Dialogic
Making Innovation Thrive™

Dialogic® NaturalAccess™ MTP3 Layer Developer's Reference Manual

July 2009 64-0460-01

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
9000-6465-11	September, 1997	GJG
9000-6465-12	July, 1998	GJG
9000-6465-13	September, 1998	GJG
9000-6465-14	March, 1999	GJG
9000-6465-15	May, 1999	GJG
9000-6465-16	April, 2000	GJG; SS7 3.5 Beta
9000-6465-17	November, 2000	GJG; SS7 3.6
9000-6465-18	August, 2001	GJG; SS7 3.8 Beta
9000-6465-19	February, 2002	MVH; SS7 3.8
9000-6465-20	November, 2003	MCM; SS7 4.0
9000-6465-21	April, 2004	CYF; SS7 4.0 GA
9000-6465-22	April, 2005	LBG; SS7 4.2 GA
9000-6465-23	July 2006	LBZ, SS7 4.3 GA
9000-6465-24	September 2008	LBG, SS7 5.0
64-0460-01	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™ MTP3 Layer Developer's Reference Manual* explains how to implement SS7 MTP 3 layer using Dialogic® NaturalAccess™ MTP3. This manual explains how to create applications using NMS MTP 3 and presents a detailed specification of its signaling procedures and functions.

Note: The product(s) to which this document pertains is/are among those sold by NMS Communications Corporation ("NMS") to Dialogic Corporation ("Dialogic") in December 2008. Certain terminology relating to the product(s) has been changed, whereas other terminology has been retained for consistency and ease of reference. For the changed terminology relating to the product(s), below is a table indicating the "New Terminology" and the "Former Terminology". The respective terminologies can be equated to each other to the extent that either/both appear within this document.

Former terminology	Current terminology
NMS SS7	Dialogic® NaturalAccess™ Signaling Software
Natural Access	Dialogic® NaturalAccess™ Software
NMS MTP 3	Dialogic® NaturalAccess™ MTP3 Layer

2 NMS MTP 3 overview

NMS MTP 3 components

A typical NMS MTP 3 implementation consists of TX board and host processor components. The TX board components include the:

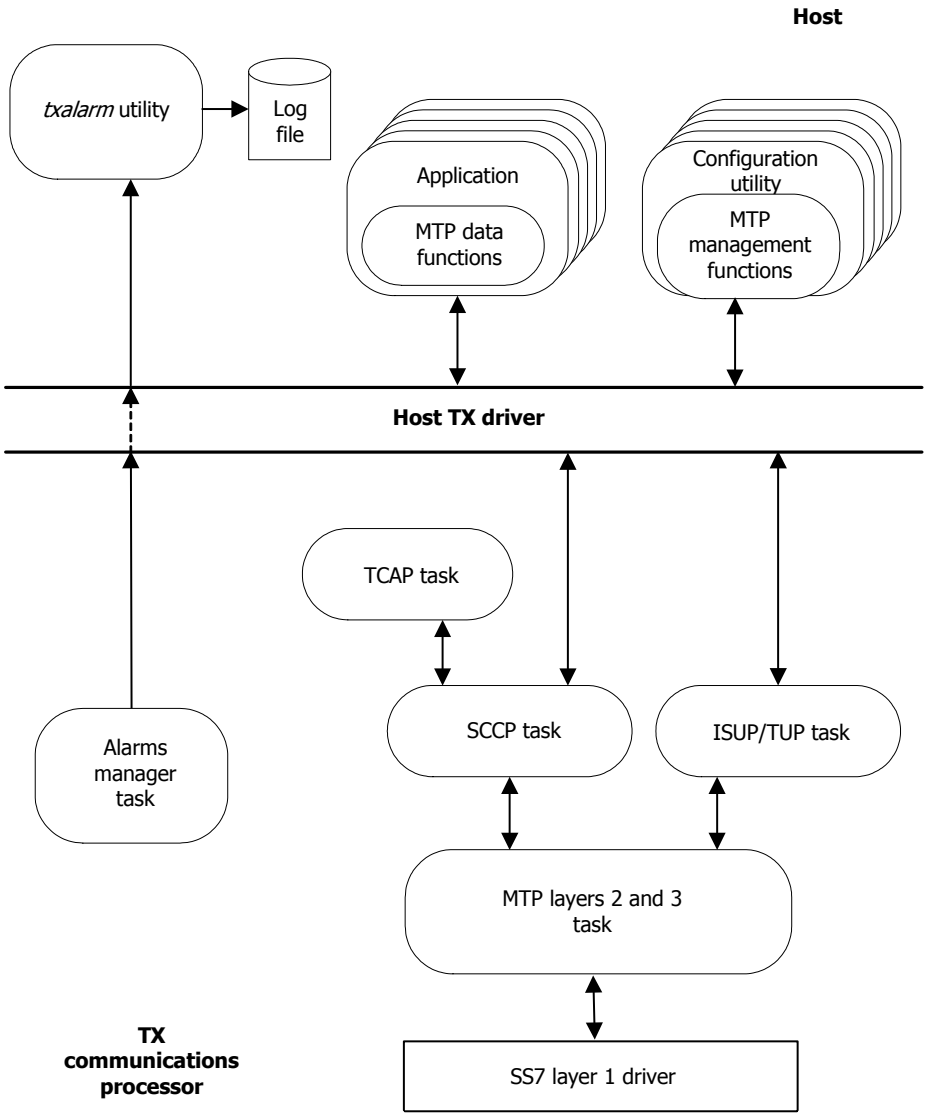
- MTP 3 task that implements the MTP 2 (data link) layer and the MTP 3 (network) layer.
- TX alarms manager task that collects unsolicited alarms (status changes) generated by the SS7 tasks and forwards them to the host for application-specific alarm processing.
- CPK/OS operating system.

