination point code status information from the M3UA layer.
M3uaGenStatus	Obtains general status information from the M3UA layer.
M3uaNSapStatus	Obtains NSAP status information from the M3UA layer.
M3uaPspRkIdStatus	Obtains information about the routing keys associated with the specified peer signaling process.
M3uaPspStatus	Obtains peer signaling process status information from the M3UA layer.
M3uaPsStatus	Obtains peer server status information from the M3UA layer.
M3uaRkStatus	Obtains routing key status information from the M3UA layer.
M3uaRteStatus	Obtains route status information from the M3UA layer.
M3uaSctSapStatus	Obtains SCT SAP status information from the M3UA layer.

Using the M3UA management function reference

This section provides an alphabetical reference to the M3UA management functions. A typical function includes:

Prototype	The prototype is followed by a list of the function arguments. Dialogic data types include: • U8 (8-bit unsigned) • S16 (16-bit signed) • U16 (16-bit unsigned) • U32 (32-bit unsigned) • Bool (8-bit unsigned)
Return values	The return value for a function is either M3UA_SUCCESS or an error code.

M3uaAddrTransStatus

Obtains address translation status information from the M3UA layer, including the number of DPCs and the number of routes.

Prototype

STATUS TXM3UAFUNC **M3uaAddrTransStatus** (U8 ***board***, M3UAAtSta ****pStatus***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
pStatus	Pointer to the M3UAAtSta structure where the requested status information is returned. For information, see *M3UAAtSta* on page 106.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

M3uaDpcStatus

Obtains DPC status information from the M3UA layer, including DPC state and congestion level.

Prototype

STATUS TXM3UAFUNC **M3uaDpcStatus** (U8 ***board***, U32 ***pc***, U8 ***nwkId***, M3UADpcSta ****pStatus***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
pc	Destination point code for the status request.
nwkId	Network identifier.
pStatus	Pointer to the M3UADpcSta structure where the requested status information is returned. For information, see *M3UADpcSta* on page 107.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

M3uaGenStatistics

Obtains and optionally resets the general statistics for the M3UA layer.

Prototype

STATUS TXM3UAFUNC **M3uaGenStatistics** (U8 ***board***, M3UAGenSts ****pSts***, Bool ***isReset***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
pSts	Pointer to the M3UAGenSts structure where the general statistics information is returned. For information, see *M3UAGenSts* on page 111.
isReset	If non-zero, statistics are set to zero after retrieval.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

M3uaGenStatus

Obtains general status information from the M3UA layer, including the size of reserved and allocated memory.

Prototype

STATUS TXM3UAFUNC **M3uaGenStatus** (U8 ***board***, M3UAGenSta ****pStatus***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
pStatus	Pointer to the M3UAGenSta structure where the requested status information is returned. For information, see *M3UAGenSta* on page 110.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

M3uaGetGenCfg

Obtains general configuration parameter values from the M3UA layer.

Prototype

STATUS TXM3UAFUNC **M3uaGetGenCfg** (U8 ***board***, M3UAGenCfg ****pGenCfg***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
pGenCfg	Pointer to the M3UAGenCfg structure where the requested configuration information is returned. For information, see *M3UAGenCfg* on page 108.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

Details

This function can be called any time after **M3uaMgmtInit**. An application must provide a pointer to a buffer large enough to hold the M3UAGenCfg structure.

See also

M3uaInitGenCfg, **M3uaSetGenCfg**

M3uaGetNSapCfg

Obtains NSAP configuration parameter values from the M3UA layer.

Prototype

STATUS TXM3UAFUNC **M3uaGetNSapCfg** (U8 ***board***, M3UANSapCfg ****pNSapCfg***, S16 ***nSapNo***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
pNSapCfg	Pointer to the M3UANSapCfg structure where the requested configuration information is returned. For information, see *M3UANSapCfg* on page 112.
nSapNo	Network SAP number from which to retrieve the configuration information. Valid range is 0 through (MAX_NSAP - 1).

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

Details

This function can be called any time after **M3uaMgmtInit**. An application must provide a pointer to a buffer large enough to hold the M3UANSapCfg structure.

See also

M3uaInitNSapCfg, **M3uaSetNSapCfg**

M3uaGetNwkCfg

Obtains network configuration parameter values from the M3UA layer.

Prototype

M3uaGetNwkCfg (U8 ***board***, M3UANwkCfg ****pNwkCfg***, U8 ***nwkId***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
pNwkCfg	Pointer to the M3UANwkCfg structure where the requested configuration information is returned. For information, see *M3UANwkCfg* on page 115.
nwkId	Network identifier.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

Details

This function can be called any time after **M3uaMgmtInit**. An application must provide a pointer to a buffer large enough to hold the M3UANwkCfg structure.

See also

M3uaInitNwkCfg, **M3uaSetNwkCfg**

M3uaGetPsCfg

Obtains peer server configuration parameter values from the M3UA layer.

Prototype

STATUS TXM3UAFUNC **M3uaGetPsCfg** (U8 ***board***, M3UAPsCfg ****cfg***, U32 ***psId***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
cfg	Pointer to the M3UAPsCfg structure where the requested configuration information is returned. For information, see *M3UAPsCfg* on page 117.
psId	Identifier of the peer server for which to obtain configuration information.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

Details

This function can be called any time after **M3uaMgmtInit**. An application must provide a pointer to a buffer large enough to hold the M3UAPsCfg structure.

See also

M3uaInitPsCfg, **M3uaSetPsCfg**

M3uaGetPspCfg

Obtains peer signaling process configuration parameter values from the M3UA layer.

Prototype

STATUS TXM3UAFUNC **M3uaGetPspCfg** (U8 ***board***, U16 ***pspId***, M3UAPspCfg *****pPspCfg***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
pspId	Identifier of the peer server for which to obtain configuration information.
pPspCfg	Pointer to the M3UAPspCfg structure where the requested configuration information is returned. For information, see *M3UAPspCfg* on page 121.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

Details

This function can be called any time after **M3uaMgmtInit**. An application must provide a pointer to a buffer large enough to hold the M3UAPspCfg structure.

See also

M3uaInitPspCfg, **M3uaSetPspCfg**

M3uaGetRteCfg

Obtains route configuration parameter values from the M3UA layer.

Prototype

STATUS TXM3UAFUNC **M3uaGetRteCfg** (U8 ***board***, M3UARteCfg ****pRouteCfg***, U32 ***dpc***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
pRouteCfg	Pointer to the M3UARteCfg structure where the requested configuration information is returned. For information, see *M3UARteCfg* on page 127.
dpc	Destination point code associated with the route for which configuration information is obtained.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

Details

This function can be called any time after **M3uaMgmtInit**. An application must provide a pointer to a buffer large enough to hold the M3UARteCfg structure.

See also

M3uaInitRteCfg, M3uaSetRteCfg

M3uaGetSctSapCfg

Obtains SCT SAP configuration parameter values from the M3UA layer.

Prototype

STATUS TXM3UAFUNC **M3uaGetSctSapCfg** (U8 ***board***, M3UASctSapCfg *****pSctSapCfg***, S16 ***suId***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
pSctSapCfg	Pointer to the M3UASctSapCfg structure where the requested configuration information is returned. For information, see *M3UASctSapCfg* on page 129.
suId	M3UA identifier of the SCT SAP for which to obtain configuration information.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

Details

This function can be called any time after **M3uaMgmtInit**. An application must provide a pointer to a buffer large enough to hold the M3UASctSapCfg structure.

See also

M3uaInitSctSapCfg, **M3uaSetSctSapCfg**

M3uaInitGenCfg

Initializes the provided M3UA general configuration structure with default values.

Prototype

STATUS TXM3UAFUNC **M3uaInitGenCfg** (M3UAGenCfg ***pGencfg***)

Argument	Description
pGenCfg	Pointer to the M3UAGenCfg structure to be initialized. For information, see *M3UAGenCfg* on page 108.

Return value

Return value	Description
M3UA_SUCCESS	

Details

After calling **M3uaInitGenCfg**, call **M3uaSetGenCfg** to set the M3UA general configuration values. You can optionally override specific field values before calling **M3uaSetGenCfg**.

For more information about configuring general parameters for the M3UA layer, see *General M3UA configuration* on page 49.

See also

M3uaGetGenCfg

M3uaInitNSapCfg

Initializes the provided M3UA NSAP configuration structure with default values and the NSAP identifier.

Prototype

STATUS TXM3UAFUNC **M3uaInitNSapCfg** (M3UANSapCfg ***pNSapCfg***, S16 ***nsapId***)

Argument	Description
pNSapCfg	Pointer to the M3UANSapCfg structure that contains the NSAP configuration values. For information, see *M3UANSapCfg* on page 112.
nsapId	Identifier of the NSAP to initialize.

Return value

Return value	Description
M3UA_SUCCESS	

Details

After calling **M3uaInitNSapCfg**, call **M3uaSetNSapCfg** to set the NSAP configuration. You can optionally override specific field values before calling **M3uaSetNSapCfg**.

For more information about defining an NSAP to the M3UA layer, see *M3UA NSAP configuration* on page 51.

See also

M3uaGetNSapCfg

M3uaInitNwkCfg

Initializes the provided M3UA network configuration structure with default values and the network identifier.

Prototype

STATUS TXM3UAFUNC **M3uaInitNwkCfg** (M3UANwkCfg ***pNwkCfg***, U8 ***nwkId***)

Argument	Description
pNwkCfg	Pointer to the M3UANwkCfg structure to be initialized. For information, see *M3UANwkCfg* on page 115.
nwkId	Identifier of the network to initialize.

Return value

Return value	Description
M3UA_SUCCESS	

Details

After calling **M3uaInitNwkCfg**, call **M3uaSetNwkCfg** to set the network configuration. You can optionally override specific field values before calling **M3uaSetNwkCfg**.

For more information about defining a network to the M3UA layer, see *M3UA network configuration* on page 50.

See Also

M3uaGetNwkCfg

M3uaInitPsCfg

Initializes the provided M3UA peer server configuration structure with default values and the peer server identifier.

Prototype

STATUS TXM3UAFUNC **M3uaInitPsCfg** (M3UAPsCfg ***pPsCfg***, U32 ***psId***)

Argument	Description
pPsCfg	Pointer to the M3UAPsCfg structure to be initialized. For information, see *M3UAPsCfg* on page 117.
psId	Identifier of the peer server to initialize.

Return value

Return value	Description
M3UA_SUCCESS	

Details

After calling **M3uaInitPsCfg**, call **M3uaSetPsCfg** to set the peer server configuration. You can optionally override specific field values before calling **M3uaSetPsCfg**.

For more information about defining a peer server to the M3UA layer, see *M3UA peer server configuration* on page 52.

See also

M3uaGetPsCfg

M3uaInitPspCfg

Initializes the provided M3UA peer signaling process configuration structure with default values, the peer signaling process identifier, and the IP address.

Prototype

STATUS TXM3UAFUNC **M3uaInitPspCfg** (M3UAPspCfg ***pPspCfg***, U16 ***pspId***, U32 ***ipAddr***)

Argument	Description
pPspCfg	Pointer to the M3UAPspCfg structure to be initialized. For information, see *M3UAPspCfg* on page 121.
pspId	Identifier of the peer signaling process to initialize.
ipAddr	IP address of the peer signaling process to initialize.

Return value

Return value	Description
M3UA_SUCCESS	

Details

After calling **M3uaInitPspCfg**, call **M3uaSetPspCfg** to set the peer server configuration. You can optionally override specific field values before calling **M3uaSetPspCfg**.

For more information about defining a peer signaling process to the M3UA layer, see *M3UA peer signaling process configuration* on page 52.

See also

M3uaGetPspCfg

M3uaInitRteCfg

Initializes the provided M3UA route configuration structure with default values and the associated DPC.

Prototype

STATUS TXM3UAFUNC **M3uaInitRteCfg** (M3UARteCfg ****rte***, U32 ***dpc***)

Argument	Description
rte	Pointer to the M3UARteCfg structure to be initialized. For information, see *M3UARteCfg* on page 127.
dpc	Destination point code of the route to initialize.

Return value

Return value	Description
M3UA_SUCCESS	

Details

After calling **M3uaInitRteCfg**, call **M3uaSetRteCfg** to set the route configuration. You can optionally override specific field values before calling **M3uaSetRteCfg**.

For more information about defining a route to the M3UA layer, see *M3UA route configuration* on page 53.

See also

M3uaGetRteCfg

M3uaInitSctSapCfg

Initializes the provided M3UA SCT SAP configuration structure with default values and the SCT SAP identifier.

Prototype

STATUS TXM3UAFUNC **M3UAInitSctSapCfg** (M3UASctSapCfg ***pSctSapCfg***, S16 ***sapId***)

Argument	Description
pSctSapCfg	Pointer to the M3UASctSapCfg structure that contains default values. For information, see *M3UASctSapCfg* on page 129.
sapId	M3UA identifier of the SCT SAP to initialize.

Return value

Return value	Description
M3UA_SUCCESS	

Details

After calling **M3uaInitSctSapCfg**, call **M3uaSetSctSapCfg** to set the SCTP SAP configuration. You can optionally override specific field values before calling **M3uaSetSctSapCfg**.

For more information about defining an SCT SAP to the M3UA layer, see *M3UA SCT SAP configuration* on page 51.

See Also

M3uaGetSctSapCfg

M3uaMgmtCtrl

Sends an M3UA control request to the M3UA layer.

Prototype

TXM3UAFUNC **M3uaMgmtCtrl** (U8 ***board***, S16 ***entity***, U8 ***action***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
entity	Type of control action the function performs. Valid values are: No value = General control ASP identifier = ASP control Association identifier = Association control Routing key identifier = Routing key control SCT SAP identifier - SCT SAP control **Note:** The entity identifier (***entity***) must have been previously defined with the appropriate set configuration function.
action	Action to take on the specified entity. See the Details section for valid actions.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
SIGTRAN_ERR_BADACT	Invalid ***action*** parameter.

Details

Use **M3uaMgmtCtrl** to activate and deactivate network task resources. The combination of ***entity*** and ***action*** tells M3UA what entity to act upon and what action to take. There are five types of control actions:

- General
- ASP
- Association
- Routing key
- SCT SAP

In the following value-description tables, do not use the management API to perform the actions marked with an asterisk, unless you are an experienced user. These actions are performed automatically by the M3UA layer, and are provided through the management API for testing and debugging purposes only.

General control actions

If the ***entity*** argument is not used, the ***action*** argument has the following valid values:

Value	Description
*M3UA_CTRL_ALARM_DIS	Disables alarms.
*M3UA_CTRL_ALARM_ENA	Enables alarms.
M3UA_CTRL_DEBUG_OFF	Stops debug logging.
M3UA_CTRL_DEBUG_ON	Starts debug logging.
*M3UA_CTRL_FLOWCTL_OFF	Ends flow control, and enable transmission of all M3UA management messages.
*M3UA_CTRL_FLOWCTL_ON	Starts flow control, and disable transmission of all except critical M3UA management messages.
*M3UA_CTRL_SHUTDOWN	Shuts down the M3UA layer.
M3UA_CTRL_TRACE_OFF	Stops trace data.
M3UA_CTRL_TRACE_ON	Starts trace data.

ASP control actions

If the ***entity*** argument specifies an ASP identifier, the ***action*** argument has the following valid values:

Value	Description
*M3UA_CTRL_ASP_AC	Sends an ASPAC (ASP Active) message.
*M3UA_CTRL_ASP_DN	Sends an ASPDN (ASP Down) message.
*M3UA_CTRL_ASP_IA	Sends an ASPIA (ASP Inactive) message.
*M3UA_CTRL_ASP_UP	Sends an ASPUP (ASP Up) message.

Association control actions

If the ***entity*** argument specifies an association identifier, the ***action*** argument has the following valid values:

Value	Description
*M3UA_CTRL_ASSOC_EST	Establishes the association.
M3UA_CTRL_ASSOC_INH	Inhibits the association.
*M3UA_CTRL_ASSOC_TERM	Terminates the association.
M3UA_CTRL_ASSOC_UNI	Uninhibits the association.

Routing key control actions

If the ***entity*** argument specifies a routing key identifier, the ***action*** argument has the following valid values:

Value	Description
*M3UA_CTRL_RK_DEREG	De-registers the dynamic routing key.
*M3UA_CTRL_RK_REG	Registers the dynamic routing key.

SCT SAP control actions

If the ***entity*** argument specifies an SCT SAP identifier, the ***action*** argument has the following valid values:

Value	Description
*M3UA_CTRL_SCTSAP_DIS	Disables the SCT SAP.
*M3UA_CTRL_SCTSAP_ENA	Enables the SCT SAP.

TOS control actions

If the ***entity*** argument specifies an ASP identifier and a new TOS value, the action argument has the following valid values:

Value	Description
M3UA_CTRL_SET_TOS	Set the Type of Service octet used on the association.

Note: The ASP identifier occupies the high byte in the entity field. The new TOS value occupies the low byte in the entity field.

See also

M3uaMgmtInit, **M3UAMgmtTerm**

M3uaMgmtInit

Initializes internal structures and opens communication with the M3UA layer on the TX board.

Prototype

STATUS TXM3UAFUNC **M3uaMgmtInit** (U8 ***board***, U8 ***srcEnt***, U8 ***srcInst***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
srcEnt	Source entity ID, which is the channel to open for communication with TX board.
srcInst	Source instance ID.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.

Details

You must call this function once before calling any other management functions. The source entity ID must be from 0x20 through 0x7F and unique for each application accessing the M3UA layer through the management API.

The source instance ID should always be 0 for host applications.

See also

M3uaMgmtCtrl, **M3uaMgmtTerm**

M3uaMgmtTerm

Terminates access to the M3UA management API for this application, making the management channel for a specified board available for other applications.

Prototype

STATUS TXM3UAFUNC **M3uaMgmtTerm** (U8 ***board***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_HANDLE	Handle closed or was never opened.

Details

Call this function to free up resources when an application terminates or finishes communication with the M3UA layer.

See also

M3uaMgmtCtrl, **M3uaMgmtInit**

M3uaNSapStatistics

Obtains and optionally resets the statistics for the specified M3UA NSAP.

Prototype

STATUS TXM3UAFUNC **M3uaNSapStatistics** (U8 ***board***, M3UANSapSts ****pSts***, Bool ***isReset***, S16 ***nsapNo***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
pSts	Pointer to the M3UANSapSts structure where the requested statistics information is returned. For information, see *M3UANSapSts* on page 114.
isReset	If non-zero, statistics are reset after returning them to the application.
nsapNo	Identifier of the NSAP for which to obtain statistics.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

M3uaNSapStatus

Obtains NSAP status information from the M3UA layer, including the high level NSAP state and the remote SAP identifier.

Prototype

STATUS TXM3UAFUNC **M3uaNSapStatus** (U8 ***board***, S16 ***nsapNo***, M3UANSapSta *****pStatus***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
nsapNo	Identifier of the NSAP for which to obtain status information.
pStatus	Pointer to the M3UANSapSta structure where the requested status information is returned. For information, see *M3UANSapSta* on page 113.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

M3uaPspRkIdStatus

Obtains the number of dynamic routing keys associated with the specified peer signaling process.

Prototype

STATUS TXM3UAFUNC **M3uaPspRkIdStatus** (U8 ***board***, M3UAPspRkIdSta *****pStatus***, U16 ***pspId***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
pStatus	Pointer to the M3uaPspRkIdSta structure where the requested status information is returned. For information, see *M3UAPspRkIdSta* on page 123.
pspId	Identifier of the peer signaling process for which to obtain routing key information.

Return values

Return value	Description
M3UA_SUCCESS	
	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

M3uaPspStatistics

Obtains and optionally resets the statistics for the specified M3UA peer signaling process.

Prototype

STATUS TXM3UAFUNC **M3uaPspStatistics** (U8 ***board***, M3UAPspSts ****pSts***, Bool ***isReset***, U16 ***pspId***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
pSts	Pointer to the M3UAPspSts structure where the requested statistics information is returned. For information, see *M3UAPspSts* on page 125.
isReset	If non-zero, statistics are reset after returning them to the application.
pspId	Identifier of the peer signaling process for which to obtain statistics.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

M3uaPspStatus

Obtains peer signaling process status information from the M3UA layer, including the state of its associations.

Prototype

STATUS TXM3UAFUNC **M3uaPspStatus** (U8 ***board***, U16 ***pspNo***, M3UAPspSta *****pStatus***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
pspNo	Identifier of the peer signaling process for which to obtain status information.
pStatus	Pointer to the M3UAPspSta structure where the requested status information is returned. For information, see *M3UAPspSta* on page 124.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

M3uaPsStatus

Obtains peer server status information from the M3UA layer, including the peer server state and the status of the associated endpoints.

Prototype

STATUS TXM3UAFUNC **M3uaPsStatus** (U8 ***board***, U32 ***id***, M3UAPsSta ****pStatus***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
id	Identifier of the peer server for which to obtain status information.
pStatus	Pointer to the M3UAPsSta structure where the requested status information is returned. For information, see *M3UAPsSta* on page 119.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

M3uaRkStatus

Obtains the number of dynamically registered routing keys defined in the M3UA layer.

Prototype

STATUS TXM3UAFUNC **M3uaRkStatus** (U8 ***board***, M3UAGenCfg ****pStatus***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
pStatus	Pointer to the M3UARkSta structure where the requested status information is returned. For information, see *M3UARkSta* on page 126.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

M3uaRteStatus

Obtains route status information from the M3UA layer.