The host processor components include:

- The operating system-independent TX device driver that provides low-level access to the TX board from the host computer.
- The MTP 3 configuration file that describes the general configuration of the MTP layer 3 process, the SS7 links and link sets available to it, and the routes to other SS7 signal points.
- An MTP 3 configuration program that reads the MTP 3 configuration file and loads the configuration to the MTP task at system startup.
- MTP data and management functions.
- Sample applications for the data and the management functions, provided in source code and object code formats.
- A log process for capturing alarms from the board.

MTP data and management functions pass messages between the application and the MTP task on the TX board, transferring data, controlling flow and communications in the SS7 network, and obtaining status and statistical information. The functions simplify data and management operations and perform the byte ordering translation, where necessary, between application processor (Intel or little endian) byte order and network (big endian) byte order.

The following illustration shows NMS SS7 architecture:



MTP 3 task

The MTP 3 task has two primary functions:

Function	Description
Message routing and distribution	Routes outgoing messages to their specified destinations and distributes incoming messages to the appropriate application. NMS MTP 3 uses a flexible configuration capable of supporting a wide variety of network routing and addressing requirements.
Signaling network management	Reconfigures the signaling network as needed to maintain signaling capability in the case of failure or congestion. NMS MTP 3 redirects traffic away from failed links, signaling points (SPs), or both. It restores traffic to restored links and SPs, and exchanges route status with adjacent SPs.

MTP 3 entities

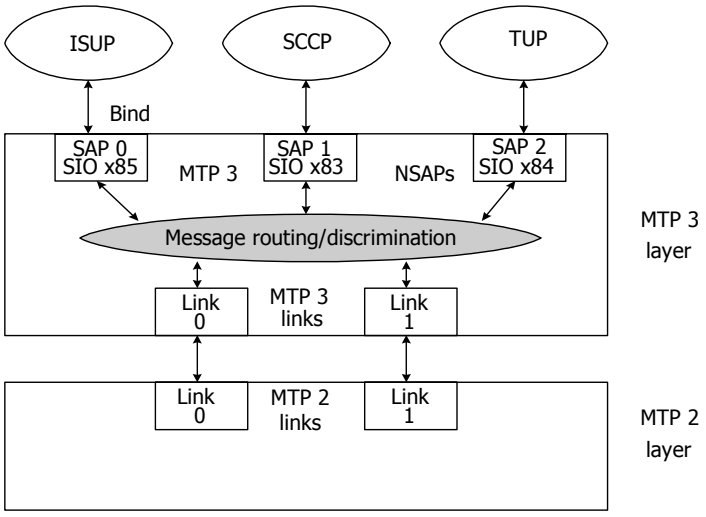
The MTP 3 layer consists of the following configurable entities:

- Service access points (SAPs)
- Signaling links
- Linksets
- Routes

Service access points (SAPs)

Service access points (SAPs) define the SS7 applications that use MTP 3. Each SAP is associated with one application, as identified by the service indicator field of a message, and one protocol variant.

The following illustration shows service access points (SAPs):



If multiple protocol variants are configured on the same MTP 3 instance (same board), two SAPs are required for each application: one for each protocol variant. In this case, a single application can associate itself with both SAPs for that service, or separate applications can be used for each protocol variant.

Signaling links

Signaling links define physical links between the TX board and the adjacent signaling points. One link configuration must be performed for each physical signaling link. The attributes of a link include the point code of the adjacent signaling point, protocol variant employed on the link, point code length, maximum packet length, various timer values, and membership in a linkset.

High speed links (HSL) meet the ANSI T1.111-1996 and Q.703/Annex A standards. Each HSL occupies a full (unchannelized) T1/E1 line and transfers data at the rate of 2.0 (1.544) Mbps. For information about configuring high speed links, refer to *Mtp3InitLinkCfg* on page 71.

Linksets

Linksets are groups of one to 16 links that directly connect two signaling points. Although a linkset usually contains all parallel signaling links between two SPs, you can define parallel linksets. Each signaling link is assigned membership in one linkset.

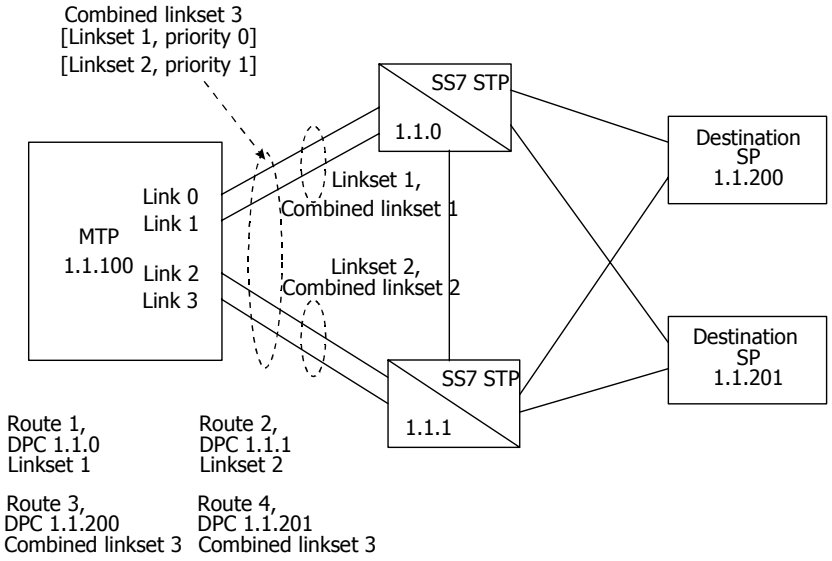
A combined linkset is a group of all linksets that can be used to reach a particular destination or group of destinations (routes). Each linkset can be associated with up to 16 combined linksets. Each linkset in a combined linkset can be assigned a priority relative to the other linksets belonging to that combined linkset.

Routes

Routes specify the destination signaling points (subnetworks or clusters) accessible from the target node. Each route is assigned a direction: up or down. One up route is required for the point code assigned to the signaling point being configured, and for each point code that is to be emulated. Up routes are used to identify incoming messages that are to be routed up to the applications or user parts. One down route is required for each remote signaling point, network, or cluster that is to be accessible from the SP being configured.

Down routes are used to route outgoing messages to the appropriate signaling links. Each down route is assigned to one combined linkset, which in turn identifies all linksets that can be used to reach that destination. Each linkset within the route's associated combined linkset can be assigned an optional priority, so that MTP routing chooses the highest priority available linkset when routing an outgoing packet to a particular destination. Any number of routes can be assigned to a combined linkset.

The following illustration shows the relationship between links, linksets, combined linksets, and routes:



3 NMS MTP 3 programming model

Signaling point configurations

The MTP 3 layer can be configured as a signal transfer point (STP) or as a signaling endpoint (SP). The primary difference between STP operation and SP operation is the handling of messages received from signaling links by the MTP 3 layer but addressed to other destinations.

When configured as an STP, MTP 3 searches for an outbound route to the message destination, and if found, routes the message over an outbound link. When configured as an SP, the MTP 3 layer discards such messages.

When configured as an STP, the MTP 3 layer also performs the additional signaling route management procedures required of an STP. These procedures involve notifying adjacent SPs when they can no longer route messages to a particular destination through that STP due to failures or congestion (transfer prohibited or restricted), and notifying them again when normal communication with the concerned destination is restored (transfer allowed).