Prototype

STATUS TXM3UAFUNC **M3uaRteStatus** (U8 ***board***, M3UARteCfg ****pStatus***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
pStatus	Pointer to the M3UARteCfg structure where the requested status information is returned. For information, see *M3UARteCfg* on page 127.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

M3uaSctSapStatistics

Obtains and optionally resets the statistics for the specified SCTP service access point.

Prototype

STATUS TXM3UAFUNC **M3uaSctSapStatistics** (U8 ***board***, M3UASctSapSts ****pSts***, Bool ***isReset***, S16 ***suId***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
pSts	Pointer to the M3UASctSapSts structure where the requested statistics information is returned. For information, see *M3UASctSapSts* on page 131.
isReset	If non-zero, statistics are reset after returning them to the application.
suId	M3UA identifier of the SCT SAP for which to obtain statistics.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

M3uaSctSapStatus

Obtains SCT SAP status information from M3UA.

Prototype

STATUS TXM3UAFUNC **M3uaSctSapStatus** (U8 ***board***, S16 ***id***, M3UASctSapSta *****pStatus***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through MAXBOARD (currently 32).
id	M3UA identifier of the SCT SAP for which to obtain status information.
pStatus	Pointer to the M3UASctSapSta structure where the requested status information is returned. For information, see *M3UASctSapSta* on page 130.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

M3uaSetGenCfg

Configures the M3UA layer with the values contained in the general configuration structure.

Prototype

STATUS TXM3UAFUNC **M3uaSetGenCfg** (U8 ***board***, M3UAGenCfg ****pGencfg***)

rgument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
pGenCfg	Pointer to the M3UAGenCfg structure that contains the general configuration values. For information, see *M3UAGenCfg* on page 108.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

Details

Call this function to configure the M3UA layer after you download the TX board and call **M3uaMgmtInit**.

An application must set the field values in the M3UAGenCfg structure before calling **M3uaSetGenCfg**. Set the values in any of the following ways:

- Call **M3uaInitGenCfg** to set the fields to default values.
- Set each field value from within the application.
- Call **M3uaInitGenCfg** and then override specific field values before passing the M3UAGenCfg structure to **M3uaSetGenCfg**.

M3uaSetGenCfg is typically called once by an application to set global values.

For more information, see *General M3UA configuration* on page 49.

See also

M3uaGetGenCfg

M3uaSetNSapCfg

Configures the M3UA layer with the values contained in the NSAP configuration structure.

Prototype

STATUS TXM3UAFUNC **M3uaSetNSapCfg** (U8 ***board***, M3UANSapCfg ****pNSapCfg***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through MAXBOARD (currently 32).
pNSapCfg	Pointer to the M3UANSapCfg structure that contains the NSAP configuration values. For information, see *M3UANSapCfg* on page 112.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

Details

Call this function after you configure an SCT SAP for the M3UA layer, but before you configure peer signaling processes.

An application must set the field values in the M3UANSapCfg structure before calling **M3uaSetNSapCfg**. Set the values in any of the following ways:

- Call **M3uaInitNSapCfg** to set the fields to default values.
- Set each field value from within the application.
- Call **M3uaInitNSapCfg** and then override specific field values before passing the M3UANSapCfg structure to **M3uaSetNsapCfg**.

M3uaSetNSapCfg is typically called once for each configured NSAP.

For more information, see *M3UA NSAP configuration* on page 51.

See also

M3uaGetNSapCfg

M3uaSetNwkCfg

Configures the M3UA layer with the values contained in the network configuration structure.

Prototype

STATUS TXM3UAFUNC **M3uaSetNwkCfg** (U8 ***board***, M3UANwkCfg ****pNwkCfg***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through MAXBOARD (currently 32).
pNwkCfg	Pointer to the M3UANwkCfg structure that contains the network configuration values. For information, see *M3UANwkCfg* on page 115.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

Details

Call this function after you set the general configuration parameters for the M3UA layer, but before you configure an SCTP SAP.

An application must set the field values in the M3UANwkCfg structure before calling **M3uaSetNwkCfg**. Set the values in any of the following ways:

- Call **M3uaInitNwkCfg** to set the fields to default values.
- Set each field value from within the application.
- Call **M3uaInitNwkCfg** and then override specific field values before passing the M3UANwkCfg structure to **M3uaSetNwkCfg**.

M3uaSetNwkCfg is typically called once for each configured network.

For more information, see *M3UA network configuration* on page 50.

See also

M3uaGetNwkCfg

M3uaSetPsCfg

Configures the M3UA layer with the values contained in the peer server configuration structure.

Prototype

STATUS TXM3UAFUNC **M3uaSetPsCfg** (U8 ***board***, M3UAPsCfg ****pPsCfg***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
pPsCfg	Pointer to the M3UAPsCfg structure that contains the peer server configuration values. For information, see *M3UAPsCfg* on page 117.

Return values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

Details

Call this function after you configure peer signaling processes for the M3UA layer, but before you configure routes.

An application must set the field values in the M3UAPsCfg structure before calling **M3uaSetPsCfg**. Set the values in any of the following ways:

- Call **M3uaInitPsCfg** to set the fields to default values.
- Set each field value from within the application.
- Call **M3uaInitPsCfg** and then override specific field values before passing the M3UAPsCfg structure to **M3uaSetPsCfg**.

M3uaSetPsCfg is typically called once for each configured peer server.

For more information, see *M3UA peer server configuration* on page 52.

See also

M3uaGetPsCfg

M3uaSetPspCfg

Configures the M3UA layer with the values contained in the peer signaling process structure.

Prototype

STATUS TXM3UAFUNC **M3uaSetPspCfg** (U8 ***board***, M3UAPspCfg ****pPspCfg***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
pPspCfg	Pointer to the M3UAPspCfg structure that contains the peer signaling process configuration values. For information, see *M3UAPspCfg* on page 121.

Returned values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

Details

Call this function after you configure NSAPs for the M3UA layer, but before you configure a peer server.

An application must set the field values in the M3UAPspCfg structure before calling **M3uaSetPspCfg**. Set the values in any of the following ways:

- Call **M3uaInitPspCfg** to set the fields to default values.
- Set each field value from within the application.
- Call **M3uaInitPspCfg** and then override specific field values before passing the M3UAPspCfg structure to **M3uaSetPspCfg**.

M3uaSetPspCfg is typically called once for each configured peer signaling process.

For more information, see *M3UA peer signaling process configuration* on page 52.

See also

M3uaGetPspCfg

M3uaSetRteCfg

Configures the M3UA layer with the values contained in the route configuration structure.

Prototype

STATUS TXM3UAFUNC **M3uaSetRteCfg** (U8 ***board***, M3UARteCfg ****pRteCfg***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
pRteCfg	Pointer to the M3UARteCfg structure that contains the route configuration values. For information, see *M3UARteCfg* on page 127.

Returned values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

Details

Call this function after you configure peer servers for the M3UA layer.

An application must set the field values in the M3UARteCfg structure before calling **M3uaSetRteCfg**. Set the values in any of the following ways:

- Call **M3uaInitRteCfg** to set the fields to default values.
- Set each field value from within the application.
- Call **M3uaInitRteCfg** and then override specific field values before passing the M3UARteCfg structure to **M3uaSetRteCfg**.

M3uaSetRteCfg is typically called once for each configured route.

For more information, see *M3UA route configuration* on page 53.

See also

M3uaGetRteCfg

M3uaSetSctSapCfg

Configures the M3UA layer with the values contained in the SCT SAP configuration structure.

Prototype

STATUS TXM3UAFUNC **M3uaSetSctSapCfg** (U8 ***board***, M3UASctSapCfg ****pSctSapCfg***)

Argument	Description
board	TX board number on which the desired M3UA layer resides. Valid range is 1 through MAXBOARD (currently 32).
pSctSapCfg	Pointer to the M3UASctSapCfg structure that contains the SCT SAP configuration values. For information, see *M3UASctSapCfg* on page 129.

Returned values

Return value	Description
M3UA_SUCCESS	
M3UA_BOARD	Invalid board number.
M3UA_DRIVER	CPI driver error.
M3UA_HANDLE	**M3UAMgmtInit** was not called for the specified board.
M3UA_PARAM	Invalid parameter.
M3UA_RESPONSE	Incorrect response from the board.
M3UA_TIMEOUT	No response from the board.

Details

Call this function after you configure networks for the M3UA layer, but before you configure an NSAP.

An application must set the field values in the M3UASctSapCfg structure before calling **M3uaSetSctSapCfg**. Set the values in any of the following ways:

- Call **M3uaInitSctSapCfg** to set the fields to default values.
- Set each field value from within the application.
- Call **M3uaInitSctSapCfg** and then override specific field values before passing the M3UASctSapCfg structure to **M3uaSetSctSapCfg**.

M3uaSetSctSapCfg is typically called once for each configured SCT SAP.

For more information, see *M3UA SCT SAP configuration* on page 51.

See also

M3uaGetSctSapCfg

8 M3UA management structures

M3UA management structures summary

This section provides an alphabetical reference to the management structures used by M3UA functions. The topics in the structure reference include the structure definition and a table of field descriptions.

The NaturalAccess SIGTRAN Stack implementation uses the following types of M3UA structures:

- Configuration
- Statistics
- Status

Configuration structures

The following table describes the M3UA configuration structures in the NaturalAccess SIGTRAN Stack implementation:

Structure	Description
AssocCfg	SCTP association configuration parameters.
M3UAGenCfg	General configuration parameters.
M3UANsapCfg	NSAP configuration parameters.
M3UANwkCfg	Network configuration parameters.
M3UAPsCfg	Peer server configuration parameters.
M3UAPspCfg	Peer signaling process configuration parameters.
M3UARteCfg	Route configuration parameters.
M3UARtFilter	Route filter parameters.
M3UASctSapCfg	SCT SAP configuration parameters.
M3UA network address sub-structures	Sub-structures that contain network address configuration information.
M3UA timer sub-structures	Sub-structures that contain timer configuration information.

Statistics structures

The following table describes the M3UA statistics structures in the NaturalAccess™ SIGTRAN implementation:

Structure	Description
DateTime	Defines time stamps that indicate when M3UA statistics counters were initialized to zero.
M3UAGenSts	General statistics.
M3UANsapSts	Network SAP statistics.
M3UAPspSts	Peer signaling process statistics.
M3UASctSapSts	SCT SAP statistics.
M3UA statistics sub-structures	Sub-structures that contain various types of statistics.

Status structures

The following table describes the M3UA status structures in the NaturalAccess™ SIGTRAN implementation:

Structure	Description
AssocSta	SCTP association status information.
M3UAAtSta	Address translation status information.
M3UADpcSta	Destination point code (DPC) status information.
M3UAGenSta	General status information.
M3UANSapSta	NSAP status information.
M3UAPsSta	Peer server status information.
M3UAPspRkIdSta	Information about the routing keys associated with the specified peer server process.
M3UAPspSta	Peer signaling process status information.
M3UAPsStaEndp	Status information for the endpoints of an M3UA peer server.
M3UARkSta	Number of dynamically registered routing keys defined in the M3UA layer.
M3UASctSapSta	SCT SAP status information.

AssocCfg

Dependent function: None.

The following AssocCfg structure contains SCTP association parameters for M3UA. Bold text indicates fields you can set. Unbolded fields are either unused or for internal use only. AssocCfg is a substructure to the M3UAPspCfg structure, and its values are set at the time of peer server process configuration.

```
typedef struct _AssocCfg          /* SCTP association configuration */
{
    NetAddr         priDstAddr;     /* primary destination address */
    NetAddrLst      dstAddrLst;     /* destination address list */
    U16             dstPort;        /* destination port */
    U16             locOutStrms;    /* number of streams supported */
    Bool            clientSide;     /* If true, we initiate associations (if IPSP) */
    U8              tos;            /* Type of Service octet */
    U8              spare1[2];            /* Alignment */
} AssocCfg;
```

The AssocCfg structure is a substructure to the M3UAPspCfg. structure. The following table describes the fields in the AssocCfg structure that you can set. These fields are not re-configurable.

Field	Type	Default	Description
priDstAddr	NetAddr	192.168.1.2	Primary destination address of the remote peer used in outgoing association requests, defined by the NetAddr structure. For information, see *NetAddr* on page 132.
dstAddrLst	NetAddrLst	None.	List of destination addresses for the association, defined by the NetAddrLst structure. This list is automatically obtained from the source addresses received in the Initiation (INIT) or Initiation Acknowledgement (INIT ACK) SCTP messages. For more information, see *NetAddrLst* on page 132.
dstPort	U16	2905	Remote SCTP port.
locOutStrms	U16	2	Number of streams supported by this association. Valid range is 1 - 255.
clientSide	Bool	True	In IPSP mode, if this value set to TRUE, an association is established automatically.
tos	U8	0	Type of Service octet value that is sent in all outgoing IP messages over this association. This can later be changed through the *M3uaMgmtCtrl* on page 78 function.

AssocSta

Dependent function: None.

The following AssocSta structure contains SCTP association status information:

```
typedef struct _AssocSta          /* Association status */
{
    U32         spAssocId;              /* Association Id */
    U8          hlSt;                   /* High level state */
    U8          aspSt;                  /* PSP's ASP state */
    Bool        inhibited;              /* Management inhibit status */
    U8          spare1;                 /* alignment */
    U16         nmbAct;                 /* number of active PSs */
    U16         nmbRegPs;               /* number of registered PSs */
    U32         actPs[M3UA_MAX_ACT_PS]; /* list of active PSs */
    U32         regPs[M3UA_MAX_PSID];   /* Registered PSs List */
} AssocSta;
```

The AssocSta structure is a substructure to the M3UAPspSta structure. It contains the following fields:

Field	Type	Description
spAssocId	U32	Association identifier.
hlSt	U8	High level state of the association. Valid values are: M3UA_ASSOC_ACTIVE = Association is up. M3UA_ASSOC_CONGSTART = Association started flow control. M3UA_ASSOC_CONGDROP = Association is dropping data due to congestion. M3UA_ASSOC_DOWN = Association is down.
aspSt	U8	Application server process state for this peer signaling process. Valid values are: M3UA_ASP_ACTIVE = Peer signaling process is actively processing traffic for this peer server. M3UA_ASP_DOWN = Peer signaling process is down. M3UA_ASP_INACTIVE = Peer signaling process is up, but not actively processing traffic for this peer server. M3UA_ASP_UNSUPP = Peer signaling process does not support this peer server.
inhibited	Bool	Indicates whether the association is inhibited by management for maintenance: TRUE = Association is inhibited. FALSE = Association is not inhibited.
spare1	U8	Alignment.
nmbAct	U16	Number of active peer servers.
nmbRegPs	U16	Number of registered peer servers.
actPs[M3UA_MAX_ACT_PS]	U32	Lists the active peer servers supported by this association, where M3UA_MAX_ACT_PS is an array of peer servers.
regPs[M3UA_MAX_PSID]	U32	List of registered peer servers, where M3UA_MAX_PSID is the maximum number of active peer servers per peer signaling process (currently 10).

DateTime (for M3UA)

Dependent function: None.

The following DateTime structure defines time stamps that indicate when M3UA statistics counters were initialized to zero:

```
typedef struct _DateTime           /* date and time */
{
U8 month;       /* month */
U8 day;         /* day */
U8 year;        /* year - since 1900 (eg. 2000 = 100) */
U8 hour;        /* hour - 24 hour clock */
U8 min;         /* minute */
U8 sec;         /* second */
U8 tenths;      /* tenths of second */
U8 fill;
} DateTime;
```

The DateTime structure is a substructure of the M3UAGenSts, M3UANSapSts, M3UAPsSts, M3UAPspSts, and M3UASctSapSts structures. It contains the following fields:

Field	Type	Description
month	U8	Month.
day	U8	Day.
year	U8	(Current year) - 1900.
hour	U8	Hour in UTC time (24 hour clock).
min	U8	Minutes.
sec	U8	Seconds.
tenths	U8	Tenths of seconds.
fill	U8	Not used.

M3UAAtSta

Dependent function: **M3uaAddrTransStatus**

The following M3UAAtSta structure contains M3UA address translation status information:

```
typedef struct _M3UAAtSta   /* Address Translation status */
{
   U16       nmbDpc;        /* Number of entries in DPC table */
   U16       nmbRout;       /* Number of entries in routing table */
} M3UAAtSta;
```

The M3UAAtSta structure contains the following fields:

Field	Type	Description
nmbDpc	U16	Number of DPCs in the DPC table.
nmbRout	U16	Number of routes in the routing table.

M3UADpcSta

Dependent function: **M3uaDpcStatus**

The following M3UADpcSta structure contains DPC status information:

```
typedef struct _M3UAUDpcSta          /* DPC Status */
{
    U32          dpc;                /* DPC value  */
    U8           nwkId;              /* Network Id */
    U8           dpcSt;              /* DPC status */
    U8           congLevel;          /* DPC Congestion Level */
    U8           spare1;             /* alignment */
} M3UADpcSta;
```

The M3UADpcSta structure contains the following fields:

Field	Type	Description
dpc	U32	DPC value.
nwkId	U8	Network identifier for the specified DPC. Valid range is 1 - 255.
dpcSt	U8	DPC state. Valid values are: AVAILABLE = DPC is available. CONGESTED = DPC received more traffic than it can handle. The congestion level is described in the congLevel field. UNAVAILABLE = DPC is not available. UNKNOWN = DPC state is unknown.
congLevel	U8	DPC congestion level. Valid values are: 0 = Not congested. 1 = Congestion level 1 in the queue; not valid in international networks. 2 = Congestion level 2 in the queue; not valid in international networks. 3 = Congestion level 3 in the queue; not valid in international networks.
spare1	U8	Alignment.

M3UAGenCfg

Dependent functions: **M3uaGetGenCfg**, **M3uaInitGenCfg**, **M3uaSetGenCfg**

The following M3UAGenCfg structure contains general M3UA configuration parameters. Bold text indicates fields you can set. Unbolded fields are either unused or for internal use only. They are set to correct values by **M3uaInitGenCfg** and must not be overridden.

```
typedef struct _M3UAGenCfg          /* M3UA general configuration */
{
    U8              nodeType;       /* type of M3UA - SGP or ASP */
    Bool            dpcLrnFlag;     /* DPC learning mode enable */
    Bool            drkmSupp;       /* DRKM supported */
    Bool            drstSupp;       /* DRST supported */
    U16             maxNmbNSap;     /* number of upper SAPs */
    U16             maxNmbSctSap;   /* number of lower SAPs */
    U16             maxNmbNwk;      /* number of Network Appearances */
    U16             maxNmbRtEnt;    /* number of Routing Entries */
    U16             maxNmbDpcEnt;   /* number of DPC entries */
    U16             maxNmbPs;       /* number of Peer Servers */
    U16             maxNmbLps;      /* number of Local PS */
    U16             maxNmbPsp;      /* number of Peer Signalling Processes */
    U16             maxNmbMsg;      /* max MTP3/M3UA messages in transit */
    U16             maxNmbRndRbnLs; /* number of Round Robin Loadshare contexts */
    U16             maxNmbSlsLs;    /* number of SLS based Loadshare contexts */
    U32             maxNmbSls;      /* number of SLS contexts */
    U32             qSize;          /* congestion queue size in M3UA */
    U32             congLevel1;     /* congestion level 1 in the queue */
    U32             congLevel2;     /* congestion level 2 in the queue */
    U32             congLevel3;     /* congestion level 3 in the queue */
    M3UAGenTimerCfg tmr;            /* gen Timer  Config */
    S16             timeRes;        /* timer resolution */
    U16             spare1;         /* alignment */
    U32             debugMask;      /* Debugging mask */
    U32             traceMask;      /* Tracing mask */
    Pst             smPst;          /* Post stack mgr structure */
} M3UAGenCfg;
```

The following table describes the fields in the M3UAGenCfg structure that you can set:

Field	Type	Default	Re-configurable?	Description
nodeType	U8	ASP	No	Type of node. Valid values are: M3UA_TYPE_ASP = ASP or IPSP configuration. M3UA_TYPE_SGP = SGP configuration. Currently only M3UA_TYPE_ASP is supported.
maxNmbNSap	U16	2	No	Maximum number of NSAPs supported simultaneously.
maxNmbNwk	U16	2	No	Maximum number of network contexts supported. There is one network context per variant and network indicator.
maxNmbRtEnt	U16	16	No	Maximum number of route entries supported, including local routes.

Field	Type	Default	Re-configurable?	Description
maxNmbDpcEnt	U16	32	No	Maximum number of DPCs supported, including configured and dynamically learned DPCs.
maxNmbPs	U16	8	No	Maximum number of peer servers supported, including both local and remote peer servers.
maxNmbPsp	U16	16	No	Maximum number of peer signaling processes supported.
maxNmbMsg	U16	128	No	Number of M3UA messages in transit supported.
maxNmbRndRbnLs	U16	4	No	Maximum number of peer servers that can use round-robin load sharing.
maxNmbSlsLs	U16	4	No	Maximum number of peer servers that can use SLS-based load sharing.
maxNmbSls	U32	128	No	Maximum number of SLS values used by all peer servers.
qSize	U32	256	Yes	Outgoing congestion queue size per association. Messages above this limit are dropped.
congLevel1	U32	64	Yes	Congestion level 1 in the queue; not valid in international networks.
congLevel2	U32	128	Yes	Congestion level 2 in the queue; not valid in international networks.
congLevel3	U32	196	Yes	Congestion level 3 in the queue; not valid in international networks.
tmr	Structure	Refer to structure	Refer to structure	Timer configurations, defined by *M3UAGenTimerCfg* on page 133.
debugMask	U32	0	Yes	Enables debug logging. Valid values are: 0 = No debug logging. 0xFFFFFFFF = Enables debug logging. You can also enable debug logging by using the m3uamgr debug command, as described in *m3uamgr commands* on page 140.
trcMask	U32	0	Yes	Enables data tracing. Values are: 0 = No data logging. 0xFFFFFFFF = Enables data tracing. You can also enable data tracing by using the m3uamgr trace command, as described in *m3uamgr commands* on page 140.