MTP 3 message handling

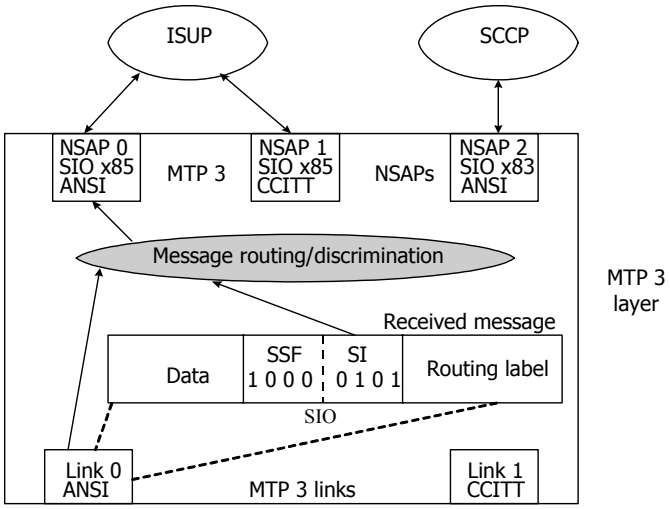
This topic describes:

- Inbound messages
- Outbound messages
- Routing masks

Inbound messages

When an inbound message is received, MTP 3 searches its routing tables for a route that matches the destination point code in the message. If a matching up route is found, the message is distributed to the appropriate user part or application based on the service indicator value in the incoming message and the link type (protocol variant) attribute of the link on which the message was received.

The following illustration shows inbound message distribution:



Only the link type of the incoming link and the service indicator portion of the SIO (service information octet) in the incoming message are used to choose the NSAP to receive the incoming message; the sub-service field of the SIO octet (national/international indicator and optional message priority) is not used.

If a matching down route is found for the message, the message is routed as an outbound message (if MTP 3 is configured as an STP) or the message is discarded. If no matching route is found, the message is always discarded.

Outbound messages

Message routing for outbound messages originated by local user parts or applications, or received messages being routed outbound when in STP mode, is based on the destination point code (DPC) and signaling link selection (SLS) field associated with the message. If opcRouting is TRUE, the origination point code (OPC) is also used for outbound routing.

When an outbound message is ready for routing, MTP 3 searches its routing tables for a route that matches the DPC (and optionally the OPC) specified for the message. If a matching down route is found for the message, message routing depends on the value specified for the SLS field, as follows:

- If MTP 3 recently routed a message with the same combination of DPC and SLS value, the same linkset and link are chosen for transmission to maintain the order of delivery for messages between the same user parts or applications in the same direction with the same SLS value. This combination of DPC/SLS value is called a route instance. A route instance remains in effect until a configurable amount of time elapses with no new messages containing the same DPC/SLS values.
- If no current route instance is in effect, MTP 3 finds all active linksets belonging to the combined linkset associated with the target route and chooses the highest priority for transmission. If multiple linksets of the same priority are active, the linkset is chosen on a round-robin basis from all those at the same (highest) priority. Within the linkset, the actual link to transmit the message on is also chosen on a priority basis, again round-robin when multiple links of the same priority are available.

The application is responsible for assigning SLS values for outgoing messages. Related messages must be assigned the same SLS value in order to be sent over the same link and therefore be guaranteed to arrive in the correct order at the other end. For unrelated messages, the SLS value is varied to achieve the desired degree of load sharing. For example, the ISDN user part (ISUP) typically uses the least significant 4 bits of its circuit identification code (CIC) as the SLS value. Therefore, all messages related to a particular circuit use the same link/linkset. Messages for different circuits are load shared across all available links/linksets.

Routing masks

MTP 3 uses routing masks to help decrease the size of the routing tables that must be configured. Routing masks are bit masks that specify a subset of a destination point code to be matched against the routing table when searching for a route for an inbound or an outbound message.

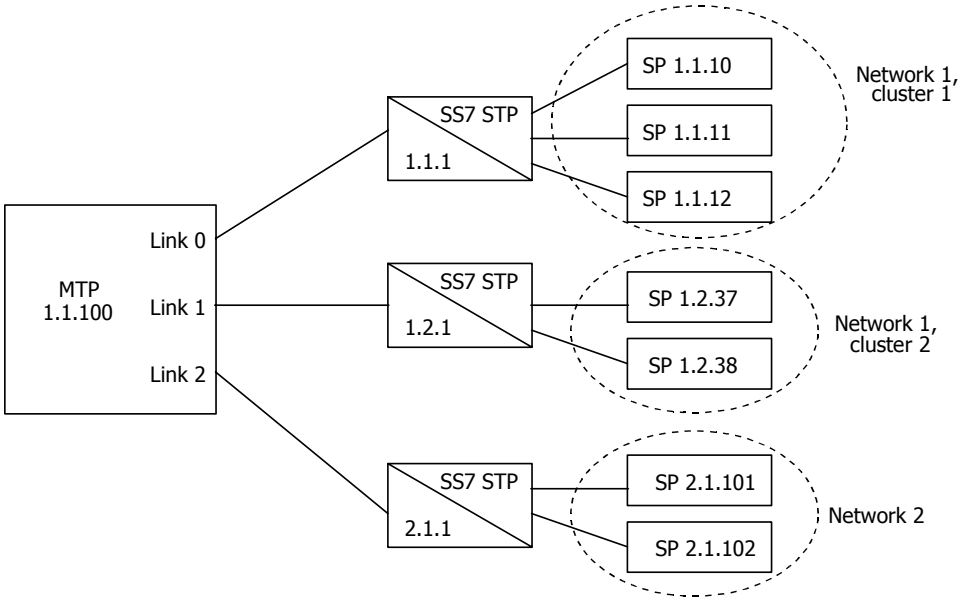
Use routing masks to implement network and cluster routing in ANSI networks. In the following example, rather than specifying explicit routes to each of the seven remote SPs, routing masks and routes are used. All point codes and routing masks, regardless of point code length, are stored internally as 32-bit unsigned integers.

Routing mask	Details
0xFFFFFFFF	Always specify exact match as first mask.
0xFFFF00	Match on network ID and cluster ID next.
0xFF0000	Match on just network ID last.

Routes	Details
1.1.100, UP	Always specify an up route to self.
1.1.1, DOWN, linkset 1	Explicit route to STP for network 1, cluster 1.
1.1.0, DOWN, linkset 1	Cluster route to network 1, cluster 1.
1.2.1, DOWN, linkset 2	Explicit route to STP for network 1, cluster 2.
1.2.0, DOWN, linkset 2	Cluster route to network 1, cluster 2.
2.1.1, DOWN, linkset 3	Explicit route to STP for network 2.
2.0.0, DOWN, linkset 3	Network route to network 2.

Routing masks are global to all links, linksets, and user parts, and are applied to both incoming and outgoing messages. Although the previous example is specific to ANSI networks, routing masks can be applied equally to other networks (ITU-T based networks with 14- or 24-bit point codes) to reduce the size of routing tables.

The following illustration shows network and cluster routing:



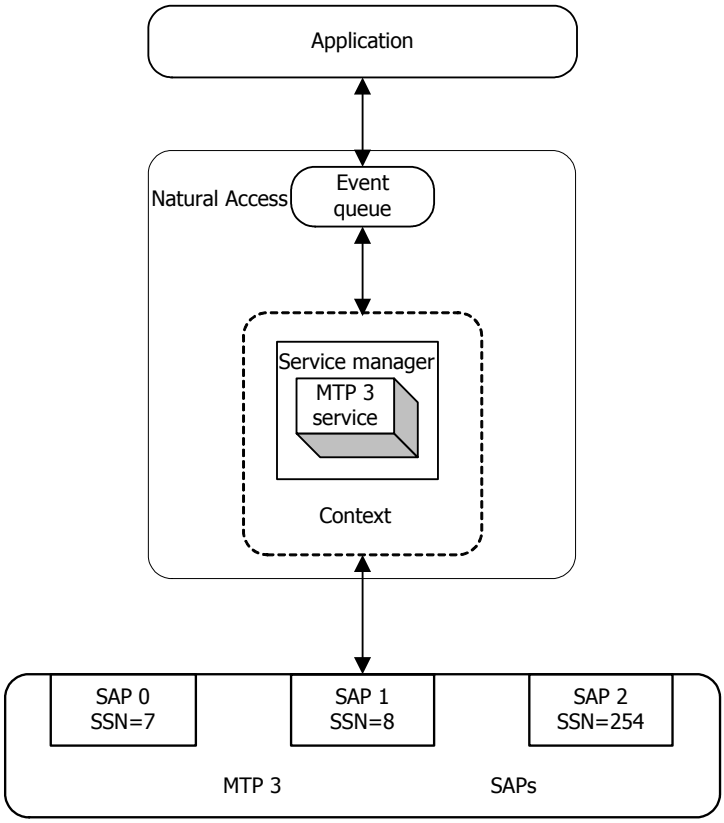
Contexts and queues

Natural Access organizes services and their associated resources around a processing object known as a context. Each instance of an application binding to an MTP 3 service access point (SAP) is a unique Natural Access context. Contexts are created with **ctaCreateContext**.

A Natural Access queue delivers all events and messages from the MTP 3 service 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.