M3UAGenSta

Dependent function: **M3uaGenStatus**

The following M3UAGenSta structure contains general M3UA status information:

```
typedef struct _M3UAGenSta          /* General status */
{
    U32         memSize;            /* Reserved memory size */
    U32         memAlloc;           /* Allocated memory size */
    U8          haState;            /* High Availability State */
    U8          isolated;           /* Isolation state.  False = connected to mate. */
    U16         spare2;             /* alignment */
} M3UAGenSta;
```

The M3UAGenSta structure contains the following fields:

Field	Type	Description
memSize	U32	Size of the memory, in bytes, reserved for M3UA.
memAlloc	U32	Size of the memory, in bytes, currently allocated by M3UA.
haState	U8	High availability state of the M3UA layer. Valid values are: M3UA_HAST_BACKUP - M3UA is the backup node in a redundant configuration. M3UA_HAST_PRIMARY - M3UA is the primary node in a redundant configuration. M3UA_HAST_STANDALONE - M3UA is operating in a non-redundant configuration. M3UA_HAST_STARTING - M3UA has not been assigned a high availability state.
isolated	U8	Isolation state. Valid values are: True = Not connected to redundant mate. False = Connected to redundant mate.
spare2	U16	Alignment.

M3UAGenSts

Dependent function: **M3uaGenStatistics**

The following M3UAGenSts structure contains general M3UA statistics:

```
typedef struct _M3UAGenSts              /* General statistics */
{
   DateTime         dt;                 /* Date and time when statistics counters
                                           are initialized to zero */
   Mtp3Sts          txMtp3Sts;          /* MTP3 message statistics - TX */
   Mtp3Sts          rxMtp3Sts;          /* MTP3 message statistics - RX */
   M3UASts          txM3uaSts;          /* M3UA message statistics - TX */
   M3UASts          rxM3uaSts;          /* M3UA message statistics - RX */
   DataSts          liTxDataSts;        /* Lower interface data statistics - TX */
   DataSts          liRxDataSts;        /* Lower interface data statistics - RX */
   DataSts          uiTxDataSts;        /* Upper interface data statistics - TX */
   DataSts          uiRxDataSts;        /* Upper interface data statistics - RX */
   DataErrSts       downDataErrSts;     /* task data error stats - downward */
   DataErrSts       upDataErrSts;       /* task data error statistics - upward */
} M3UAGenSts;
```

The M3UAGenSts structure contains the following fields:

Argument	Type	Description
dt	Structure	Date and time when the statistics counters were reset to zero, defined by *DateTime (for M3UA)* on page 105.
txMtp3Sts	Structure	MTP3 transmit message statistics. **Note:** This field applies to a signaling gateway, which is not currently supported.
rxMtp3Sts	Structure	MTP3 receive message statistics. **Note:** This field applies to a signaling gateway, which is not currently supported.
txM3uaSts	Structure	M3UA transmit message statistics, defined by *M3UASts* on page 137.
rxM3uaSts	Structure	M3UA receive message statistics, defined by M3UASts.
liTxDataSts	Structure	Lower interface transmit statistics, defined by *DataSts* on page 136.
liRxDataSts	Structure	Lower interface receive statistics, defined by DataSts structure.
uiTxDataSts	Structure	Upper interface transmit statistics, defined by DataSts.
uiRxDataSts	Structure	Upper interface receive statistics, defined by DataSts.
downDataErrSts	Structure	Downward or outbound error statistics, defined by *DataErrSts* on page 136.
upDataErrSts	Structure	Upward or inbound error statistics, defined by DataErrSts.

M3UANSapCfg

Dependent functions: **M3uaGetNSapCfg**, **M3uaInitNSapCfg**, **M3uaSetNSapCfg**

The following M3UANSapCfg structure contains M3UA NSAP configuration parameters. Bold text indicates fields you can set. Unbolded fields are either unused or for internal use only. They are set to correct values by **M3uaInitNSapCfg** and must not be overridden.

```
typedef struct _M3UANSapCfg          /* M3UA Network SAP configuration */
{
    S16             sapId;           /* NSAP id */
    U8              nwkId;           /* logical network ID */
    U8              suType;          /* service user protocol type */
    MemoryId        mem;             /* memory region and pool id */
    U8              selector;        /* upper layer selector */
    U8              prior;           /* priority */
    U8              route;           /* route */
    U8              spare1;          /* alignment */
} M3UANSapCfg;
```

The following table describes the fields in the M3UANsapCfg structure that you can set. These fields are not re-configurable.

Field	Type	Default	Description
sapId	S16	0	Identifier for this NSAP. Valid values range from 0 to the result of (genCfg.maxNmbNSap - 1).
nwkId	U8	1	Logical network identifier for this NSAP.
suType	U8	M3UA_SU_ISUP	Type of NSAP service user. Valid values are: M3UA_SU_AAL2 = AAL2 user M3UA_SU_BICC = BICC user M3UA_SU_B_ISUP = B-ISUP user M3UA_SU_DUP = DUP user M3UA_SU_DUPF = DUPF user M3UA_SU_ISUP = ISUP user M3UA_SU_MGCP = MGCP user M3UA_SP_MTP3 = MTP3 user M3UA_SU_MTUP = MTUP user M3UA_SU_S_ISUP = S-ISUP user M3UA_SU_SCCP = SCCP user M3UA_SU_TUP = TUP user

M3UANSapSta

Dependent function: **M3uaNSapStatus**

The following M3UANsapSta structure contains M3UA NSAP status information:

```
typedef struct _M3UANSapSta          /* Upper SAP status */
{
    S16          lclSapId;           /* local  SAP Id */
    S16          remSapId;           /* remote (MTP3/M3UA-User) SAP Id */
    U8           hlSt;               /* High level state */
    U8           spare1;             /* alignment */
    U16          spare2;             /* alignment */
```

The M3UANSapSta structure contains the following fields:

Field	Type	Description
lclSapId	S16	Local SAP identifier.
remSapId	S16	Remote SAP identifier.
hlSt	U8	High level state of this NSAP. Valid values are: BOUND = NSAP is bound. READY = NSAP is ready for use. UNBOUND = NSAP is unbound. WAIT_BIND = NSAP is waiting for bind confirmation. WAIT_OPEN = NSAP is waiting for open confirmation. UNKNOWN = NSAP state is unknown.
spare1	U8	Alignment.
spare2	U8	Alignment.

M3UANSapSts

Dependent function: **M3uaNsapStatistics**

The following M3UANSapSts structure contains M3UA NSAP statistics:

```
typedef struct _M3UANSapSts         /* M3UA Statistics for NSAP */
{
    S16         spId;               /* NSAP Id  */
    U16         spare1;             /* alignment */
    DateTime    dt;                 /* date and time when statistics counters are */
                                    /* initialized to zero */
    Mtp3Sts     txMtp3Sts;          /* MTP3 message statistics - TX */
    Mtp3Sts     rxMtp3Sts;          /* MTP3 message statistics - RX */
    DataSts     txDataSts;          /* SAP data statistics - TX */
    DataSts     rxDataSts;          /* SAP data statistics - RX */
    DataErrSts  dataErrSts;         /* SAP data error statistics - downward */
} M3UANSapSts;
```

The M3UANSapSts structure contains the following fields:

Field	Type	Description
spid	S16	NSAP identifier.
spare1	U16	Alignment.
dt	Structure	Date and time when the statistics counters were reset to zero, defined by *DateTime (for M3UA)* on page 105.
txMtp3Sts	Structure	MTP3 transmit message statistics. **Note:** This field applies to a signaling gateway, which is not currently supported.
rxMtp3Sts	Structure	MTP3 receive message statistics. **Note:** This field applies to a signaling gateway, which is not currently supported.
txDataSts	Structure	NSAP transmit data statistics, defined by *DataSts* on page 136.
rxDataSts	Structure	NSAP receive data statistics, defined by DataSts.
dataErrSts	Structure	NSAP data error statistics, defined by *DataErrSts* on page 136.

M3UANwkCfg

Dependent functions: **M3uaGetNwkCfg**, **M3UAInitNwkCfg**, **M3uaSetNwkCfg**

The following M3UANwkCfg structure contains M3UA network configuration parameters:

```
typedef struct _M3UANwkCfg          /* M3UA network configuration */
{
    U8      nwkId;                  /* network ID */
    U8      ssf;                    /* sub service field */
    U8      dpcLen;                 /* dpc or opc length */
    U8      slsLen;                 /* sls length */
    S16     suSwtch;                /* protocol variant of service user */
    S16     su2Swtch;               /* protocol variant of user of service user */
    U32     nwkApp[M3UA_MAX_PSP];   /* network appearance code */
}M3UANwkCfg;
```

The following table describes the fields in the M3UANwkCfg structure. These fields are not re-configurable.

Field	Type	Default	Description
nwkId	U8	1	Network identifier. Valid range is 1 - 255.
ssf	U8	M3UA_SSF_NAT	Subservice field. Valid values are: M3UA_SSF_INTL = International M3UA_SSF_NAT = National M3UA_SSF_SPARE = Spare M3UA_SSF_RES = Reserved
dpcLen	U8	DPC24	DPC or OPC length. Valid values are: DPC14 = Length for ITU networks. DPC16 = Length for Japanese networks. DPC24 = Length for ANSI networks and other national variants.
slsLen	U8	M3UA_SLS5	SLS length, in bits. Valid values are: M3UA_SLS4 = 4 bits M3UA_SLS5 = 5 bits M3UA_SLS8 = 8 bits
suSwtch	S16	M3UA_SW_ANS	Protocol variant of the M3UA service user, such as ISUP, SCCP, and TUP. Valid values are: M3UA_SW_ANS = ANSI variant M3UA_SW_BICI = BICI variant M3UA_SW_ANS96 = ANSI 96 variant M3UA_SW_ITU = CCITT variant M3UA_SW_CHINA = CHINA variant M3UA_SW_NTT = NTT variant M3UA_SW_TTC = TTC variant

Field	Type	Default	Description
su2Swtch	S16	M3UA_SW2_ANS	Protocol variant for user of the M3UA service user, such as TCAP, which uses SCCP. Valid values are: M3UA_SW2_ANS = TCAP type ANSI M3UA_NWK_ANSI = ANSI network M3UA_SW2_ETS = TCAP type ETSI M3UA_SW2_ITU = TCAP type ITU M3UA_NWK_ITU = ITU network 3UA_NWK_NSS = NSS network M3UA_SW2_TTC = TCAP type TTC M3UA_SW2_UNUSED = su2Swtch is unused
nwkApp[M3UA_MAX_PSP]	32	0	Network appearance code, where M3UA_MAX_PSP is an array of network appearance codes. Network appearance values are determined and configured by network operators on each side of an association.

M3UAPsCfg

Dependent functions: **M3uaGetPsCfg**, **M3UAInitPsCfg**, **M3uaSetPsCfg**

The following M3UAPsCfg structure contains M3UA peer server configuration parameters. Bold text indicates fields you can set. Unbolded fields are either unused or for internal use only. They are set to correct values by **M3uaInitPsCfg** and must not be overridden.

```
typedef struct _M3UAPsCfg           /* Peer Server configuration */
{
    U8              nwkId;          /* Network ID */
    U8              mode;           /* Active/Standby or load sharing */
    U8              loadShareMode;  /* Round robin, SLS mapping etc. */
    Bool            reqAvail;       /* TRUE if required for SPMC availability */
    U32             psId;           /* Peer Server ID */
    U32             routCtx;        /* Routing Context */
    U16             nmbActPspReqd;  /* number of active PSPs sharing load */
    U16             nmbPsp;         /* number of entries in PSP list */
    U16             psp[M3UA_MAX_PSP]; /* ordered list of PSPs */
    Bool            lclFlag;        /* PS type is local if set */
    U8              spare1;         /* alignment */
    U16             spare2;         /* alignment */
} M3UAPsCfg;
```

The following table describes the fields in the M3UAPsCfg structure that you can set:

Field	Type	Default	Re-configurable?	Description
nwkId	U8	1	No	Peer server network identifier.
mode	U8	M3UA_MODE_ACTSTANDBY	Yes	Indicates whether the peer signaling process is in active/standby mode or load sharing mode. Valid values are: M3UA_MODE_ACTSTANDBY = Active/standby mode M3UA_MODE_LOADSHARE = Load sharing mode
loadShareMode	U8	M3UA_LOADSH_RNDROBIN	No	(Remote peer server only, used only if mode = M3UA_MODE_LOADSHARE) Type of load sharing. Valid values are: M3UA_LOADSH_RNDROBIN = Round robin. M3UA_LOADSH_SLS = By link selector.
psId	U32	1	No	Peer server identifier.
routCtx	U32	0	Yes	(Local peer server only) Routing context.
nmbPsp	U16	1	Yes	Total number of peer signaling processes defined for this peer server.

Field	Type	Default	Re-configurable?	Description
psp[M3UA_MAX_PSP]	U16	Psp[0] = 1	Yes, but may affect service	An ordered list of peer signaling processes referenced by pspId. Preference is given to earlier entries when performing failover and failback operations.
lclFlag	Bool	FALSE	No	Indicates whether the peer server is local or remote: TRUE = Local peer server FALSE = Remote peer server

M3UAPsSta

Dependent function: **M3uaPsStatus**

The following M3UAPsSta structure contains M3UA peer server status information:

```
typedef struct _M3UAPsSta           /* Peer Server status */
{
    U32             psId;               /* Peer Server Id */
    U8              asSt;               /* PS's AS state */
    U8              spare1;             /* alignment */
    U16             spare2;             /* alignment */
    M3UAPsStaEndp   psStaEndp[M3UA_MAX_SEP];    /* Status per endpoint */
} M3UAPsSta;
```

The M3UAPsSta structure contains the following fields:

Field	Type	Description
psId	U32	Peer server identifier.
asSt	U8	Application server state of the peer server. Valid values are: ACTIVE = State is active. DOWN = State is down. INACTIVE = State is up, but not active. PENDING = State is queuing, pending reactivation. UNKNOWN = State is unknown.
spare1	U8	Alignment.
spare2	U16	Alignment.
psStaEndp[M3UA_MAX_SEP]	Structure	Status for each peer server endpoint, defined by *M3UAPsStaEndp* on page 120.

M3UAPsStaEndp

Dependent function: None.

The following M3UAsStaEndp structure contains peer signaling process status information for an M3UA endpoint:

```
typedef struct _M3UAPsStaEndp
{
    U8          aspSt[M3UA_MAX_PSP];    /* PS's assoc states (currently 20) */
    U16         nmbPspReg;              /* number of registered assocs */
    U16         nmbAct;                 /* number of active assocs */
    U16         actPsp[M3UA_MAX_PSP];   /* list of active assocs */
    struct
    {
        U32     rCtx;                   /* routing Context */
        Bool    rcValid;                /* routing context valid flag */
        U8      mode;                   /* Dynamically/Static */
        U16     spare1;                 /* alignment */
    } rteCtx[M3UA_MAX_PSP];
} M3UAPsStaEndp;
```

The M3UAPsStaEndp structure is a substructure to M3UAPsSta. It contains the following fields:

Field	Type	Description
aspSt[M3UA_MAX_PSP]	U8	For an endpoint, the peer server's ASP state per peer, where M3UA_MAX_PSP is an array of ASP states. Valid values are: M3UA_ASP_ACTIVE = Peer signaling process is actively processing traffic for this peer server. M3UA_ASP_DOWN = Peer signaling process is down. M3UA_ASP_INACTIVE = Peer signaling process is up, but not actively processing traffic for this peer server. M3UA_ASP_UNSUPP = Peer signaling process does not support this peer server.
nmbPspReg	U16	Number of registered peer signaling processes for this peer server.
nmbAct	U16	Number of currently active peer signaling servers for this peer server.
actPsp[M3UA_MAX_PSP]	U16	List of currently active peer signaling processes for this peer server. M3UA_MAX_PSP is currently set to 20.
rCtx	U32	Routing context corresponding to each configured peer signaling process.
rcValid	Bool	Indicates whether the routing context is valid.
mode	U8	Indicates how the routing context was created: IT_RC_DYNAMIC_REGD = Dynamically created by routing key registration. IT_RC_STATIC_REGD = Statically created by configuration.
spare1	U16	Alignment.

M3UAPspCfg

Dependent functions: **M3uaGetPspCfg**, **M3uaInitPspCfg**, **M3uaSetPspCfg**

The following M3UAPspCfg structure contains M3UA peer signaling parameters. Bold text indicates fields you can set. Unbolded fields are either unused or for internal use only. They are set to correct values by **M3uaInitPspCfg** and must not be overridden.

```
typedef struct _M3UAPspCfg          /* Peer Signalling Process configuration */
{
    U16             pspId;          /* PSP id */
    U8              pspType;        /* PSP client/server type */
    U8              ipspMode;       /* IPSP single ended/double ended type */
    Bool            dynRegRkallwd;  /* PSP is authorized for DRKM mesg */
    U8              dfltLshareMode; /* Default mode for dynamic cfg of ps */
    Bool            nwkAppIncl;     /* Network Appearance flag */
    Bool            rxTxAspId;      /* ASP Id parameter */
    U32             selfAspId;      /* ASP Id to be sent in ASP UP message */
    U8              nwkId;          /* Default network ID */
    Bool            cfgForAllLps;   /* Configure for all local PSs? */
    U16             spare1          /* Alginment
    AssocCfg        assocCfg;       /* Association configuration */
} M3UAPspCfg;
```

Note: Peer signaling process 0 (pspId 0) is the local peer signaling process and is automatically created by the M3UA layer during initialization. It cannot be configured.

The following table describes the fields in the M3UAPspCfg structure that you can set:

Field	Type	Default	Re-configurable?	Description
pspId	U16	1	No	Peer signaling process identifier. Values range from 1 to the result of (MAX_PSP - 1). pspId 0 is created internally and is reserved for the local peer signaling process.
pspType	U8	M3UA_PSPTYPE_IPSP	No	Remote peer signaling process type. Valid values are: M3UA_PSPTYPE_SGP = Process is a signaling gateway process. M3UA_PSPTYPE_IPSP = Process is an IP service process.
ipspMode	U8	M3UA_IPSMODE_SE	No	(Valid when the value of the pspType field is M3UA_PSPTYPE_IPSP) Indicates whether the IP service process mode is single-ended or double-ended. Valid values are: M3UA_IPSMODE_DE = Double-ended mode. M3UA_IPSMODE_SE = Single-ended mode.

Field	Type	Default	Re-configurable?	Description
dynRegRkallwd	Bool	FALSE	No	Indicates whether this peer signaling process can send and receive DRKM (Dynamic Routing Key Management) messages. Valid values are: TRUE = Peer signaling process can send and receive DRKM messages. FALSE = Peer signaling process cannot send or receive DRKM messages.
nwkAppIncl	Bool	FALSE	Yes	Determines whether the optional network appearance parameter is included when communicating with the remote peer. Valid values are: TRUE = Includes the network appearance parameter. FALSE = Does not include the network appearance parameter.
rxTxAspId	Bool	0	Yes	Indicates whether an application server process identifier is required to be sent or received in ASPUP (ASP Up) and ASPUP ACK (ASP Up Acknowledgement) messages. Valid values are: M3UA_RX_ASPID = ASP ID is required in received ASPUP and ASPUP ACK messages. M3UA_TX_ASPID = Identifier is required in transmitted ASPUP and ASPUP ACK messages.
selfAspId	U32	0	Yes, in the ASP Down state only	ASP identifier to send if required by the rxTxAspId field.
nwkId	U8	1	Yes	Default network context identifier for incoming messages, if the messages do not include one.
assocCfg	Structure	Refer to structure.	Refer to structure.	Remote association configuration, defined by *AssocCfg* on page 103.

M3UAPspRkIdSta

Dependent function: **M3uaPspRkIdStatus**

The following M3UAPspRkIdSta structure contains information about the routing keys associated with the specified peer signaling process:

```
typedef struct _M3UAPspRkIdSta                /* PSP's RK-Ids awaiting response */
{
   U16       pspId;                           /* PSP Id */
   S16       sctSuId;                         /* SCT SAP SU Id */
   U32       nmbRkReqPend;                    /* nmb of RKs req pending */
   U32       lclRkId[M3UA_MAX_RK_IN_DRKM];    /* RK-ids */
} M3UAPspRkIdSta;
```

The M3UAPspRkIdSta structure contains the following fields:

Field	Type	Description
pspId	U16	Identifier of the peer signaling process.
sctSuId	S16	SCT SAP service user identifier.
nmbRkReqPend	U32	Number of pending routing key requests. These are routing key requests to the other peer for which no response has been received.
lclRkId[M3UA_RK_IN_DRKM]	U32	List of routing key identifiers.