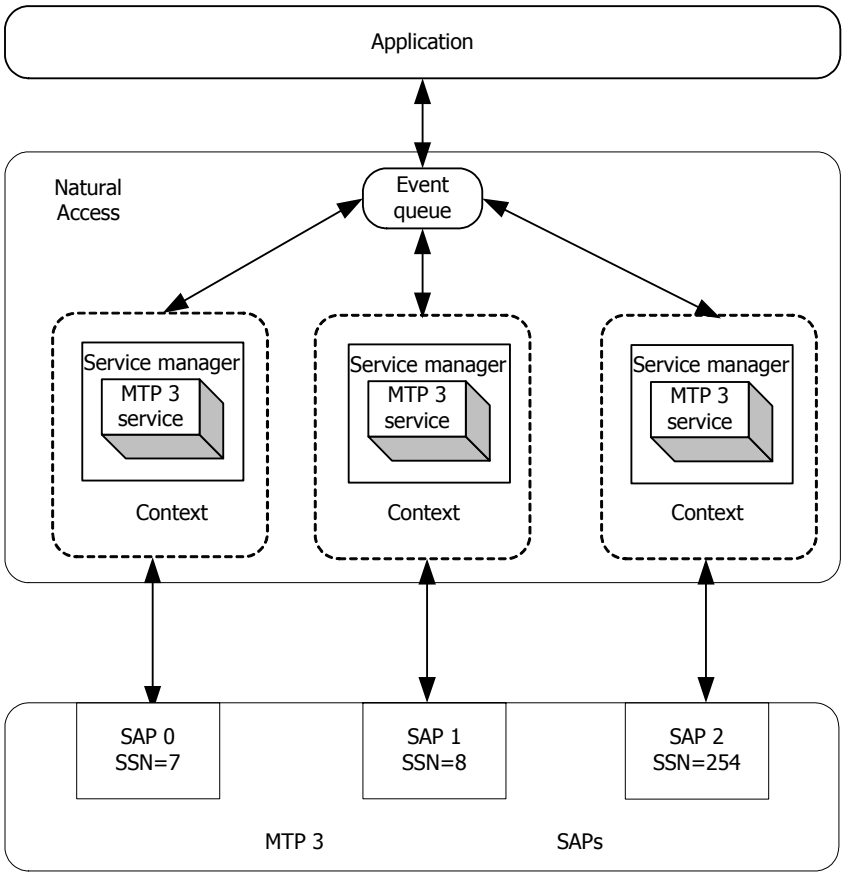
Different application programming models are possible depending on how many MTP 3 SAPs (subsystems) are implemented by the application and how the application is organized.

An application that uses a single MTP 3 SAP uses a single context, single queue model, as shown in the following illustration:



For a single threaded application that uses multiple MTP 3 SAPs (multiple MTP 3 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.

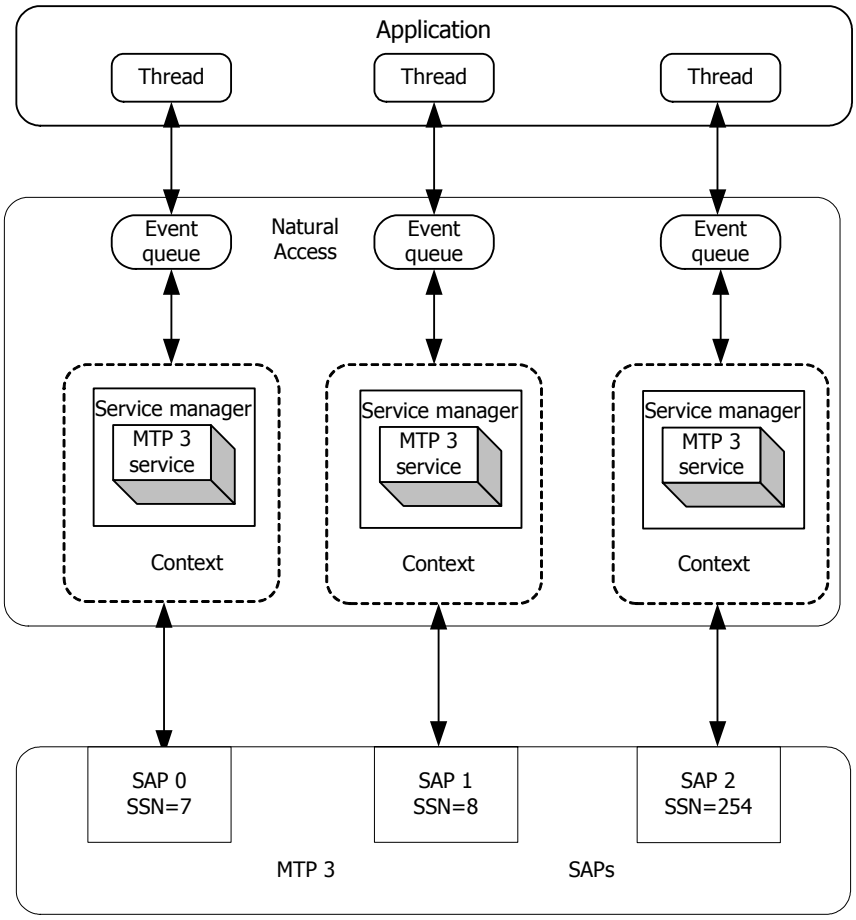
The following illustration shows a multiple context, single queue model:



For multi-threaded applications using multiple MTP 3 SAPs (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 MTP 3 SAP 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:



Signaling network management

Signaling network management provides the functions necessary to maintain signaling service during failures and congestion, and restore normal signaling service after recovery from these conditions. MTP 3 supports all required ANSI and ITU-T network management procedures without intervention from the user parts or applications. MTP 3 notifies applications of significant network events that might impact their operation, such as changes in the accessibility status of remote signaling points, the onset and abatement of congestion, and signaling point restarts.

This topic discusses:

- Changeover and changeback
- Forced and controlled rerouting
- MTP restart
- Signaling link management
- Signaling link test
- Signaling route management

Changeover and changeback

MTP 3 automatically initiates changeover procedures whenever a link fails, is deactivated, is remotely blocked, or is inhibited. No user part or application action is required. MTP 3 does not notify the user part or application when the changeover occurs.

When a signaling link is restored, unblocked, or uninhibited, MTP 3 automatically performs the changeback function without interacting with the user parts or applications.

Forced and controlled rerouting

MTP 3 also handles forced and controlled rerouting upon receipt of the transfer prohibited and transfer allowed or transfer restricted messages. On receipt of a transfer prohibited (TFP) message, MTP 3 attempts to redirect all traffic for the prohibited destination to an alternate route. If no alternate routes are available, the destination is declared inaccessible and each user part or application is notified with a StatPaused status indication for the concerned destination. Destinations can also be declared inaccessible for other reasons such as signaling link or signaling point failures, which result in similar StatPaused indications to the user parts.

Traffic routing over an unavailable or restricted route is automatically restored upon receipt of the transfer allowed (TFA) or transfer restricted (TFR) message for that route. If the TFA/TFR make a previously inaccessible destination accessible, each user part is notified with a StatResumed indication for that destination.

MTP restart

MTP 3 can be configured with or without the MTP restart capability. The MTP restart function allows a signaling point just becoming available (such as after a failure) to bring up sufficient links to handle the expected traffic load before receiving new traffic.

If configured to do so, MTP 3 performs the restart function when the first signaling link becomes active (such as at system startup or after a total failure affecting all links), or on command from a management primitive. At the beginning of an MTP restart, each user part or application is notified with a StatRestart indication. Any new traffic requests generated by user parts during the restart are discarded. When the restart is complete and the MTP 3 layer is ready for traffic, each user part or application receives a StatRestartEnds indication.

Signaling link management

MTP 3 provides the basic link management functions and optionally the signaling link management procedures based on automatic allocation of signaling terminals described in the ANSI and ITU-T MTP standards.

Each linkset has a minNmbActLnk attribute that determines the normal number of active links in the linkset. Typically this number includes all links in the linkset, but you can configure extra alternate links that are activated only in the presence of failures of other links.

When a linkset is activated (such as at system startup time), MTP 3 attempts to activate the minNmbActLnk highest priority links in that linkset. If a link fails or cannot be aligned successfully, MTP 3 periodically attempts to restore the link until successful or until the link is manually disabled through a management function.

If the current number of active links in a linkset drops below the minNmbActLnk threshold, MTP 3 attempts to activate the highest priority inactive link not currently inhibited, remotely blocked, or manually disabled. If no other links in the linkset meet this criteria, MTP 3 attempts to uninhibit the highest priority inhibited link in the linkset.

If the current number of active links in a linkset goes above the minNmbActLnk threshold due to a link restoration, MTP 3 attempts to deactivate the lowest priority active link to return the number of active links in the linkset to its normal condition.

Automatic activation or deactivation of links is normally performed without interacting with the user parts or applications, unless it results in one or more destinations becoming accessible or inaccessible. In this case, the user parts are notified with a StatResumed or StatPaused indication.