M3UAPspSta

Dependent function: **M3uaPspStatus**

The following M3UAPspSta structure contains M3UA peer signaling process status information:

```
typedef struct _M3UAPspSta              /* Peer Signalling Process status */
{
    U16         pspId;                  /* PSP Id */
    U16         spare1;                 /* alignment */
    AssocSta    assocSta[M3UA_MAX_SEP]; /*status of PSP's all associations*/
} M3UAPspSta;
```

The M3UAPspSta structure contains the following fields:

Field	Type	Description
pspId	U16	Peer signaling process identifier.
spare1	U16	Alignment.
assocSta[M3UA_MAX_SEP]	Structure	Status of each of the associations associated with this peer signaling process, defined by the AssocSta structure. For information, see *AssocSta* on page 104.

M3UAPspSts

Dependent function: **M3uaPspStatistics**

The following M3UAPspSts structure contains M3UA peer signaling process statistics:

```
typedef struct _M3UAPspSts          /* M3UA Statistics for remote PSP */
{
    U16         pspId;              /* PSP ID */
    U16         spare1;             /* alignment */
    DateTime    dt;                 /* date and time when statistics counters are initial
ized to zero */
    struct {
        M3UASts         txM3uaSts;      /* M3UA message statistics - TX */
        M3UASts         rxM3uaSts;      /* M3UA message statistics - RX */
        DataSts         txDataSts;      /* PSP data statistics - TX */
        DataSts         rxDataSts;      /* PSP data statistics - RX */
        DataErrSts      dataErrSts;     /* PSP data error statistics - upward */
    } assocSts[M3UA_MAX_SEP];
} M3UAPspSts;
```

The M3UAPspSts structure contains the following fields:

Field	Type	Description
pspId	U16	Peer signaling process identifier.
dt	Structure	Date and time when the statistics counters were reset to zero, defined by *DateTime (for M3UA)* on page 105.
txM3uaSts	Structure	Statistics for M3UA messages transmitted by this peer signaling process, defined by *M3UASts* on page 137.
rxM3UASts	Structure	Statistics for M3UA messages received by this peer signaling process, defined by M3UASts.
txDataSts	Structure	Statistics for data transmitted by this peer signaling process, defined by *DataSts* on page 136.
rxDataSts	Structure	Statistics for data received by this peer signaling process, defined by DataSts.
dataErrSts	Structure	Data error statistics, defined by *DataErrSts* on page 136.

M3UARkSta

Dependent function: **M3uaRkStatus**

The following M3UARkSta structure contains the number of dynamically registered routing keys in the M3UA layer:

```
typedef struct _M3UARkSta               /* Dynamic Routing Key status */
{
    U16             nmbRkReg;           /* Num of RK dynamically registered */
    U16             spare1;             /* alignment */
} M3UARkSta;
```

The M3UARkSta structure contains the following field:

Field	Type	Description
nmbRkReq	U16	Number of dynamically registered routing keys in the M3UA layer.
spare1	U16	Alignment.

M3UARteCfg

Dependent functions: **M3uaGetRteCfg**, **M3uaInitRteCfg**, **M3uaSetRteCfg**

The following M3UARteCfg structure contains M3UA route configuration entry parameters. Bold text indicates fields you can set. Unbolded fields are either unused or for internal use only. They are set to correct values by **M3uaInitRteCfg** and must not be overridden.

```
typedef struct _M3UARteCfg          /* Routing entry */
{
    U8              nwkId;          /* network ID */
    U8              rtType;         /* Route Type */
    U16             spare1;         /* alignment */
    M3UARtFilter    rtFilter;       /* M3UA routing filter */
    U32             psId;           /* Peer Server ID */
    Bool            noStatus;       /* TRUE if status suppressed */
    Bool            nSapIdPres;     /* Upper SAP ID present */
    S16             nSapId;         /* Upper SAP ID */
} M3UARteCfg;
```

The following table describes the fields in the M3UARteCfg structure that you can set. These fields are not re-configurable.

Field	Type	Default	Description
nwkId	U8	1	Network identifier. Valid range is 1 - 255.
rtType	U8	M3UA_RTTYPE_PS	Route type. Only valid value is: M3UA_RTTYPE_PS = Route to a peer server.
rtFilter	Structure	Refer to structure	Routing filter parameters specified in *M3UARtFilter* on page 128.
psId	U32	0	(Valid when rtType is M3UA_RTTYPE_PS.) Identifier of the peer server associated with this route.
nSapIdPres	Bool	FALSE	Determines whether an NSAP identifier is associated with this route. Valid values are: TRUE = NSAP identifier is associated with this route. Use this value for local (up) routes. FALSE = NSAP identifier is not associated with this route. Use this value for remote (down) routes.
nSapId	S16	0	(Valid when nSapIdPres is TRUE.) NSAP identifier with which this route is associated.

For more information about configuring routes, see *M3UA route configuration* on page 53.

M3UARtFilter

Dependent function: None.

The following M3UARtFilter structure contains M3UA route filter parameters. Bold text indicates fields you can set. Unbolded fields are either unused or for internal use only. They are set to correct values by **M3uaInitRteCfg** and must not be overridden.

```
typedef struct _M3UARtFilter        /* M3UA routing filter */
{
    U32             dpcMask;        /* DPC wildcard mask */
    U32             dpc;            /* Destination Point Code */
    U32             opcMask;        /* OPC wildcard mask */
    U32             opc;            /* Originator Point Code */
    U8              slsMask;        /* Link Selector wildcard mask */
    U8              sls;            /* Link Selector */
    U8              sioMask;        /* SIO wildcard mask */
    U8              sio;            /* Service Identifier Octet */
    U16             cicStart;       /* Start of CIC range */
    U16             cicEnd;         /* End of CIC range */
    Bool            includeCic;     /* TRUE = include ISUP CIC in filter */
    Bool            includeSsn;     /* TRUE = include SCCP SSN in filter */
    U8              ssn;            /* Subsystem Number */
    Bool            includeTrid;    /* TRUE = include TCAP TRID in filter */
    U32             tridStart;      /* Start of TCAP Transaction ID range */
    U32             tridEnd;        /* End of TCAP Transaction ID range */
} M3UARtFilter;
```

The M3UARtFilter structure is a substructure to M3UARteCfg. The following table describes the fields in the M3UARtFilter structure. These fields are not re-configurable.

Field	Type	Default	Description
dpcMask	U32	0xFFFFFF	Wildcard mask for the destination point code (DPC).
dpc	U32	0	DPC associated with this route.
opcMask	U32	0	Wildcard mask for the originating point code (OPC). Leave the default value at 0, if OPC matching is not used.
opc	U32	0	OPC associated with this route, if any.
sioMask	U8	0	Wildcard mask for the service information octet (SIO). Leave the default value at 0, if SIO matching is not used. Set to 0xF if SIO is used for routing.
sio	U8	0	SIO associated with this route, if any.

M3UASctSapCfg

Dependent functions: **M3uaGetSctSapCfg**, **M3uaInitSctSapCfg**, **M3uaSetSctSapCfg**

The following M3UASctSapCfg structure contains M3UA SCT SAP configuration parameters. Bold text indicates fields you can set. Unbolded fields are either unused or for internal use only. They are set to the correct value by **M3uaInitSctSapCfg** and must not be overridden.

```
typedef struct _M3UASctSapCfg       /* SCT SAP configuration */
{
    S16             suId;           /* service user SAP ID */
    U16             srcPort;        /* source port for listening endpoint */
    NetAddrLst      srcAddrLst;     /* source address list */
    TmrCfg          tmrPrim;        /* lower SAP primitive time */
    TmrCfg          tmrSta;         /* congestion poll time */
    MemoryId        mem;            /* memory region and pool ID */
    U8              selector;       /* selector for SCT */
    U8              spare1;         /* alignment */
    U16             procId;         /* processor ID */
    U8              ent;            /* entity */
    U8              inst;           /* instance */
    U8              prior;          /* priority */
    U8              route;          /* route */
    S16             spId;           /* service provider ID */
    U16             spare2;         /* alignment */
} M3UASctSapCfg;
```

Only one SCT SAP may be configured. The following table describes the fields in the M3UASctSapCfg structure that you can set. These fields are not re-configurable.

Field	Type	Default	Description
suId	S16	0	M3UA identifier for this SCT SAP. The only valid value is 0.
srcPort	U16	2905	Source port for the listening endpoint.
spId	S16	0	SCTP identifier for this SCT SAP. The only valid value is 0.

M3UASctSapSta

Dependent function: M3uaSctSapStatus

The following M3UASctSapSta structure contains M3UA SCT SAP status information:

```
typedef struct _M3UASctSapSta           /* lower SAP status */
{
    S16             suId;               /* Service user SAP Id */
    U8              hlSt;               /* High level state */
    U8              spare1;             /* alignment */
    U32             spEndpId;           /* SP ID of my endpoint */
    U16             nmbActAssoc;        /* Number of active associations */
    U16             spare2;             /* alignment */
} M3UASctSapSta;
```

The M3UASctSapSta structure contains the following fields:

Field	Type	Description
suid	S16	M3UA ID for this SAP.
hlSt	U8	High level state of this SAP. Valid values are: BOUND = SAP is bound. READY = SAP is ready for use. UNBOUND = SAP is unbound. WAIT_BIND = Waiting for bind confirmation. WAIT_OPEN = Waiting for open confirmation. UNKNOWN = SAP state is unknown.
spare1	U8	Alignment.
spEndpId	U32	SAP ID of local endpoint.
nmbActAssoc	U16	Number of active associations over this SCT SAP.
spare2	U16	Alignment.

M3UASctSapSts

Dependent function: **M3uaSctSapStatistics**

The following M3UASctSapSts structure contains M3UA SCT SAP statistics:

```
typedef struct _M3UASctSapSts       /* M3UA Statistics for SCTSAP */
{
    S16         suId;               /* Service user SAP Id */
    U16         spare1;             /* alignment */
    DateTime    dt;                 /* date and time when statistics counters */
                                    /* are initialized to zero */
    DataSts     txDataSts;          /* SAP data statistics - TX */
    DataSts     rxDataSts;          /* SAP data statistics - RX */
} M3UASctSapSts;
```

The M3UASctSapStsstructure contains the following fields:

Field	Type	Description
suid	S16	M3UA identifier for this SCT SAP.
spare1	U16	Alignment.
dt	Structure	Date and time when the statistics counters were reset to zero, defined by *DateTime (for M3UA)* on page 105.
txDataSts	Structure	Data transmit statistics, defined by *DataSts* on page 136.
rxDataSts	Structure	Data receive statistics, defined by *DataSts* on page 136.

M3UA network address sub-structures

This topic describes the following M3UA network address sub-structures:

- NetAddrLst
- NetAddr

NetAddrLst

The following NetAddrLst structure defines an array of network addresses. NetAddrLst is a substructure to the AssocCfg and M3UASctSapCfg structures.

```
typedef struct _NetAddrLst
{
   U32            nmb;                          /* Number of Network Addresses */
   NetAddr       nAddr[SCT_MAX_NET_ADDRS];      /* List of Network Addresses */
} NetAddrLst;
```

The following table describes the fields in the NetAddrLst structure. These fields are not re-configurable.

Field	Type	Description
nmb	U32	Number of network addresses.
nAddr[SCT_MAX_NET_ADDRS]	NetAddr	List of network addresses, defined by the NetAddr structure, where SCT_MAX_NET_ADDRS is an array of network addresses. For information, see NetAddr.

NetAddr

The following NetAddr structure defines an IPv4 or IPv6 network address. NetAddr is a substructure to the AssocCfg and NetAddrLst structures.

```
typedef struct NetAddr
{
   U8       type;       /* type of network address */
   union
      {
           U32       ipv4NetAddr;           /* IP network address */
           U8        ipv6NetAddr[16];       /* IPv6 network address */
      }u;
} NetAddr;
```

The following table describes the fields in the NetAddr structure. These fields are not re-configurable.

Field	Type	Description
type	U8	Network address type. Valid values are: CM_NETADDR_IPV4 = IPV4 CM_NETADDR_IPV6 = IPV6 **Note:** IPV6 is not supported in the current release.
ipv4NetAddr	U32	IPv4 network address.
ipv6NetAddr[16]	U8	Not supported in the current release.

M3UA timer sub-structures

This topic describes the following M3UA timer sub-structures:

- M3UAGenTimerCfg
- TmrCfg

M3UAGenTimerCfg

The following M3UAGenTimerCfg structure contains general timer configurations. M3UAGenTimerCfg is a substructure to the M3UAGenCfg structure.

```
typedef struct _M3UAGenTimerCfg /* M3UA general timer configuration structure */
{
    U8      nmbAspUp1;          /* initial number of ASPUP */
    U8      maxNmbRkTry;        /* initial number of DRKM msgs */
    U16     spare1;             /* alignment */
    TmrCfg  tmrRestart;         /* restart hold-off time */
    TmrCfg  tmrMtp3Sta;         /* MTP3 status poll time */
    TmrCfg  tmrAsPend;          /* AS-PENDING time */
    TmrCfg  tmrHeartbeat;       /* heartbeat period */
    TmrCfg  tmrAspUp1;          /* initial time between ASPUP */
    TmrCfg  tmrAspUp2;          /* steady-state time between ASPUP */
    TmrCfg  tmrAspDn;           /* time between ASPDN */
    TmrCfg  tmrAspM;            /* timeout value for ASPM messages */
    TmrCfg  tmrDaud;            /* time between DAUD */
    TmrCfg  tmrDrkm;            /* time between DRKM msgs */
    TmrCfg  tmrDunaSettle;      /* time to settle DUNA message */
    TmrCfg  tmrSeqCntrl;        /* Sequence Control timer */
} M3UAGenTimerCfg;
```

The following table describes the fields in the M3UAGenTimerCfg structure. These fields are re-configurable. All timer values are specified in milliseconds.

Fields not listed in the table are either unused or for internal use only, and should not be modified.

Field	Type	Default	Description
nmbAspUp1	U8	3	Number of initial attempts at sending ASPUP messages at interval tmrAspUp1 before sending them at interval tmrAspUp2.
maxNmbRkTry	U8	3	Number of DRKM attempts before failing.
spare1	U16	NA	Alignment.
tmrRestart	Structure	1000	(ASP only) Restart hold-off time, defined by the TmrCfg structure. This timer starts when all point codes in a network are marked Inactive by DUNA (Destination Unavailable) messages received from an SG. When the timer expires, all related PS states change to Inactive.
tmrMtp3Sta	Structure	0	(SG only) MTP3 status poll time, defined by the TmrCfg structure.
tmrAsPend	Structure	50000	Time for which a peer server can remain in an AS_PENDING state, defined by the TmrCfg structure.
tmrAspUp1	Structure	2000	Initial time between ASPUP (ASP Up) retries, defined by the TmrCfg structure.
tmrAspUp2	Structure	2000	Steady state time between ASPUP retries, defined by the TmrCfg structure.
tmrAspDn	Structure	2000	Time between ASPDN (ASP Down) retries, defined by the TmrCfg structure.
tmrAspM	Structure	2000	Time to wait before failing, after sending ASPAC (ASP Active) or ASPIA (ASP Inactive) messages, defined by the TmrCfg structure.
tmrDaud	Structure	2000	Time between Destination State Audit (DAUD) messages, defined by the TmrCfg structure.
tmrDrkm	Structure	2000	Time between DRKM (Dynamic Routing Key Management) messages, defined by the TmrCfg structure.
trmDunaSettle	Structure	1000	(SG only) Time to settle DUNA (Destination Unavailable) messages.
tmrSeqCntrl	Structure	1000	Delay used when diverting traffic to maintain sequencing, defined by the TmrCfg structure.

TmrCfg

The following TmrCfg structure configures a timer. TmrCfg is a substructure to the M3UAGenCfg structure.

```
typedef struct tmrCfg           /* timer configuration structure */
{
    Bool    enb;                /* enable */
    U8      spare1;             /* alignment */
    U16     val;                /* value */
}TmrCfg;
```

The following table describes the fields in the TmrCfg structure. These fields are re-configurable.

Field	Type	Description
enb	Bool	Indicates whether the timer is enabled: 0 = Disabled 1 = Enabled
spare1	U8	Alignment.
val	U16	Timer value, in ms.

M3UA statistics sub-structures

This topic describes the following M3UA statistics sub-structures:

- CongSts
- DataSts
- DataErrSts
- MtpSts
- M3uaSts

CongSts

The following CongSts substructure contains congestion statistics for M3UA peer signaling processes. CongSts is a substructure to the M3UAPspSts structure.

```
typedef struct _CongSts
{
U32         cong;          /* Association congested */
U32         cong1;         /* Association at congestion level 1 */
U32         cong2;         /* Association at congestion level 2 */
U32         cong3;         /* Association at congestion level 3 */
Ticks       durCong;       /* Duration of association congestion */
} CongSts;
```

DataSts

The following DataSts substructure contains data statistics for M3UA peer signaling processes and SCT SAPs. DataSts is a substructure to the M3UAGenSts, M3UAPspSts, and M3UASctSapSts structures.

```
typedef struct _DataSts
{
   U32       nPdus;          /* Number of PDUs */
   U32       pduBytes;       /* Total size of PDUs */
} DataSts;
```

DataErrSts

The following DataErrSts substructure contains data error statistics for M3UA peer signaling processes and SCT SAPs. DataErrSts is a substructure to the M3UAGenSts and M3UAPspSts structures.

```
typedef struct _DataErrSts
{
    U32       dropNoRoute;          /* Data dropped, no route found */
    U32       dropPcUnavail;        /* Data dropped, point code unavail */
    U32       dropPcCong;           /* Data dropped, point code congested */
    U32       dropNoPspAvail;       /* Data dropped, no PSP avail */
    U32       dropNoNSapAvail;      /* Data dropped, no NSAP avail */
    U32       dropLoadShFail;       /* Data dropped, load-sharing failed */
    U32       dropMmhFail;          /* Data dropped, M3UA message failed */
    U32       dataQCong;            /* Data queued in congestion queue */
    U32       dataQAsPend;          /* Data queued in AS-PENDING queue */
} DataErrSts;
```

Mtp3Sts

The following Mtp3Sts structure contains signaling gateway statistics. It is currently unsupported.

```
typedef struct _Mtp3Sts
{
    U32         data;           /* MTP3 Data primitives */
    U32         pause;          /* MTP3 Pause status primitives */
    U32         resume;         /* MTP3 Resume status primitives */
    U32         cong;           /* MTP3 Cong status primitives */
    U32         drst;           /* MTP3 Restrict status primitive */
    U32         rstBeg;         /* MTP3 Reset begin status primitives */
    U32         rstEnd;         /* MTP3 Reset end status primitives */
    U32         upu;            /* MTP3 UPU status primitives */
} Mtp3Sts;
```

M3UASts

The following M3UASts structure contains service function statistics for the M3UA layer. M3uaSts is a substructure to the M3UAGenCfg, M3UAPsSts, and M3UAPspSts structures.

```
typedef struct _M3UASts
{
    U32       data;           /* M3UA DATA messages */
    U32       duna;           /* M3UA DUNA messages */
    U32       dava;           /* M3UA DAVA messages */
    U32       daud;           /* M3UA DAUD messages */
    U32       scon;           /* M3UA SCON messages */
    U32       dupu;           /* M3UA DUPU messages */
    U32       drst;           /* M3UA DRST messages */
    U32       regReq;         /* M3UA REG-REQ messages */
    U32       deRegReq;       /* M3UA DEREG-REQ messages */
    U32       regRsp;         /* M3UA REG-RSP messages */
    U32       deRegRsp;       /* M3UA DEREG-RSP messages */
    U32       aspUp;          /* M3UA ASPUP messages */
    U32       aspUpAck;       /* M3UA ASPUP ACK messages */
    U32       aspDn;          /* M3UA ASPDN messages */
    U32       aspDnAck;       /* M3UA ASPDN ACK messages */
    U32       aspAc;          /* M3UA ASPAC messages */
    U32       aspAcAck;       /* M3UA ASPAC ACK messages */
    U32       aspIa;          /* M3UA ASPIA messages */
    U32       aspIaAck;       /* M3UA ASPIA ACK messages */
    U32       hBeat;          /* M3UA HBEAT messages */
   U32       hBeatAck;       /* M3UA HBEAT ACK messages */
    U32       err;            /* M3UA ERR messages */
    U32       notify;         /* M3UA NTFY messages */
} M3UASts;
```

9 m3uamgr utility

m3uamgr overview

m3uamgr is an M3UA utility for managing and troubleshooting SS7 signaling. Use *m3uamgr* to:

- Send ASP or association messages to the M3UA layer
- Register or de-register routing keys
- Switch the *m3uamgr* context to the other board on the host
- Enable or disable SCT SAPs
- Display statistics for the M3UA layer and M3UA entities
- Display status information and configuration values for the M3UA layer and M3UA entities
- Dynamically change the Type of Service (TOS) sent in the IP header of messages over an association

Complete the following steps to use *m3uamgr*:

Step	Action

1	Start *m3uamgr* by entering the following command:
	`m3uamgr`
	The following information displays:
	`m3uamgr: sample M3UA management application version 1.0 Jun  3 2008`
	`m3uamgr[1]>`

2	Enter *m3uamgr* commands at the command prompt. For information, see *m3uamgr commands* on page 140.
3	End an *m3uamgr* session by typing **q** and pressing **Enter**.

m3uamgr commands

The following table describes the available *m3uamgr* commands. The commands and arguments can be in upper, lower, or mixed case.

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td>?</td><td>Displays the usage of all supported m3uamgr commands.
<code>?</code></td></tr>
<tr><td>asp</td><td>Causes the M3UA layer to send an ASP message.
<code>asp message <pspId></code>
where <i>message</i> is one of the following arguments:
<table>
<tr><th>Argument</th><th>Description</th></tr>
<tr><td>ac</td><td>ASPAC (ASP Active) message.</td></tr>
<tr><td>dn</td><td>ASPDN (ASP Down) message.</td></tr>
<tr><td>ia</td><td>ASPIA (Inactive) message.</td></tr>
<tr><td>up</td><td>ASPUP (ASP Up) message.</td></tr>
</table>
<pspId> is the ID of the psp to send the mesage to.</td></tr>
<tr><td>assoc</td><td>Causes the M3UA layer to initiate, terminate, inhibit, or uninhibit an association.
<code>assoc message <pspId></code>
where <i>message</i> is one of the following arguments:
<table>
<tr><th>Argument</th><th>Description</th></tr>
<tr><td>est</td><td>Association establish message.</td></tr>
<tr><td>inh</td><td>Association inhibit message.</td></tr>
<tr><td>term</td><td>Association terminate message.</td></tr>
<tr><td>uni</td><td>Association uninhibit message.</td></tr>
</table>
<pspId> is the ID of the psp related to the association.</td></tr>
<tr><td>board</td><td>Switches the m3uamgr communication to the board specified by boardNum.
<code>board boardNum</code></td></tr>
<tr><td>debug</td><td>Enables or disables debug logging for the M3UA layer.
<code>debug operation</code>
where operation is one of the following arguments:
<table>
<tr><th>Argument</th><th>Valid values</th></tr>
<tr><td>ena</td><td>Enables debug logging.</td></tr>
<tr><td>dis</td><td>Disables debug logging.</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td>rk</td><td>Registers or de-registers the routing key.
<code>rk operation</code>
where operation is one of the following arguments:
<table><tr><th>Argument</th><th>Description</th></tr><tr><td>reg</td><td>Registers the routing key.</td></tr><tr><td>dereg</td><td>De-registers the routing key.</td></tr></table></td></tr>
<tr><td>sctsap</td><td>Enables or disables the SCT SAP.
<code>sctsap operation</code>
where operation is one of the following arguments:
<table><tr><th>Argument</th><th>Description</th></tr><tr><td>ena</td><td>Enables the SCT SAP.</td></tr><tr><td>dis</td><td>Disables the SCT SAP.</td></tr></table></td></tr>
<tr><td>stats</td><td>Displays and optionally clears current statistics for the M3UA layer or an M3UA entity.
<code>stats entity [reset]</code>
where entity is one of the following arguments:
<table><tr><th>Argument</th><th>Description</th></tr><tr><td>m3ua</td><td>Displays MTP 3 counters, M3UA counters, data statistics, and error statistics for the M3UA layer.</td></tr><tr><td>psp pspId</td><td>Displays counters, data statistics, and error statistics for the association with the peer signaling process specified by pspId.</td></tr><tr><td>sctsap sapId</td><td>Displays data statistics for the SCT SAP specified by sapId.</td></tr><tr><td>nsap sapId</td><td>Displays counters, data statistics, and error statistics for the NSAP specified by sapId.</td></tr></table>If specified, reset sets all statistics for the specified object to zero (0) immediately after the current statistics are displayed.
See m3uamgr statistics command examples on page 143 for more information.</td></tr>
</table>

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td>status</td><td>Displays status information and configuration values for the M3UA layer and M3UA entities.
<code>status entity</code>
where entity is one of the following arguments:
<table><tr><th>Argument</th><th>Description</th></tr><tr><td>addr</td><td>Displays the address translation table status for the M3UA layer.</td></tr><tr><td>dpc nwkId dpc</td><td>Displays the status of the specified destination point code
The DPC can be specified as a decimal number (for example, 2) or as a hexadecimal number (for example, 0x2).</td></tr><tr><td>m3ua</td><td>Displays general status information and configuration values for the M3UA layer.</td></tr><tr><td>nsap sapId</td><td>Displays status information and configuration values for the NSAP specified by sapId.</td></tr><tr><td>nwk nwkId</td><td>Displays configuration values for the network specified by nwkId.</td></tr><tr><td>ps psId</td><td>Displays status information and configuration values for the peer server specified by psId.</td></tr><tr><td>psp pspId</td><td>Displays status information and configuration values for the peer signaling process specified by pspId.</td></tr><tr><td>psprk pspId</td><td>Displays routing key status information for the PSP specified by pspId.</td></tr><tr><td>rk</td><td>Displays the number of registered routing keys in the M3UA layer.</td></tr><tr><td>sctsap sapId</td><td>Displays status information and configuration values for the SCT SAP specified by sapId.</td></tr></table>See m3uamgr status command examples on page 146 for more information.</td></tr>
<tr><td>trace</td><td>Enables or disables data tracing for the M3UA layer.
<code>trace operation</code>
where operation is one of the following arguments:
<table><tr><th>Argument</th><th>Valid values</th></tr><tr><td>ena</td><td>Enables data tracing.</td></tr><tr><td>dis</td><td>Disables data tracing.</td></tr></table></td></tr>
<tr><td>tos</td><td>Sets the Type of Service (TOS) octet in the IP header of all outgoing message over an association.
<code>Tos <<i>pspId</i>> <<i>tosVal</i>></code>
Where <<i>pspId</i>> is the ID of the PSP related to the association where TOS is to be changed. <<i>tosVal</i>> is the new TOS value to use.</td></tr>
<tr><td>q</td><td>Quits the m3uamgr application.</td></tr>
</table>

m3uamgr statistics command examples

This topic provides examples of the following *m3uamgr* statistics commands:

- stats m3ua
- stats nsap
- stats psp
- stats sctsap

stats m3ua

The following example displays general statistics for the M3UA layer:

```
m3uamgr[1]>stats m3ua
=========M3UA General Statistics since 8-15-2008 16:43:11=========
        MTP3 Statistics:
        Counter       Tx              Rx
        ======================================================
        Data          51777           7
        Pause         2               0
        Resume        5               0
        Cong          0               0
        Restrict      0               0
        ResetBegin    0               0
        ResetEnd      0               0
        UPUs          0               0

        M3UA Statistics:
        Counter       Tx              Rx
        ======================================================
        Data          2               51777
        DUNA          0               0
        DAVA          0               0
        DAUD          0               0
        SCON          0               0
        DUPU          0               0
        DRST          0               0
        REGREQ        0               0
        DEREGREQ      0               0
        REGRSP        0               0
        DEREGRSP      0               0
        ASPUP         3               1
        ASPUPACK      1               3
        ASPDN         0               0
        ASPDNACK      0               0
        ASPAC         3               2
        ASPACACK      4               6
        ASPIA         0               0
        ASPIAACK      0               0
        HBEAT         0               0
        HBEATACK      0               0
        ERROR         0               0
        NOTIFY        4               12

        DATA Statistics:
        Tx/Rx           PDUs           Bytes
        ======================================================
        LowerIntf Tx    17               484
        LowerIntf Rx    51801            7870480
        UpperIntf Tx    51777            6109498
        UpperIntf Rx    7                879

        Error Statistics:
        Error           Upward          Downward
        ======================================================
        NoRoute         0               0
```

```
        PC-Unavail      0               0
        PC-Cong         0               0
        PSP-Unavail     0               0
        NSAP-Unavail    0               0
        LoadSH-Fail     0               0
        M3uaMsgFail     0               0
        Que-Cong        0               0
        AS-Pending      0               0
```

stats nsap

The following example displays statistics for NSAP 1:

```
m3uamgr[1]>stats nsap 1
=========M3UA NSAP 1 Statistics since 8-15-2008 16:43:20=========
        MTP3 Statistics:
        Counter       Tx              Rx
        ======================================================
        Data          2               51898
        Pause         3               0
        Resume        3               0
        Cong          0               0
        Restrict      0               0
        ResetBegin    0               0
        ResetEnd      0               0
        UPUs          0               0

        DATA Statistics:
        Tx/Rx       PDUs            Bytes
        ======================================================
        Tx          2               48
        Rx          51898           6123776

        Error Statistics:
        Error        Msg Dropped
        ======================================================
        NoRoute           0
        PC-Unavail        2
        PC-Cong           0
        PSP-Unavail       0
        NSAP-Unavail      0
        LoadSH-Fail       0
        M3uaMsgFail       0
        Que-Cong          0
        AS-Pending        2
```

stats psp

The following example displays statistics for peer signaling process 1:

```
m3uamgr[1]>stats psp 1
=======M3UA PSP 1 Statistics since 8-15-2008 16:43:11=======
        M3UA Statistics for Association 0:
        Counter     Tx              Rx
        ========================================================
        Data        2               51919
        DUNA        0               0
        DAVA        0               0
        DAUD        0               0
        SCON        0               0
        DUPU        0               0
        DRST        0               0
        REGREQ      0               0
        DEREGREQ    0               0
        REGRSP      0               0
        DEREGRSP    0               0
        ASPUP       4               1
        ASPUPACK    1               4
        ASPDN       0               0
        ASPDNACK    0               0
        ASPAC       4               3
        ASPACACK    6               8
        ASPIA       0               0
        ASPIAACK    0               0
        HBEAT       0               0
        HBEATACK    0               0
        ERROR       0               0
        NOTIFY      6               16

        DATA Statistics for Association 0
        Tx/Rx       PDUs            Bytes
        ========================================================
        Tx          23              632
        Rx          51951           7892260

        Error Statistics for Association 0
        Error        Msg Dropped
        ========================================================
        NoRoute           0
        PC-Unavail        0
        PC-Cong           0
        PSP-Unavail       0
        NSAP-Unavail      0
        LoadSH-Fail       0
        M3uaMsgFail       0
        Que-Cong          0
        AS-Pending        0
```

stats sctsap

The following example displays statistics for SCT SAP 0:

```
m3uamgr[2]>stats sctsap 0
=========M3UA SCTSAP 0 Statistics since 8-15-2008 16:43:20=========
        DATA Statistics:
        Tx/Rx       PDUs            Bytes
        ========================================================
        Tx          51932           7890028
        Rx          20              564
```

m3uamgr status command examples

This topic provides examples of the following *m3uamgr* status commands:

- status addr
- status dpc
- status m3ua
- status nsap
- status nwk
- status ps
- status psp
- status psprk
- status rk
- status sctsap

status addr

The following example displays the address translation table status for the M3UA layer:

```
m3uamgr[1]>status addr
====================Address Translation Status====================
Number of DPCs      = 2
Number of Routes    = 4
```

status dpc

The following example displays status information for DPC 1 in network 1:

```
m3uamgr[1]>status dpc 1 1
======================DPC Status ======================
Network Id      = 1
       DPC      = 0x1
DPC Status      = AVAILABLE
Cong level      = 0
```

status m3ua

The following example displays general status information and configuration values for the M3UA layer:

```
m3uamgr[1]>status m3ua
M3UA Memory Size        = 53145
M3UA Memory Allocated   = 7571
M3UA HA State           = STANDALONE
====================Current M3UA Configuration====================
NodeType       = ASP     dpcLrnFlag     = FALSE   maxNmbNSap     = 2
maxNmbSctSap   = 1       maxNmbNwk      = 2       maxNmbRtEnt    = 16
maxNmbDpcEnt   = 32      maxNmbPs       = 8       maxNmbPsp      = 16
maxNmbMsg      = 128     maxNmbRndRbnLs= 4        maxNmbSlsLs    = 4
maxNmbSls      = 128     drkmSupp       = FALSE   drstSupp       = FALSE
qSize          = 256     congLevel1     = 64      congLevel2     = 128
congLevel3     = 196     timeRes        = 1
======================Current M3UA Timers======================
tmrRestart     = 10000   tmrMtp3Sta     = 0       tmrAsPend      = 5000
tmrHeartbeat   = 0       tmrAspUp1      = 2000    tmrAspUp2      = 2000
nmbAspUp1      = 3       tmrAspDn       = 2000    tmrAspM        = 2000
tmrDaud        = 2000    tmrDrkm        = 2000    maxNmbRkTry    = 3
tmrDunaSettle = 1000     tmrSeqCntrl    = 1000
```

status nsap

The following example displays status information and configuration values for NSAP 0:

```
m3uamgr[1]>status nsap 0
====================NSAP 0 Status=========================
Local SAP ID        = 0
Remote SAP ID       = 0
High Level Status   = BOUND
====================NSAP 0 Configuration==================
SAP ID      = 0       Network Id  = 1       Su Type     = ISUP
Selector    = UNKNWN  Mem Pool    = 0       Mem Region  = 0
Priority    = 0       Route       = 0
```

status nwk

The following example displays configuration values for network 1:

```
m3uamgr[1]>status nwk 1
====================Network 1 Configuration===============
ssf         = NAT       dpcLen     = DPC24     slsLen     = 5
suSwtch     = ANSI      suSwtch    = ANSI
NetAppear[1 ] = 1234
```

status ps

The following example displays status information and configuration values for peer server 1:

```
m3uamgr[1]>status ps 1
====================Peer Server(PS) 1 Status==============
PS ID       = 1       AS STATE    = ACTIVE
=======================PS STATUS PER SEP=====================
Active PSPs    = 0       Registerd PSPs = 0
PspId = 1     ASP State = UNSUP RtCtx = 0     Valid = NO    Mode = STATIC
====================Current PS 1 Configuration============
PS Id       = 1       Net Id      = 1       Mode       = ACT/STBY
RoutCtx     = 10      LoadShMode  = RNDRBN  ReqAvail   = TRUE
ActPspReqd = 1       NmbPsps     = 1       PSLocal    = YES
Psp Id      = 1
```

status psp

The following example displays status information and configuration values for peer signaling process 1:

```
m3uamgr[1]>status psp 1
====================PSP 1 Association Status==============
AssocId = 0       State = ACTIVE  ASP State = ACTV    Inhibit = NO
=================Active PSs (4)=================
PsId   = 1
PsId   = 2
PsId   = 3
PsId   = 4
=================Registered PSs (0)==============
None
====================Current PSP 1 Confuration=================
pspType       = IPSP    ipspMode      = DBL END dynRegAllow   = NO
loadShareMode = RNDRBN  nwkAppIncl    = NO      rxTxAspId     = NO
selfAspId     = 0       nwkId         = 1       PriDestAddr   =
10.51.1.103
DestPort      = 2905    locOutStrms   = 2       TOS           = 0
```

status psprk

The following example displays routing key status information for peer signaling process 1:

```
m3uamgr[1]>status psprk 1
================== Routing Key Status for PSP 1 ================
SCT SuID        = 0       Pending RK Req. = 0
Local RK IDs
1. 0       2. 0       3. 0       4. 0
```

status rk

The following example displays the number of registered routing keys in the M3UA layer:

```
m3uamgr[1]>status rk
====================== Registered Routing Keys ====================
Number RK registered = 0
```

status sctsap

The following example displays status information and configuration values for SCT SAP 0:

```
m3uamgr[1]>status sctsap 0
=========================User SAP 0 Status=======================
State        = READY   EndPoint SP  = 0       Active Assoc = 1
=================Current Configuration for SAP 0=================
tmrPrim    = 512001  tmrSta     = 0       srcPort    = 2905
Selector   = 1       Mem.Pool   = 0       Mem.Region = 0
ProcId     = 1792    Ent        = 136     Inst       = 0
Prior      = 0       Route      = 0       SP Id      = 0
SrcAddrLst[1] = 10.51.1.102
```

10 Managing the SCTP layer

Configuring SCTP entities

Configure SCTP entities after calling **SctpMgmtInit**. You must configure SCTP entities in the following order:

- General SCTP configuration
- SCT SAP
- TSAP

For information about the initial configuration of SCTP, see the *Dialogic® NaturalAccess™ Signaling Software Configuration Manual*.

General SCTP configuration

General configuration parameters define and control the general operation of the SCTP layer. The configurable attributes for the general SCTP configuration include the:

- Maximum number of queued datagrams, open SCTP associations, and other elements to control memory allocation
- Timer values

Define the general parameters for SCTP once at board download time, before you configure the other SCTP entities. To define the general parameters, follow these steps:

Step	Action
1	Call **SctpInitGenCfg** to initialize the general configuration parameter structure (SCTPGenCfg) to default values.
2	Change the parameter values, as appropriate.
3	Call **SctpSetGenCfg** to set the configuration.

After the general configuration is defined, you can optionally change the parameters in the SCTPGenReCfg structure. To change these parameter values, follow these steps:

Step	Action
1	Call **SctpGetGenCfg** to obtain the current general configuration values.
2	Change the parameter values, as appropriate.
3	Call **SctpSetGenCfg** to set the configuration with the specified values.

You must download the board again to change any of the parameter values in the SCTPGenCfg structure. For more information, see *SctpInitGenCfg* on page 162, *SctpSetGenCfg* on page 171, *SCTPGenCfg* on page 181, and *SCTPGenReCfg* on page 183.

SCTP SCT SAP configuration

The SCT SAP defines the interface that the M3UA application uses to access the SCTP layer. The configurable attributes of the SCT SAP includes:

- The SCT SAP identifier
- Timers and other parameters affecting communication with the peer
- Flow control thresholds

Define the SCT SAP for the SCTP layer after you define the SCTP general configuration parameters. Currently, you can define only one SCT SAP, whose identifier must be 0.

To define an SCT SAP configuration, follow these steps:

Step	Action
1	Call **SctpInitSctSapCfg** to initialize the SCT SAP parameter structure (SCTPSctSapCfg) to default values.
2	Change the default values, as appropriate.
3	Call **SctpSetSctSapCfg** to set the configuration with the specified values.

After the SCT SAP configuration is defined, you can optionally change the values for the parameters in the SCTPSctSapReCfg structure. To change these parameter values, follow these steps:

Step	Action
1	Call **SctpGetSctSapCfg** to obtain the current SCT SAP configuration values.
2	Change the parameter values in the SCTPSctSapReCfg structure as appropriate.
3	Call **SctpSetSctSapCfg** to set the configuration with the specified values.

You must download the board again to change any of the parameter values in the SCTPSctSapCfg structure. For more information, see *SctpInitSctSapCfg* on page 163, *SctpSetSctSapCfg* on page 172, *SctpGetSctSapCfg* on page 160, *SCTPSctSapCfg* on page 187, and *SCTPSctSapReCfg* on page 188.

SCTP TSAP configuration

The transport service access point (TSAP) configuration defines the interface to the layer below SCTP, which is the IP layer. The configurable attributes of a TSAP include the:

- TSAP identifier from the perspective of both SCTP and the lower layer. Currently, these identifiers must be set to 0, and only one TSAP can be defined.
- Maximum number of bind request retries allowed.
- Time interval for which the SCTP layer waits for bind or status confirmations from the lower layer.

Define the TSAP for the SCTP layer right after you define the SCTP SAP configuration parameters. To define a TSAP configuration, follow these steps:

Step	Action
1	Call **SctpInitTSapCfg** to initialize the TSAP parameter structure (SCTPTSapCfg) to default values.
2	Change the default values, as appropriate.
3	Call **SctpSetTSapCfg** to set the configuration with the specified values.

After the TSAP configuration is defined, you can optionally change the values for the parameters in the SCTPTsapReCfg structure. To change these parameter values, follow these steps:

Step	Action
1	Call **SctpGetTSapCfg** to obtain the current SCT SAP configuration values.
2	Change the parameter values in the SCTPTSapReCfg structure as appropriate.
3	Call **SctpSetTSapCfg** to set the configuration with the specified values.

You must download the board again to change any of the parameter values in the SCTPTSapCfg structure. For more information, see *SctpInitTSapCfg* on page 164, *SctpSetTsapCfg* on page 173, *SctpGetTSapCfg* on page 161, *SCTPTSapCfg* on page 191, and *SCTPTSapReCfg* on page 192.

Controlling SCTP entities

Use **SctpMgmtCtrl** to enable the application to control SCTP entities. The following table describes the available control requests by entity:

Entity	Control request
None specified	One of the following general control requests: • Enable or disable alarms. • Start or stop debug logging or trace data. • Start flow control and disable the transmission of all SCTP management messages, except critical messages. • End flow control and enable the transmission of all SCTP management messages. • Shut down the SCTP layer.
SCT SAP	Enable or disable the SCT SAP.
TSAP	Enable or disable the TSAP.

Retrieving SCTP status information

Use the following SCTP status functions to retrieve status information from the SCTP layer:

Entity	Function	Status information returned includes...
SCTP association	**SctpAssocStatus**	Identifier, destination and source network address lists, destination port, and association state for the specified association.
Destination transport addresses	**SctpDestTransAddrStatus**	Destination network address and address state, port, path MTU, and retransmission timeout for the destination transport address within a specific SCTP association.
General SCTP layer	**SctpGenStatus**	Memory size allocated for the SCTP layer. Also contains the number of open associations, open endpoints, local addresses in use, and peer addresses in use.
SCT SAP	**SctpSapStatus**	Protocol in use and state for the specified SCT SAP.

Retrieving SCTP statistics

Use the following SCTP statistics functions to retrieve and optionally reset the following statistics:

Entity	Function	Statistics returned include...
None specified	**SctpGenStatistics**	Statistics counters for incoming and outgoing chunks, total incoming and outgoing bytes, and DNS transactions in the SCTP layer.
SCT SAP	**SctpSctSapStatistics**	Protocol variant and statistics counters for total incoming and outgoing bytes on the SCTP SAP.
TSAP	**SctpTSapStatistics**	Statistics counters for incoming and outgoing chunks, total incoming and outgoing bytes, and number of bind retries on the TSAP.

11 SCTP management functions

SCTP management function summary

NaturalAccess SCTP consists of the following synchronous management functions, in which the action is completed before control is returned to the application:

- Control
- Configuration
- Status
- Statistics

Control functions

Function	Description
SctpMgmtCtrl	Sends a control request to the SCTP layer.
SctpMgmtInit	Initializes internal structures and opens communication with the SCTP process on the TX board. This function must be called before calling any other SCTP management functions.
SctpMgmtTerm	Terminates access to the SCTP management API for this application.

Configuration functions

Function	Description
	Obtains general configuration parameter values from the SCTP layer.
SctpGetSctSapCfg	Obtains SCTP configuration parameter values from the SCTP layer.
SctpGetTSapCfg	Obtains TSAP configuration parameter values from the SCTP layer.
SctpInitGenCfg	Initializes the provided general configuration structure with default values.
SctpInitSctSapCfg	Initializes the provided SCT SAP configuration structure with default values and the SCT SAP identifier.
SctpInitTSapCfg	Initializes the provided TSAP configuration structure with default values and the TSAP identifier.
SctpSetGenCfg	Configures the SCTP layer with the values contained in the general configuration structure.
SctpSetSctSapCfg	Configures the SCTP layer with the values contained in the SCT SAP configuration structure.
SctpSetTSapCfg	Configures the SCTP layer with the values contained in the TSAP configuration structure.

Status functions

Function	Description
SctpAssocStatus	Obtains status information for the specified SCTP association.
SctpDestTransAddrStatus	Obtains status information for all destination transport addresses within a specific SCTP association.
SctpGenStatus	Obtains general status information from the SCTP layer.
SctpSapStatus	Obtains status information for the specified SCT SAP or TSAP from the SCTP layer.

Statistics functions

Function	Description
DateTime	Defines time stamps that indicate when SCTP statistics counters were initialized to zero.
SctpGenStatistics	Obtains and optionally resets the general statistics for the SCTP layer.
SctpSctSapStatistics	Obtains and optionally resets the statistics for the specified SCT SAP.
SctpTsapStatistics	Obtains and optionally resets the statistics for the specified TSAP.

Using the SCTP management function reference

This section provides an alphabetical reference to the SCTP management functions. A typical function includes:

Prototype	The prototype is followed by a list of the function arguments. Dialogic data types include: • U8 (8-bit unsigned) • S16 (16-bit signed) • U16 (16-bit unsigned) • U32 (32-bit unsigned) • Bool (8-bit unsigned)
Return values	The return value for a function is either SCTP_SUCCESS or an error code.

SctpAssocStatus

Obtains status information for the specified SCTP association.

Prototype

SCTP_STATUS TXSCTPFUNC **SctpAssocStatus** (U8 ***board***, SCTPAssocSta ****sta***, U32 ***assocId***)

Argument	Description
board	TX board number on which the desired SCTP layer resides. Valid range is 1 through MAXBOARD (currently 32).
sta	Pointer to the SCTPAssocSta structure where the requested status information is returned. For information, see *SCTPAssocSta* on page 178.
assocId	Identifier of the association for which status is requested.

Return values

Return value	Description
SCTP_SUCCESS	
SCTP_BOARD	Invalid board number.
SCTP_DRIVER	CPI driver error.
SCTP_HANDLE	**SctpMgmtInit** was not called for the specified board.
SCTP_PARAM	Invalid parameter.
SCTP_RESPONSE	Incorrect response from the board.
SCTP_TIMEOUT	No response from the board.

SctpDestTransAddrStatus

Obtains status information for the destination transport addresses within the specified SCTP association.

Prototype

SCTP_STATUS TXSCTPFUNC **SctpDestTransAddrStatus** (U8 ***board***, SCTPDtaSta ****sta***, S16 ***sapId***)

Argument	Description
board	TX board number on which the desired SCTP layer resides. Valid range is 1 through MAXBOARD (currently 32).
sta	Pointer to the SCTPDtaSta structure where the requested status information is returned. For information, see *SCTPDtaSta* on page 180.
sapId	SCTP association identifier.

Return values

Return value	Description
SCTP_SUCCESS	
SCTP_BOARD	Invalid board number.
SCTP_DRIVER	CPI driver error.
SCTP_HANDLE	**SctpMgmtInit** was not called for the specified board.
SCTP_PARAM	Invalid parameter
SCTP_RESPONSE	Incorrect response from the board.
SCTP_TIMEOUT	No response from the board.

SctpGenStatistics

Obtains and optionally resets the general statistics for the SCTP layer.

Prototype

SCTP_STATUS TXSCTPFUNC **SctpGenStatistics** (U8 ***board***, SCTPGenSts ****pStats***, Bool ***isReset***)

Argument	Description
board	TX board number on which the desired SCTP layer resides. Valid range is 1 through MAXBOARD (currently 32).
pStats	Pointer to the SCTPGenSts structure where the requested statistics information is returned. For information, see *SCTPGenSts* on page 185.
isReset	If non-zero, statistics are set to zero after retrieval.

Return values

Return value	Description
SCTP_SUCCESS	
SCTP_BOARD	Invalid board number.
SCTP_DRIVER	CPI driver error.
SCTP_HANDLE	**SctpMgmtInit** was not called for the specified board.
SCTP_PARAM	Invalid parameter.
SCTP_RESPONSE	Incorrect response from the board.
SCTP_TIMEOUT	No response from the board.

SctpGenStatus

Obtains general status information for the SCTP layer, including the size of reserved and allocated memory.

Prototype

SCTP_STATUS TXSCTPFUNC **SctpGenStatus** (U8 ***board***, SCTPGenSta ****sta***)

Argument	Description
board	TX board number on which the desired SCTP layer resides. Valid range is 1 through MAXBOARD (currently 32).
sta	Pointer to the SCTPGenSta structure where the requested status information is returned. For information, see *SCTPGenSta* on page 184.

Return values

Return value	Description
SCTP_SUCCESS	
SCTP_BOARD	Invalid board number.
SCTP_DRIVER	CPI driver error.
SCTP_HANDLE	**SctpMgmtInit** was not called for the specified board.
SCTP_PARAM	Invalid parameter.
SCTP_RESPONSE	Incorrect response from the board.
SCTP_TIMEOUT	No response from the board.

SctpGetGenCfg

Obtains general configuration parameter values from the SCTP layer.

Prototype

SCTP_STATUS TXSCTPFUNC **SctpGetGenCfg** (U8 ***board***, SCTPGenCfg ****pGenCfg***)

Argument	Description
board	TX board number on which the desired SCTP layer resides. Valid range is 1 through MAXBOARD (currently 32).
pGenCfg	Pointer to the SCTPGenCfg structure where the requested configuration information is returned. For information, see *SCTPGenCfg* on page 181.

Return values

Return value	Description
SCTP_SUCCESS	
SCTP_BOARD	Invalid board number.
SCTP_DRIVER	CPI driver error.
SCTP_HANDLE	**SctpMgmtInit** was not called for the specified board.
SCTP_PARAM	Invalid parameter
SCTP_RESPONSE	Incorrect response from the board.
SCTP_TIMEOUT	No response from the board.

See also

SctpInitGenCfg, **SctpSetGenCfg**

SctpGetSctSapCfg

Obtains SCT SAP parameter values from the SCTP layer.

Prototype

SCTP_STATUS TXSCTPFUNC **SctpGetSctSapCfg** (U8 ***board***, SCTPSctSapCfg *****pSctSapCfg***, S16 ***sapid***)

Argument	Description
board	TX board number on which the desired SCTP layer resides. Valid range is 1 through MAXBOARD (currently 32).
pSctSapCfg	Pointer to the SCTPSctSapCfg structure where the requested configuration information is returned. For information, see *SCTPSctSapCfg* on page 187.
sapid	Identifier of the upper SAP for which to obtain information.

Return values

Return value	Description
SCTP_SUCCESS	
SCTP_BOARD	Invalid board number.
SCTP_DRIVER	CPI driver error.
SCTP_HANDLE	**SctpMgmtInit** was not called for the specified board.
SCTP_PARAM	Invalid parameter
SCTP_RESPONSE	Incorrect response from the board.
SCTP_TIMEOUT	No response from the board.

See also

SctpInitSctSapCfg, **SctpSetSctSapCfg**

SctpGetTSapCfg

Obtains TSAP configuration parameter values from the SCTP layer.

Prototype

SCTP_STATUS TXSCTPFUNC **SctpGetTSapCfg** (U8 ***board***, SCTPTSapCfg *****pTSapCfg***, U16 ***sapid***)

Argument	Description
board	TX board number on which the desired SCTP layer resides. Valid range is 1 through MAXBOARD (currently 32).
pTSapCfg	Pointer to the SCTPTSapCfg structure where the requested configuration information is returned. For information, see *SCTPTSapCfg* on page 191.
sapid	Identifier of the lower SAP for which to obtain information.

Return values

Return value	Description
SCTP_SUCCESS	
SCTP_BOARD	Invalid board number.
SCTP_DRIVER	CPI driver error.
SCTP_HANDLE	**SctpMgmtInit** was not called for the specified board.
SCTP_PARAM	Invalid parameter
SCTP_RESPONSE	Incorrect response from the board.
SCTP_TIMEOUT	No response from the board.

See also

SctpInitTSapCfg, **SctpSetTsapCfg**

SctpInitGenCfg

Initializes the SCTP general parameter structures (SCTPGenCfg and SCTPGenReCfg) with default values. SCTPGenReCfg is a substructure to SCTPGenCfg.

Prototype

SCTP_STATUS TXSCTPFUNC **SctpInitGenCfg** (SCTPGenCfg ***pCfg***)

Argument	Description
pCfg	Pointer to the SCTPGenCfg structure to initialize. For information, see *SCTPGenCfg* on page 181.

Return values

Return value	Description
SCTP_SUCCESS	

Details

Call this function prior to calling **SctpSetGenCfg** to define the general configuration parameters for SCTP. You can optionally override specific field values before calling **M3uaSetGenCfg**.

For more information, see *General SCTP configuration* on page 149.

See also

SctpGetGenCfg

SctpInitSctSapCfg

Initializes the SCT SAP configuration structures (SCTPSctSapCfg and SCTPSctSapReCfg) with default values. SCTPSctSapReCfg is a substructure to SCTPSctSapCfg.

Prototype

SCTP_STATUS TXSCTPFUNC **SctpInitSctSapCfg** (SCTPSctSapCfg ***pCfg**, U16 ***id***)

Argument	Description
pCfg	Pointer to the SCTPSctSapCfg structure that contains the SCT SAP configuration values to set. For information, see *SCTPSctSapCfg* on page 187.
id	Identifier of the SCT SAP to initialize. This value is always 0 (zero).

Return values

Return value	Description
SCTP_SUCCESS	

Details

Call this function prior to calling **SctpSetSctSapCfg** to define an SCT SAP. You can optionally override specific field values before calling **M3uaSetSctSapCfg**.

For more information, see *SCTP SCT SAP configuration* on page 150.

See also

SctpGetSctSapCfg

SctpInitTSapCfg

Initializes the TSAP configuration structures (SCTPTSapCfg and SCTPTSapReCfg) with default values. SCTPTSapReCfg is a dependent structure to SCTPTSapCfg.

Prototype

SCTP_STATUS TXSCTPFUNC **SctpInitTSapCfg** (SCTPTSapCfg ****pCfg***, U16 ***id***)

Argument	Description
pCfg	Pointer to the SCTPTSapCfg structure that contains the TSAP configuration values to set. For information, see *SCTPTSapCfg* on page 191.
id	Identifier of the TSAP to initialize. This value is always zero (0).

Return values

Return value	Description
SCTP_SUCCESS	

Details

Call this function prior to calling **SctpSetTsapCfg** to define a TSAP. You can optionally override specific field values before calling **M3uaSetTsapCfg**.

For more information, see *SCTP TSAP configuration* on page 150.

See also

SctpGetTsapCfg

SctpMgmtCtrl

Sends an SCTP control request to the SCTP layer.

Prototype

SCTP_STATUS TXSCTPFUNC **SctpMgmtCtrl** (U8 ***board***, S16 ***entity***, U8 ***action***)

Argument	Description
board	TX board number on which the desired SCTP layer resides. Valid range is 1 through MAXBOARD (currently 32).
entity	Type of control action the function performs. Valid values are: No value = General control SCT SAP identifier - SCT SAP control TSAP identifier = TSAP control **Note:** The entity identifier (***entity***) must have been previously defined with the appropriate set configuration function.
action	Action to take on the specified entity. See the Details section for valid actions.

Return values

Return value	Description
SCTP_SUCCESS	
SCTP_BOARD	Invalid board number.
SCTP_HANDLE	**SctpMgmtInit** was not called for the specified board.
SIGTRAN_ERR_BADACT	Invalid ***action*** argument.

Details

Use **SCTPMgmtCntrl** to activate and deactivate SCTP resources. The combination of ***entity*** and ***action*** tells SCTP what entity to act upon and what action to take. There are three types of control actions:

- General
- SCT SAP
- TSAP

Note: In the following value-description tables, do not use the management API to perform the actions marked with an asterisk, unless you are an experienced user. These actions are performed automatically by the SCTP layer, and are provided through the management API for testing and debugging purposes only.

General control actions

If the ***entity*** argument is not used, the ***action*** argument has the following valid values.

Value	Description
*SCTP_CTRL_ALARM_DIS	Disables alarms.
*SCTP_CTRL_ALARM_ENA	Enables alarms.
SCTP_CTRL_DEBUG_OFF	Stops debug logging.
SCTP_CTRL_DEBUG_ON	Starts debug logging.
*SCTP_CTRL_FLOWCTL_OFF	Ends flow control and enables transmission of all SCTP management messages.
*SCTP_CTRL_FLOWCTL_ON	Starts flow control, and disable transmission of all except critical M3UA management messages.
*SCTP_CTRL_SHUTDOWN	Shuts down the SCTP layer.
SCTP_CTRL_TRACE_OFF	Stops trace data.
SCTP_CTRL_TRACE_ON	Starts trace data.

SCT SAP control actions

If the ***entity*** argument specifies an SCT SAP identifier, the ***action*** argument has the following valid values:

Value	Description
*SCTP_CTRL_SCTSAP_DEL	Deletes the SCT SAP.
*SCTP_CTRL_SCTSAP_DIS	Disables the SCT SAP.
*SCTP_CTRL_SCTSAP_ENA	Enables the SCT SAP.

TSAP control actions

If the ***entity*** argument specifies a TSAP identifier, the ***action*** argument has the following valid values:

Value	Description
*SCTP_CTRL_TSAP_DEL	Deletes the TSAP.
*SCTP_CTRL_TSAP_DIS	Disables the TSAP.
*SCTP_CTRL_TSAP_ENA	Enables the TSAP.

See also

SctpMgmtInit, **SctpMgmtTerm**

SctpMgmtInit

Initializes internal structures and opens communication with the SCTP task on the TX board.

Prototype

SCTP_STATUS TXSCTPFUNC **SctpMgmtInit** (U8 ***board***, U8 ***srcEnt***, U8 ***srcInst***)

Argument	Description
board	TX board number on which the desired SCTP layer resides. Valid range is 1 through MAXBOARD (currently 32).
srcEnt	Source entity ID, which is channel to open for communication with TX board.
srcInst	Source instance ID.

Return values

Return value	Description
SCTP_SUCCESS	
SCTP_BOARD	Invalid board number.
SCTP_DRIVER	CPI driver error.

Details

You must call this function once before calling any other SCTP management functions. The source entity ID must be from 0x20 through 0x7F and unique for each application accessing the SCTP layer through the management API.

Source instance ID should always be 0 for host applications.

See also

SctpMgmtCtrl, **SctpMgmtTerm**

SctpMgmtTerm

Terminates access to the SCTP management API for this application, making the management channel for a specified board available for other uses.

Prototype

SCTP_STATUS TXSCTPFUNC **SctpMgmtTerm** (U8 ***board***)

Argument	Description
board	TX board number on which the SCTP layer resides. Valid range is 1 through MAXBOARD (currently 32).

Return values

Return value	Description
SCTP_SUCCESS	
SCTP_BOARD	Invalid board number.
SCTP_HANDLE	Handle closed or was never opened.

Details

Call this function to free up resources when an application terminates or finishes communication with the SCTP layer.

See also

SctpMgmtCtrl, **SctpMgmtInit**

SctpSctSapStatistics

Retrieves and optionally resets the statistics for the specified SCTP SCT SAP.

Prototype

SCTP_STATUS TXSCTPFUNC **SctpSctSapStatistics** (U8 ***board***, SCTPSctSapSts ****pStats***, S16 ***sapid***, Bool ***isReset***)

Argument	Description
board	TX board number on which the desired SCTP layer resides. Valid range is 1 through MAXBOARD (currently 32).
pStats	Pointer to the SCTPSctSapSts structure where the requested statistics information is returned. For information, see *SCTPSctSapSts* on page 190.
	SCT SAP identifier. This value is always zero (0).
isReset	If non-zero, statistics are set to zero after retrieval.

Return values

Return value	Description
SCTP_SUCCESS	
SCTP_BOARD	Invalid board number.
SCTP_DRIVER	CPI driver error.
SCTP_HANDLE	**SctpMgmtInit** was not called for the specified board.
SCTP_PARAM	Invalid parameter.
SCTP_RESPONSE	Incorrect response from the board.
SCTP_TIMEOUT	No response from the board.

SctpSapStatus

Obtains SCT SAP status information from the SCTP layer, including the SCTP protocol in use and the high level SAP state.

Prototype

SCTP_STATUS TXSCTPFUNC **SctpSapStatus** (U8 ***board***, SCTPSapSta ****sta***, S16 ***sapId***)

Argument	Description
board	TX board number on which the desired SCTP layer resides. Valid range is 1 through MAXBOARD (currently 32).
sta	Pointer to the SCTPSapSta structure where the requested status information is returned. For information, see *SCTPSapSta* on page 186.
sapId	SCT SAP identifier. This value is always zero (0).

Return values

Return value	Description
SCTP_SUCCESS	
SCTP_BOARD	
SCTP_DRIVER	CPI driver error.
SCTP_HANDLE	**SctpMgmtInit** was not called for the specified board.
SCTP_PARAM	
SCTP_RESPONSE	
SCTP_TIMEOUT	

SctpSetGenCfg

Configures the SCTP layer with the values contained in the general configuration structure.

Prototype

SCTP_STATUS TXSCTPFUNC **SctpSetGenCfg** (U8 ***board***, SCTPGenCfg ****pGenCfg***)

Argument	Description
board	TX board number on which the desired SCTP layer resides. Valid range is 1 through MAXBOARD (currently 32).
pGenCfg	Pointer to the SCTPGenCfg structure that contains the values to be configured. For information, see *SCTPGenCfg* on page 181.

Return values

Return value	Description
SCTP_SUCCESS	
SCTP_BOARD	
SCTP_DRIVER	CPI driver error.
SCTP_HANDLE	**SctpMgmtInit** was not called for the specified board.
SCTP_PARAM	Invalid parameter.
SCTP_RESPONSE	Incorrect response from the board.
SCTP_TIMEOUT	No response from the board.

Details

Call this function to configure the SCTP layer after you download the TX board and call **SctpMgmtInit**.

An application must set the field values in the SCTPGenCfg structure before calling **SctpSetGenCfg**. Set the values in any of the following ways:

- Call **SctpInitGenCfg** to set the fields to default values.
- Set each field value from within the application.
- Call **SctpInitGenCfg** and then override specific field values before passing the SCTPGenCfg structure to SctpSetGenCfg.

SctpSetGenCfg is typically called once by an application to set global values.

For more information, see *General SCTP configuration* on page 149.

See also

SctpGetGenCfg

SctpSetSctSapCfg

Configures the SCTP layer with the values contained in the SCT SAP configuration structure.

Prototype

SCTP_STATUS TXSCTPFUNC **SctpSetSctSapCfg** (U8 **board**, SCTPSctSapCfg **pSctSapCfg*)

Argument	Description
board	TX board number on which the desired SCTP layer resides. Valid range is 1 through MAXBOARD (currently 32).
pSctSapCfg	Pointer to the SCTPSctSapCfg structure that contains the values to be configured. For information, see *SCTPSctSapCfg* on page 187.

Return values

Return value	Description
SCTP_SUCCESS	
SCTP_BOARD	Invalid board number.
SCTP_DRIVER	CPI driver error.
SCTP_HANDLE	**SctpMgmtInit** was not called for the specified board.
SCTP_PARAM	Invalid parameter.
SCTP_RESPONSE	Incorrect response from the board.
SCTP_TIMEOUT	No response from the board.

Details

Call this function after you set the general configuration parameters for the SCTP layer, but before you configure a TSAP.

An application must set the field values in the SCTPSctSapCfg structure before calling **SctpSetSctSapCfg**. Set the values in any of the following ways:

- Call **SctpInitSctSapCfg** to set the fields to default values, and then pass the SCTPSapCfg structure.
- Set each field value from within the application.
- Call **SctpInitSctSapCfg** and then override specific field values before passing the SCTPSapCfg structure.

For more information, see *SCTP SCT SAP configuration* on page 150.

See also

SctpGetSctSapCfg

SctpSetTsapCfg

Configures the SCTP layer with the values contained in the TSAP configuration structure.

Prototype

SCTP_STATUS TXSCTPFUNC **SctpSetTSapCfg** (U8 ***board***, SCTPTSapCfg *****pTSapCfg***)

Argument	Description
board	TX board number on which the desired SCTP layer resides. Valid range is 1 through MAXBOARD (currently 32).
pTSapCfg	Pointer to the SCTPTSapCfg structure that contains the values to be configured. For information, see *SCTPTSapCfg* on page 191.

Return values

Return value	Description
SCTP_SUCCESS	
SCTP_BOARD	Invalid board number.
SCTP_DRIVER	CPI driver error.
SCTP_HANDLE	**SctpMgmtInit** was not called for the specified board.
SCTP_PARAM	Invalid parameter.
SCTP_RESPONSE	Incorrect response from the board.
SCTP_TIMEOUT	No response from the board.

Details

Call this function any time after you configure an SCT SAP, but before an application attempts to send data for transaction processing.

An application must set the field values in the SCTPTSapCfg structure before calling **SctpSetTSapCfg**. Set the values in any of the following ways:

- Call **SctpInitTSapCfg** to set the fields to default values, and then pass the SCTPTSapCfg structure.
- Set each field value from within the application.
- Call **SctpInitTSapCfg** and then override specific field values before passing the SCTPTSapCfg structure.

For more information, see *SCTP TSAP configuration* on page 150.

See also

SctpGetTsapCfg

SctpTSapStatistics

Obtains and optionally resets the statistics for the specified SCTP TSAP.

Prototype

SCTP_STATUS TXSCTPFUNC **SctpTSapStatistics** (U8 ***board***, SCTPTSapSts *****pStats***, S16 ***sapid***, Bool ***isReset***)

Argument	Description
board	TX board number on which the desired SCTP layer resides. Valid range is 1 through MAXBOARD (currently 32).
pStats	Pointer to the SCTPTSapSts structure where the requested statistics information is returned. For information, see *SCTPTSapSts* on page 193.
sapid	Lower SAP identifier.
	If non-zero, statistics are set to zero after retrieval.

Return values

Return value	Description
SCTP_SUCCESS	
SCTP_BOARD	Invalid board number.
SCTP_DRIVER	CPI driver error.
SCTP_HANDLE	**SctpMgmtInit** was not called for the specified board.
SCTP_PARAM	Invalid parameter.
SCTP_RESPONSE	Incorrect response from the board.
SCTP_TIMEOUT	No response from the board.

12 SCTP management structures

SCTP management structures summary

This section provides an alphabetical reference to the structures used by SCTP management functions. The topics in the structure reference include the structure definition and a table of field descriptions.

The NaturalAccess™ SIGTRAN implementation task uses the following types of SCTP management structures:

- Configuration
- Statistics
- Status

Configuration structures

The following table describes the SCTP configuration structures in the NaturalAccess™ SIGTRAN implementation:

Structure	Description
SCTPGenCfg	General configuration parameters that cannot be re-configured after setting them with **SctpSetGenCfg**.
	General configuration parameters can be re-configured after setting them with **SctpSetGenCfg**.
SCTPSctSapCfg	SCT SAP configuration fields that cannot be re-configured after setting them with **SctpSetSctSapCfg**.
SCTPSctSapReCfg	SCT SAP configuration fields that can be re-configured after setting them with **SctpSetSctSapCfg**.
SCTPTSapCfg	TSAP configuration fields that cannot be re-configured after setting them with **SctpSetTsapCfg**.
SCTPTSapReCfg	TSAP configuration fields that can be re-configured after setting them with **SctpSetTsapCfg**.

Statistics structures

The following table describes the SCTP statistics structures in the NaturalAccess™ SIGTRAN implementation:

Structure	Description
DateTime	Defines time stamps that indicate when SCTP statistics counters were initialized to zero.
SCTPGenSts	General statistics for the SCTP layer.
	Statistics for the specified SCT SAP.
SCTPTSapSts	Statistics for the specified TSAP.
SCTP statistics sub-structures	Sub-structures to the other statistics structures. These sub-structures hold various types of statistics.

Status structures

The following table describes the SCTP status structures in the NaturalAccess™ SIGTRAN

implementation:

Structure	Description
SCTPAssocSta	Status information for the specified SCTP association.
SCTPDtaSta	Status information for a destination transport address within the specified SCTP association.
SCTPGenSta	General SCTP status information.
	Status information for the SCT SAP.

DateTime (for SCTP)

Dependent function: None.

The following DateTime structure configures timestamps that indicate when statistics counters for SCTP were initialized to zero:

```
typedef struct _DateTime           /* date and time */
{
   U8 month;       /* month */
   U8 day;         /* day */
   U8 year;        /* year - since 1900 (eg. 2000 = 100) */
   U8 hour;        /* hour - 24 hour clock */
   U8 min;         /* minute */
   U8 sec;         /* second */
   U8 tenths;      /* tenths of second */
   U8 fill;
} DateTime;
```

The DateTime structure is a substructure to the SCTPGenSts, SCTPSctSapSts, and SCTPTSapSts structures. It contains the following fields:

Field	Type	Description
month	U8	Month.
day	U8	Day.
year	U8	(Current year) - 1900.
hour	U8	Hour in UTC time (24 hour clock).
min	U8	Minutes.
sec	U8	Seconds.
tenths	U8	Tenths of seconds.
fill	U8	Not used.

SCTPAssocSta

Dependent function: **SctpAssocStatus**

The following SCTPAssocSta structure contains status information for a specific SCTP association:

```
typedef struct _sctpAssocSta     /* Association related solicited status */
{
    U32                 assocId;          /* association identifier */
    U8                  assocState;       /* association state */
    U8                  spare1;           /* alignment */
    U16                 spare2;           /* alignment */
    SCTPNetAddrLst      dstNAddrLst;      /* destination network address list */
    SCTPNetAddrLst      srcNAddrLst;      /* source network address list */
    CmNetAddr           priNAddr;         /* primary network address */
    U16                 dstPort;          /* destination port */
    U16                 srcPort;          /* source port */
} SCTPAssocSta;
```

The SCTPAssocSta structure contains the following fields:

Field	Type	Description
assocId		SCTP association identifier. The value of this field is zero (0) if the status is not available.
assocState	U8	Association state. Valid values are: SCTP_ASSOC_STATE_CLOSED No association is active or open. SCTP_ASSOC_STATE_COOKIE_SENT Association sent a cookie message to the service and is waiting for a cookie acknowledgement message. SCTP_ASSOC_STATE_COOKIE_WAIT Association is waiting for a cookie in an INIT ACK (Initiation Acknowledgement) message from the service user. SCTP_ASSOC_STATE_ESTABLISHED Association is established and ready for two-way communication. SCTP_ASSOC_STATE_OPEN Association is capable of accepting association requests from a remote peer. SCTP_ASSOC_STATE_SDOWNACK_SENT Association sent a SHUTDOWN ACK (Shutdown Acknowledgement) message to the service and is waiting for a SHUTDOWN COMPLETE message. SCTP_ASSOC_STATE_SHUTDOWN_PEND Association received a SHUTDOWN message from the service user. SCTP_ASSOC_STATE_SHUTDOWN_RCVD Association received a SHUTDOWN message from the service user and is waiting for outstanding data to be acknowledged SCTP_ASSOC_STATE_SHUTDOWN_SENT Association sent a SHUTDOWN message to the service and is waiting for a SHUTDOWN ACK (Shutdown Acknowledgement) message.
spare1	U8	Alignment.
spare2	U16	Alignment.
dstNAddrLst	SCTPNetAddrLst	Lists destination addresses that SCTP uses for this association, defined by *SCTPNetAddrLst* on page 194.

Field	Type	Description
srcNAddrLst	SCTPNetAddrLst	Lists source network addresses that SCTP uses for this association, defined by *SCTPNetAddrLst* on page 194.
priNAddr	CmNetAddr	Primary network address for the association, defined by *CmNetAddr* on page 194.
dstPort	U16	Port number for the remote SCTP endpoint in the association.
srcPort	U16	Port number for the local SCTP endpoint in the association.

SCTPDtaSta

Dependent function: **SctpDestTransAddrStatus**

The following SCTPDtaSta structure contains status information for a destination transport address within a specific SCTP association:

```
typedef struct _sctpDtaSta       /* Destination transport address status */
{
    CmNetAddr       dstNAddr;       /* destination network address */
    U16             dstPort;        /* destination port */
    U16             mtu;            /* path MTU */
    U32             assocId;        /* association ID */
    U8              state;          /* destination transport address state */
    U8              spare1;         /* alignment */
    U16             rto;            /* retransmission timeout */
} SCTPDtaSta;
```

The SCTPDtaSta structure contains the following fields:

Field	Type	Description
dstNAddr	CMNetAddr	IP address of the destination transport address, defined by *CmNetAddr* on page 194.
dstPort	U16	Port of the destination transport address.
mtu	U16	Maximum transmission unit for the destination transport address.
assocId	U32	Identifier for the association to which the destination transport address is defined.
state	U8	State of the destination transport address. Valid values are: INACTIVE = Address is unable to send and receive user messages.
spare1	U8	Alignment.
rto	U16	Retransmission timeout value in ms. A re-transmission timeout value of zero (0) means immediate re-transmission.

SCTPGenCfg

Dependent functions: **SctpGetGenCfg**, **SctpInitGenCfg**, **SctpSetGenCfg**

The following SCTPGenCfg structure contains general SCTP configuration parameters. Bold text indicates fields you can set. Unbolded fields are either unused or for internal use only. They are set to correct values by **SctpInitGenCfg** and must not be overridden.

```
typedef struct _sctpGenCfg       /* SCTP General Configuration */
{
    U8          serviceType;     /* TUCL transport protocol (IP/UDP) */
    U16         spare2;          /* alignment */
    U16         maxNmbSctSaps;   /* max no. SCT SAPS */
    U16         maxNmbTSaps;     /* max no. Transport SAPS */
    U16         maxNmbEndp;      /* max no. endpoints */
    U16         maxNmbAssoc;     /* max no. associations */
    U16         maxNmbDstAddr;   /* max no. dest. addresses */
    U16         maxNmbSrcAddr;   /* max no. src. addresses */
    U32         maxNmbTxChunks;  /* max no. outgoing chunks */
    U32         maxNmbRxChunks;  /* max no. recv chunks */
    U16         maxNmbInStrms;   /* max no. in streams PER ASSOCIATION */
    U16         maxNmbOutStrms;  /* max no. out streams PER ASSOCIATION */
    U32         initARwnd;       /* max receiver window space */
    U16         mtuInitial;      /* Initial MTU size */
    U16         mtuMinInitial;   /* Initial minimum MTU size */
    U16         mtuMaxInitial;   /* Initial maximum MTU size */
    Bool        performMtu;      /* Perform path MTU discovery */
    Bool        useHstName;      /* Flag whether hostname is to be used in INIT */
                                 .* and INITACK msg */
    U16         timeRes;         /* timer resolution */
    U16         spare3;          /* alignment */
    U32         debugMask;       /* Debugging mask */
    U32         traceMask;       /* Tracing mask */
    Pst         smPst;           /* layer manager post structure for alarms */
    U8          hostname[CM_DNS_DNAME_LEN]; /* Own Domain Name */
    SCTPGenReCfg  reConfig;       /* reconfigurable params */
} SCTPGenCfg;
```

The SCTPTGenCfg structure contains fields that you can set the first time you call **SctpSetGenCfg**. Once you use **SctpSetGenCfg**, you cannot change the field values in this structure, unless you download the TX board and call **SctpSetGenCfg** again.

The following table describes the fields in the SCTPGenCfg structure that you can set:

Field	Type	Default	Description
maxNmbAssoc	U16	4	Maximum number of SCTP associations the service user can open simultaneously. Valid range is 1 - 65535.
maxNmbDstAddr	U16	8	Maximum number of destination addresses that can be active simultaneously in SCTP. Valid range is 1 - 65535.
maxNmbTxChunks	U32	256	Maximum number of datagrams that can be queued for sending to the peer. Valid range is 1 - the result of (2^32-1).
maxNmbRxChunks	U32	256	Maximum number of datagrams received from the peer that can be queued before being sent up to the service user.
maxNmbInStrms	U16	8	Maximum number of incoming streams per association. Valid range is 1 - 65545.

Field	Type	Default	Description
maxNmbOutStrms	U16	8	Maximum number of outgoing streams per association. Valid range is 1 - 65545.
mtuInitial	U16	1400	Initial path max transmit unit (MTU) in bytes. Valid range is 1 - the result of (2^32-1).
mtuMinInitial	U16	500	Minimum value in bytes when searching for an optimal MTU size using the midpoint algorithm. This field is mandatory if the value of the performMtu field is TRUE. Valid range is 1 - the result of (2^32-1).
mtuMaxInitial	U16	1400	Maximum value in bytes when searching for an optimal MTU size using the midpoint algorithm. This field is mandatory if the value of the performMtu field is TRUE. Valid range is 1 - the result of (2^32-1).
performMtu	Bool	FALSE	Indicates whether or not to perform MTU discovery. Valid values are: TRUE = Perform MTU discovery. FALSE = Do not perform MTU discovery.
reConfig	Structure	Refer to structure.	General parameters that can be re-configured in subsequent calls to **SctpSetGenCfg**. For information, see *SCTPGenReCfg* on page 183.

SCTPGenReCfg

Dependent function: None.

The following SCTPGenReCfg structure contains re-configurable general SCTP configuration parameters. Bold text indicates fields you can modify. The spare1 field is for internal use only and must not be overridden.

```
typedef struct _sctpGenReCfg         /* SCTP General Reconfiguration */
{
   U8        maxInitReTx;       /* Maximum Association Init Retransmits */
   U8        maxAssocReTx;      /* Maximum Retransmissions for an association */
   U8        maxPathReTx;       /* Maximum Retransmission for a destination address */
   Bool      altAcceptFlg;
  /* Accept or don't accept additional life time parameter */
                                /* for init from peer */
   U16       keyTm;             /* Initial value for MD5 key expiry timer */
   U16       alpha;             /* Used in RTT calculations */
   U16       beta;              /* Used in RTT calculations */
   U16       spare1;            /* alignment */
} SCTPGenReCfg;
```

SCTPGenReCfg is a substructure to SCTPGenCfg. Unlike with SCTPGenCfg, you can modify the field values for the bolded fields at any time.

The SCTPGenReCfg structure contains the following fields. Fields not listed in the table are either unused or for internal use only.

Field	Type	Default	Description
maxInitReTx	U8	0	Maximum number of retries for the Initiation (INIT) message to open an association. Valid range is 0 - 255. When maxInitReTx is reached, the association is terminated, and M3UA is notified. M3UA will immediately attempt to reestablish the association with new TSN and InitTag values. INIT retry attempts are rotated through all known IP addresses for the destination. Set to 0 for infinite retries.
maxAssocReTx	U8	10	Maximum number of datagram re-transmissions for an association. The endpoint will be declared unreachable after maxAssocReTx number of consecutive re-transmissions to an endpoint on any transport address. Valid range is 0 to 255.
maxPathReTx	U8	5	Maximum number of datagram re-transmissions to a destination address. The destination address will be declared unreachable after maxPathReTx number of consecutive re-transmissions to a destination address. Valid range is 0 to 255.
altAcceptFlg	Bool	FALSE	If TRUE, accepts an additional parameter from the peer to extended cookie lifetime.
keyTm	U16	60000	Lifetime of an MD5 key. A new private key is generated when this timer expires. Valid range is 1 - 65535.
alpha	U16	12	Used for round trip time (RTT) calculations.
beta	U16	25	Used for RTT calculations.

SCTPGenSta

Dependent function: **SctpGenStatus**

The following SCTP structure contains general SCTP status information:

```
typedef struct _sctpGenSta       /* General Solicited Status */
{
    U8          status;         /* Status */
    U8          spare1;         /* alignment */
    U16         spare2;         /* alignment */
    Size        memSize;        /* Memory Size Reserved (U32) */
    Size        memAlloc;       /* Memory Size Allocated (U32) */
    U16         nmbAssoc;       /* Number of open associations */
    U16         nmbEndp;        /* Number of open endpoints */
    U16         nmbLocalAddr;   /* Number of local addresses in use */
    U16         nmbPeerAddr;    /* Number of peer addresses in use */
1`} SCTPGenSta;
```

The SCTPGenSta structure contains the following fields:

Field	Type	Description
status	U8	Unused.
	U8	Alignment.
spare2	U16	Alignment.
memSize	Size	Total static memory reserved for SCTP, in octets.
memAlloc		Total static memory currently allocated by SCTP, in octets.
nmbAssoc	U16	Number of open SCTP associations.
nmbEndp	U16	Number of open endpoints.
nmbLocalAddr	U16	Number of local addresses used by SCTP.
nmbPeerAddr	U16	

SCTPGenSts

Dependent function: **SctpGenStatistics**

The following SCTPGenSts structure contains general statistics for the SCTP layer:

```
typedef struct _sctpGenSts
{
   DateTime      dt;          /* Date and time when statistics counters are */
                              /* initialized to zero */
   SCTPChunkSts  sbChunkSts;  /* Statistics counters for incoming and outgoing chunks */
   SCTPByteSts   sbByteSts;   /* Statistics counters for total incoming and */
                              /* outgoing bytes */
   SCTPDnsSts    sbDnsSts;    /* Statistics couters for Dns Transations */
} SCTPGenSts;
```

The SCTPTGenSts structure contains the following fields:

Field	Type	Description
dt	Structure	Date and time when the general SCTP statistics were initialized to zero, defined by *DateTime (for SCTP)* on page 177.
sbChunkSts	Structure	Statistics counters for incoming and outgoing chunks, defined by *SCTPChunkSts* on page 195.
sbByteSts	Structure	*SCTPByteSts* on page 195.
sbDnsSts	Structure	Statistics counters for DNS transactions, defined by *SCTPDnsSts* on page 195

SCTPSapSta

Dependent function: **SctpSapStatus**

The following SCTPSapSta structure contains status information for the SCT SAP:

```
typedef struct _sctpSapSta       /* SAP Status */
{
    S16     swtch;              /* Protocol Switch */
    U8      hlSt;               /* SAP State */
    U8      spare1;             /* alignment */
} SCTPSapSta;
```

The SctpSapStatus structure contains the following fields:

Field	Type	Description
swtch	S16	SCTP protocol in use. Only value is LBS_SW_RFC_REL1 (RFC 4460).
hlst	U8	High level upper SAP state. Valid values are: LSB_SAP_BND_ENBL = Bound and enabled.
spare1	U8	Alignment.

SCTPSctSapCfg

Dependent functions: **SctpGetSctSapCfg**, **SctpInitSctSapCfg**, **SctpSetSctSapCfg**

The following SCTPSctSapCfg structure contains SCT SAP configuration parameters for SCTP. Bold text indicates fields you can set. Unbolded fields are either unused or for internal use only. They are set to correct values by **SctpInitSctSapCfg** and must not be overridden.

```
{
    S16             swtch;          /* Protocol Switch */
    S16             spId;           /* Service Provider SAP Id */
    MemoryId        memId;          /* Memory region and pool id for SCTP user */
    U8              sel;            /* Coupling selector for SCTP user */
    U8              prior;          /* Message priority for SCTP user messages */
    U8              route;          /* Route */
    U8              spare1;         /* alignment */
    SCTPSctSapReCfg reConfig;       /* Reconfigurable parameters */
} SCTPSctSapCfg;
```

The SCTPSctSapCfg structure contains fields that you set the first time you call **SctpSetSctSapCfg**. Once you call **SctpSctSetSapCfg**, you cannot change the field values in this structure, unless you re-download the TX board and call **SctpSctSetSapCfg** again.

The following table describes the fields in the SCTPSctSapCfg structure that you can set:

Field	Type	Default	Description
spId	S16	0	SAP identifier used by the SCTP layer. Only value is 0.
reConfig	SCTPSctSapReCfg	Refer to structure.	Re-configurable upper SAP parameters, defined by *SCTPSctSapReCfg* on page 188.

SCTPSctSapReCfg

Dependent function: None.

The following SCTPSctSapReCfg structure contains re-configurable SCT SAP configuration parameters for SCTP. Bold text indicates fields you can modify. Unbolded fields are either unused or for internal use only. They are set to correct values by **SctpInitSctSapCfg** and must not be overridden.

```
typedef struct _SctSapReCfg /* SCT SAP Reconfiguration */
{
    U16             maxAckDelayTm;      /* Maximum time delay for generating Acks */
    U16             maxAckDelayDg;      /* Maximum # of datagrams after which an Ack */
                                        /* shall be sent */
    U16             rtoInitial;         /* Initial value of RTO */
    U16             rtoMin;             /* Minimum RTO */
    U16             rtoMax;             /* Maximum RTO */
    U16             freezeTm;           /* Default Freeze timer value */
    U16             cookieLife;         /* Life time for a Valid Cookie */
    U16             intervalTm;         /* Default Heartbeat interval timer value */
    U16             maxBurst;           /* new protocol parameter defined */
    U16             maxHbBurst;         /* new protocol parameter defined */
    U16             t5SdownGrdTm;       /* T5 Shutdown Guard Timer value */
    U16             spare1;             /* alignment */
    Bool            handleInitFlg;      /* Flag to indicate whether SCTP should
                                         * handle INIT itself */
    Bool            negAbrtFlg;         /* Negotiate or Abort the init if MIS is
                                         * less than OS */
    Bool            hBeatEnable;        /* Enable HeartBeat by Default */
    U8              spare2;             /* alignment */
    U32             flcUpThr;           /* Flow Control upper threshold */
    U32             flcLowThr;          /* Flow Control lower threshold */
    U32             spare3;             /* alignment */
} SCTPSctSapReCfg;
```

SCTPSctSapReCfg is a substructure to SCTPSctSapCfg. Unlike with SCTPSctSapCfg, you can modify the field values for bolded fields at any time.

The following table describes the fields in the SCTPSctSapReCfg structure. Fields not listed in the table are either unused or for internal use only.

Field	Type	Default	Description
maxAckDelayTm	U16	200	Maximum time to wait before the SCTP layer must send a Selective Acknowledgement (SACK) message. Valid range is 1 - 165535.
maxAckDelayDg	U16	2	Maximum number of messages to receive before the SCTP layer must send a SACK message. Valid range is 1 - 165535.
rtoInitial	U16	3000	Initial value of the retransmission timer (RTO). The SCTP layer retransmits data after waiting for feedback during this time period. Valid range is 1 - 65535.
rtoMin	U16	1000	Minimum value used for the RTO. If the computed value of RTO is less than rtoMin, the computed value is rounded up to this value.
rtoMax	U16	10000	Maxiumum value used for RTO. If the computed value of RTO is greater than rtoMax, the computed value is rounded down to this value.
cookieLife	U16	60000	Base cookie lifetime for the cookie in the Initiation Acknowledgement (INIT ACK) message.
intervalTm	U16	3000	Default heartbeat interval timer. Valid range is 1 - 65535.
maxBurst	U16	4	Maximum burst value. Valid range is 1 - 65535.

Field	Type	Default	Description
maxHbBrust	U16	1	Maximum number of heartbeats sent at each retransmission timeout (RTO). Valid range is 1 - 65535.
t5SdownGrdTm	U16	15000	Shutdown guard timer value for graceful shutdowns.
negAbrtFlg	Bool	FALSE	Action to take when the receiver's number of incoming streams is less than the sender's number of outgoing streams. Valid values are: TRUE = Accept incoming stream and continue association. FALSE = Abort the association.
hBeatEnable	Bool	TRUE	Whether to enable or disable heartbeat by default. Valid values are: TRUE = Enable heartbeat. FALSE = Disable heartbeat.
flcUpThr	U32	192	Flow control start threshold. When the number of messages in SCTP’s message queue reaches this value, flow control starts.
flcLowThr	U32	64	Flow control stop threshold. When the number of messages in SCTP’s message queue reaches this value, flow control stops.

SCTPSctSapSts

Dependent function: **SctpSctSapStatistics**

The following SCTPSctSapSts structure contains statistics for the specified SCTP upper SAP:

```
typedef struct _sctpSctSapSts    /* SCTP Statistics for SCTSAP */
{
    S16             swtch;          /* Protocol switch */
    U16             spare1;         /* alignment */
    DateTime        dt;             /* date and time when statistics counters are */
                                    /* initialized to zero */
    SCTPByteSts     sbByteSts;      /* Statistics counters for total incoming and /*
                                    /*outgoing bytes */
} SCTPSctSapSts;
```

The SCTPSctSapSts structure contains the following fields:

Field	Type	Description
swtch	S16	SCTP protocol in use. Only value is LBS_SW_RFC_REL1 (RFC 4460).
spare1	U16	Alignment.
dt	Structure	Date and time when the statistics for this SAP were initialized to zero, defined by *DateTime (for SCTP)* on page 177.
sbByteSts	Structure	Statistics counters for incoming and outgoing bytes, defined by *SCTPByteSts* on page 195.

SCTPTSapCfg

Dependent functions: **SctpGetTsapCfg**, **SctpInitTSapCfg**, **SctpSetTsapCfg**

The following SCTPTSapCfg structure contains TSAP configuration parameters for SCTP. Bold text indicates fields you can set. Unbolded fields are either unused or for internal use only. They are set to correct values by **SctpInitTSapCfg** and must not be overridden.

```
typedef struct _sctpTSapCfg      /* Transport SAP Configuration */
{
    S16             swtch;          /* Protocol Switch */
    S16             suId;           /* Service User SAP Id */
    U8              sel;            /* Coupling selector for SCTP provider */
    U8              ent;            /* Service Provider Entity */
    U8              inst;           /* Service Provider Instance */
    U8              spare1;         /* alignment */
    U16             procId;         /* Service Provider Processor Id */
    U8              prior;          /* Message priority for SCTP provider
                                    /* messages */
    U8              route;          /* Route */
    MemoryId        memId;          /* Memory region and pool id for SCTP provider *
    SCTPNetAddrLst  srcNAddrLst;    /* Source Network Address List */
    SCTPTSapReCfg   reConfig;       /* Reconfigurable parameters */
} SCTPTSapCfg;
```

The SCTPTSapCfg structure contains fields that you set the first time you call **SctpSetTsapCfg**. Once you use **SctpSetTSapCfg**, you cannot change the field values in this structure, unless you re-download the TX board and call **SctpTSapCfg** again.

The following table describes the fields in the SCTPTSapCfg structure that you can set:

Field	Type	Default	Description
suId	S16	0	TSAP identifier.
	Structure	Refer to structure	Re-configurable TSAP configuration parameters, defined by *SCTPTSapReCfg* on page 192.

SCTPTSapReCfg

Dependent function: None.

The following SCTPTSapReCfg structure contains re-configurable TSAP configuration parameters for SCTP. Bold text indicates fields you can set. Unbolded fields are either unused or for internal use only. They are set to correct values by **SctpInitTSapCfg** and must not be overridden.

```
typedef struct _sctpTSapReCfg    /* Transport SAP Reonfiguration */
{
    S16             spId;           /* Service provider SAP ID */
    U8              maxBndRetry;    /* Maximum number of bind retries allowed */
    U8              spare1;         /* alignment */
    U16             tIntTmr;        /* Default time to wait for bind confirm */
    U16             spare2;         /* alignment */
    SCTPDnsCfg      sbDnsCfg;       /* Dns Configuration structure */
} SCTPTSapReCfg;
```

SCTPTSapReCfg is a substructure to SCTPTSapCfg. Unlike with SCTPTSapCfg, You can modify the field values for the bolded fields at any time.

The following table describes the fields in the SCTPTSapReCfg structure. Fields not listed in the table are either unused or for internal use only.

Field	Type	Default	Description
spId	S16	0	Identifier for this TSAP. Only value is 0.
maxBndRetry	U8	3	
tIntTmr	U18	200	Time interval for which the SCTP layer waits for bind or status confirmations from the lower layer.

SCTPTSapSts

Dependent function: **SctpTsapStatistics**

The following SCTPSctSapSts structure contains statistics for the specified SCT SAP:

```
typedef struct _sctpTSapSts      /* SCTP Statistics for TSAP */
{
    S16             swtch;          /* Protocol switch */
    U16             spare1;         /* alignment */
    DateTime        dt;             /* date and time when statistics counters are */
                                    /* initialized to zero */
    SCTPChunkSts    sbChunkSts;     /* Statistics counters for incoming and */
                                    /* outgoing chunks */
    SCTPByteSts     sbByteSts;      /* Statistics counters for total incoming and */
                                    /* outgoing bytes */
    U32             nmbBndRetry;    /* number of bind retries on the SAP */
} SCTPTSapSts;
```

The SCTPTSapSts structure contains the following fields:

Field	Type	Description
swtch	S16	SCTP protocol in use. Only value is LSB_SW_RFC_REL0 (RFC 2960).
spare1	U16	Alignment.
dt	Structure	Date and time when the statistics for this TSAP were initialized to zero, defined by *DateTime (for SCTP)* on page 177.
SbChunkSts	Structure	Statistics counters for incoming and outgoing chunks, defined by *SCTPChunkSts* on page 195.
	Structure	Statistics counters for incoming and outgoing bytes, defined by *SCTPByteSts* on page 195.
nmbBndRetry	U32	Number of blind retries on this TSAP.

SCTP network address substructures

This topic describes the following SCTP network address substructures:

- SCTPNetAddrLst
- CmNetAddr

SCTPNetAddrLst

The following SCTPNetAddrLst defines an array of network addresses. SCTPNetAddrLst is a substructure to the SCTPAssocSta, SCTPSctSapCfg, and SCTPTSapCfg structures.

```
typedef struct _sctptNetAddrLst
{
   U32           nmb;                        /* Number of Network Addresses */
   CmNetAddr     nAddr[SCT_MAX_NET_ADDRS];   /* List of Network Addresses */
} SCTPNetAddrLst;
```

The SCTPNetAddrLst structure contains the following fields:

Field	Type	Description
nmb	U32	Number of network addresses.
nAddr[SCT_MAX_NET_ADDRS]	CmNetAddr	List of network addresses, defined by the CmNetAddr structure, where SCT_MAX_NET_ADDRS is an array of network addresses. For information, see CmNetAddr.

CmNetAddr

The following CmNetAddr structure defines an IPv4 or IPv6 network address. CmNetAddr is a substructure to the SCTPAssocSta, SCTPDtaSta, and SCTPNetAddrLst structures.

```
typedef struct cmNetAddr
{
    U32     type;       /* type of network address */
    union
    {
        Ip4Addr ipv4NetAddr;    /* IP network address (U32) */
        Ip6Addr ipv6NetAddr;    /* IPv6 network address (16 bytes) */
    }u;
} CmNetAddr;
```

The following table describes the fields in the CmNetAddr structure. These fields are not re-configurable.

Field	Type	Description
type	U8	Network address type. Valid values are: CM_NETADDR_IPV4 = IPV4 CM_NETADDR_IPV6 = IPV6 **Note:** IPV6 is not supported in the current release.
ipv4NetAddr	U32	IPv4 network address.
ipv6NetAddr[16]	U8	Not supported in the current release.

SCTP statistics substructures

This topic describes the following SCTP network address substructures:

- SCTPByteSts
- SCTPChunkSts
- SCTPDnsSts

SCTPByteSts

The following SCTPByteSts structure contains counts of bytes sent and received. SCTPByteSts is a substructure to the SCTPGenSts, SCTPSctSapSts, and SCTPTSapSts structures.

```
typedef struct _sctpByteSts   /*  Statistics counters for bytes */
{
   U32       bytesTx;       /* bytes sent */
   U32       bytesRx;       /* bytes received */
} SCTPByteSts;
```

SCTPChunkSts

The following SCTPChunkSts structure contains counts of incoming and outgoing data chunks. SCTPChunkSts is a substructure to the SCTPGenSts, SCTPSctSapSts, and SCTPTSapSts structures.

```
typedef struct _sctpChunkSts     /* Statistics counters for chunks */
{
   U32       noInitTx;       /* number INITs sent */
   U32       noInitReTx;     /* number INITs resent */
   U32       noInitRx;       /* number INITs received */
   U32       noIAckTx;       /* number INIT_ACKs sent */
   U32       noIAckRx;       /* number INIT_ACKs received */
   U32       noShDwnTx;      /* number SHUTDOWNs sent */
   U32       noShDwnReTx;    /* number SHUTDOWNs resent */
   U32       noShDwnRx;      /* number SHUTDOWNs received */
   U32       noShDwnAckTx;   /* number SHUTDOWN_ACKs sent */
   U32       noShDwnAckReTx; /* number SHUTDOWN_ACKSs resent */
   U32       noShDwnAckRx;   /* number SHUTDOWN_ACKs received */
   U32       noCookieTx;     /* number COOKIEs sent */
   U32       noCookieReTx;   /* number COOKIEs resent */
   U32       noCookieRx;     /* number COOKIEs received */
   U32       noCkAckTx;      /* number COOKIE_ACKs sent */
   U32       noCkAckRx;      /* number COOKIE_ACKs received */
   U32       noDataTx;       /* number DATAs sent */
   U32       noDataReTx;     /* number DATAs resent */
   U32       noDataRx;       /* number DATAs received */
   U32       noDAckTx;       /* number SACKs sent */
   U32       noDAckRx;       /* number SACKs received */
   U32       noShDwnCmpltTx; /* number of Shutdown completed sent */
   U32       noShDwnCmpltRx; /* number of Shutdown completed sent */
} SCTPChunkSts;
```

SCTPDnsSts

The following SCTPDnsSts structure contains counts of DNS queries sent, resent, and received. SCTPDnsSts is a substructure to SCTPGenSts.

```
typedef struct _sctpDnsSts
{
   U32       noQueryTx;          /* Number of DNS Queries Txmitted */
   U32       noQueryReTx;        /* Number of DNS Queries ReTxmitted */
   U32       noQueryRspRx;       /* Number of DNS Query Responses Received */
} SCTPDnsSts;
```

13 sctpmgr utility

sctpmgr overview

sctpmgr is an SCTP utility for managing and troubleshooting the SCTP layer. Use *sctpmgr* to:

- Enable or disable SCT SAPs and TSAPs
- Display statistics for the SCTP layer and SCTP entities
- Display general status information and configuration values for the SCTP layer and SCTP entities

Complete the following steps to use *sctpmgr*:

Step	Action

1	Start *sctpmgr* by entering:
	`sctpmgr`
	The following information displays:
	`sctpmgr: sample SCTP management application version 1.0 Jun 12 2008`
	`sctpmgr[1]>`

2	Enter *sctpmgr* commands at the command prompt. For information, see *sctpmgr commands* on page 198.
3	End an *sctpmgr* session by typing **q** and pressing **Enter**.

sctpmgr commands

The following table describes the available *sctpmgr* commands. The commands and arguments can be in upper, lower, or mixed case.

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td></td><td>Displays the usage of all supported sctpmgr commands.
<code>?</code></td></tr>
<tr><td>board</td><td>Switches the sctpmgr communication to the board specified by boardNum.
<code>board boardNum</code></td></tr>
<tr><td>debug</td><td>Enables or disables debug logging for the SCTP layer.
<code>debug operation</code>
where operation is one of the following arguments:
<table>
<tr><th>Argument</th><th>Valid values</th></tr>
<tr><td>ena</td><td>Enables debug logging.</td></tr>
<tr><td>dis</td><td>Disables debug logging.</td></tr>
</table></td></tr>
<tr><td>stats</td><td>Displays and optionally resets current statistics for the SCTP layer.
<code>stats entity [reset]</code>
where entity is one of the following arguments:
<table>
<tr><th>Argument</th><th></th></tr>
<tr><td>sctp</td><td>Displays data chunk counters for the SCTP layer.</td></tr>
<tr><td>sctsap sapId</td><td>Displays transmitted, received, and re-transmitted message statistics for the SCT SAP specified by sapid.</td></tr>
<tr><td>tsap sapId</td><td>for the TSAP specified by sapId.</td></tr>
</table>
If specified, reset sets all statistics for the specified object to zero (0) immediately after the current statistics are displayed.</td></tr>
<tr><td>status</td><td>Displays status information and configuration values for the SCTP layer and its entities.
<code>status entity</code>
where entity is one of the following arguments:
<table>
<tr><th>Argument</th><th>Description</th></tr>
<tr><td>assoc assocId</td><td>Displays status information and configuration values for the association specified by assocId.</td></tr>
<tr><td>sap sapId</td><td>Displays status information and configuration values for the SCT SAP or TSAP specified by sapId.</td></tr>
<tr><td>sctp</td><td>Displays general status information and configuration values for the SCTP layer.</td></tr>
</table>
For more information, see sctpmgr status command examples on page 201.</td></tr>
</table>

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td>trace</td><td>Enables or disables data tracing for the SCTP layer.
<code>trace operation</code>
where operation is one of the following arguments:
<table><tr><th>Argument</th><th>Valid values</th></tr><tr><td>ena</td><td>Enables data tracing.</td></tr><tr><td>dis</td><td>Disables data tracing.</td></tr></table></td></tr>
<tr><td>tsap</td><td>Enables or disables the specified TSAP. For example:
<code>TSAP operation sapId</code>
where operation is one of the following arguments:
<table><tr><th>Argument</th><th></th></tr><tr><td>ena</td><td>Enables TSAP sapId.</td></tr><tr><td>dis</td><td>Disables TSAP sapId.</td></tr></table></td></tr>
<tr><td>q</td><td>Quits the sctpmgr application.</td></tr>
</table>

sctpmgr statistics command examples

This topic provides examples of the following *sctpmgr* stats commands:

- stats sctp
- stats sctsap
- stats tsap

stats sctp

The following example displays statistics for the SCTP layer:

```
sctpmgr[1]>stats sctp
============SCTP Statistics Since 8-15-2008 16:43:11=============
Statstics for Rx & Tx Chunks
        Counter     Tx              Rx              Re-Tx
================================================================
        INIT        2               0               2
        INIT ACK    0               2               n/a
        SHUTDOWN    0               0               0
        DOWN ACK    0               0               0
        COOKIES     2               0               0
        CK ACKS     0               2               n/a
        DATA        11              51285           0
        DAT ACKS    51279           9               n/a
        DWN COMPL   0               0               n/a
        BYTES       7793068         336             n/a
        DNS QRY     0               0               0
```

stats sctsap

The following example displays statistics for SCT SAP 0:

```
sctpmgr[1]>stats sctsap 0
=============SCTSAP 0 Statistic Since 8-15-2008 16:43:11==============
        Total Bytes Received:   336
        Total Bytes Transmit:   7801580
```

stats tsap

The following example displays statistics for TSAP 0:

```
sctpmgr[1]>stats tsap 0
==============TSAP 0 Statistic Since 8-15-2008 16:43:11==============
        Counter     Tx              Rx              Re-Tx
================================================================
        INIT        2               0               2
        INIT ACK    0               2               n/a
        SHUTDOWN    0               0               0
        DOWN ACK    0               0               0
        COOKIES     2               0               0
        CK ACKS     0               2               n/a
        DATA        11              51365           0
        DAT ACKS    51359           9               n/a
        DWN COMPL   0               0               n/a
        BYTES       0               0               n/a
        BIND RETRY  n/a             n/a             0
```

sctpmgr status command examples

This topic provides examples of the following *sctpmgr* status commands:

- status assoc
- status sap
- status sctp

status assoc

The following example displays status information and configuration values for association 0:

```
sctpmgr[1]>status assoc 0
|         Association Status for 0
|         State:    ESTABLISHED
|         Destination Address List
|         (1) 10.51.1.101
|         Source Address List
|         (1) 10.51.1.100
|         Primary Network Address: 10.51.1.101
|         Source Port:             2905
|         Destination Port:        2905
```

status sap

The following example displays status information and configuration values for the SCTP SAP and TSAP in the SCTP layer:

```
sctpmgr[1]>status sap 0
Protocol Switch :1
SAP Status      :BND
====================Current SCT SAP Configurations====================
swtch         = 1       spId          = 0       selector      = 1
MemRegion     = 0       Mempool       = 0       prior         = 0
route         = 0       maxAckDelayTm = 200     maxAckDelayDg = 2
rtoInitial    = 3000    rtoMin        = 1000    rtoMax        = 10000
freezeTm      = 3000    cookieLife    = 60000   intervalTm    = 3000
maxBurst      = 4       maxHbBurst    = 1       t5SdownGrdTm  = 15000
handleInitFlg = FALSE   negAbrtFlg    = FALSE   hBeatEnable   = TRUE
flcUpThr      = 192     flcLowThr     = 64
====================Current TSAP Configurations====================
swtch         = 1       suId          = 0       sel           = 1
ent           = ENTHI   inst          = 0       procId        = 0
MemPool       = 0       prior         = 0       route         = 0
SrcAddrType   = 4       srcAddress    = a330166 srcNAdrNmb    = 1
spId          = 0       maxBndRetry   = 3       BndCfmTmr     = 200
```

status sctp

The following example displays general status information and configuration values for the SCTP layer:

```
sctpmgr[1]>status sctp
Status            = 0
Mem Reserved      = 0
Mem Allocated     = 0
Associations    = 1
Local Addr in Use = 1
Peer Addr in Use  = 1
=====================Current SCTP Configurations=====================
serviceType    = TCP     maxNmbSctSaps  = 1       maxNmbTSaps    = 1
maxNmbEndp     = 1       maxNmbAssoc    = 4       maxNmbDstAddr  = 8
maxNmbSrcAddr  = 3       maxNmbTxChunks = 256     maxNmbRxChunks = 256
maxNmbInStrms  = 8       maxNmbOutStrms = 8       initARwnd      = 32767
mtuInitial     = 1400    mtuMinInitial  = 500     mtuMaxInitial  = 1400
performMtu     = FALSE   UseHstName     = FALSE   Hostname       =
timeRes        = 10      maxInitReTx    = 5       maxAssocReTx   = 5
maxPathReTx    = 5       altAcceptFlg   = 0       keyTm          = 60000
alpha          = 12      beta           = 25
```

Index

N