If no automatic activation or deactivation of links is desired, then the minNmbActLnk threshold can be set to the actual number of links in the linkset, or greater than the number of links in the linkset.

Signaling link test

MTP 3 requires a successful signaling link test (SLTM generated and SLTA response expected) as part of link activation before considering a signaling link active. Then the signaling link test is performed periodically on each active signaling link, at a configurable period (link timer T34, corresponding to ANSI T1.111.7/ITU-T Q.707 timer T2). Signaling link testing is performed with no user part or application interaction.

The SLTM/SLTA exchange is not used in Japanese variants. In this case, successful alignment at MTP layer 2 is considered successful alignment at MTP layer 3.

Signaling route management

When configured as an STP, MTP 3 implements the signaling route management procedures transfer prohibited, transfer allowed, transfer restricted, signaling route set test, and signaling route set congestion test described in the ANSI and ITU-T MTP standards. MTP 3 performs these procedures without interacting with the user parts or applications.

MTP 3 service users

The MTP 3 data interface supports one or more applications using service access points (SAPs). One SAP is defined for each application that uses the MTP 3 service. An application binds to a particular SAP at initialization time, specifying the SAP ID to which it wants to bind. Each SAP is associated with a service information octet (SIO) value, which in turn identifies the upper layer protocol in use on that SAP (for example ISUP, TUP, SCCP). Therefore, only one application process can handle incoming messages for a particular upper layer protocol (only one process receiving incoming ISUP messages).

The MTP 3 configuration file specifies the number of SAPs (MTP 3 users or MAX_USERS) and the characteristics of each SAP. For more information, refer to *Mtp3InitNSapCfg* on page 79 and the *NMS SS7 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. Allocate entity IDs as follows:

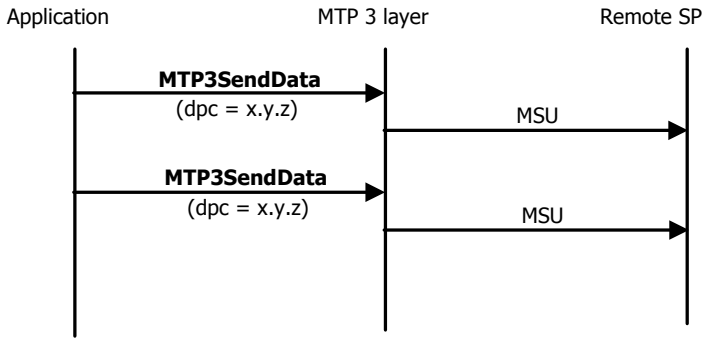
Range	Usage
0x00 through 0x1F 0x80 through 0xFF	Reserved for system utilities, configuration utilities, and management utilities.
0x20 through 0x7F	Reserved for applications.

Instance IDs identify the processor on which the entity executes. The host is always processor 0 (zero). All host-resident MTP 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

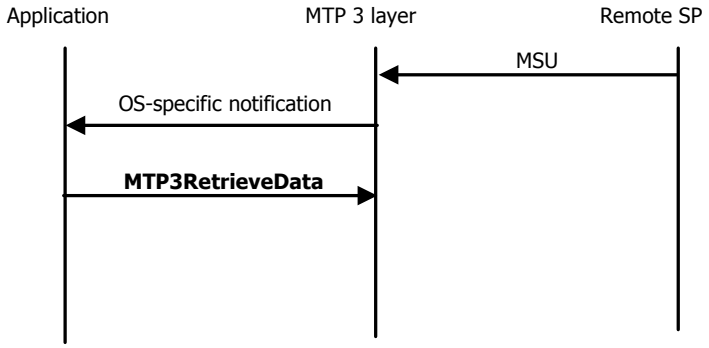
MTP 3 is a connectionless protocol: after an application binds to the MTP 3 layer, it can begin sending data using **MTP3SendData**. If the call succeeds, but the message is undeliverable, the application receives a status indication. When the message is successfully delivered, no status indication is sent. For more information, refer to *Status indications* on page 47.

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 **MTP3RetrieveMessage** 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 MTP 3 task. **MTP3RetrieveMessage** returns MTP3_NO_MSG until a message is available. It returns MTP3_SUCCESS when a message is available.

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



Receiving 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. It 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 after a StatPaused indication if 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 MTP 3 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 MTP 3 layer discards the message.
StatRestart	Local MTP 3 layer is going through a restart. The restart is followed by a StatRestartEnds. The application must not send any data messages after receiving this indication. Messages will be discarded during the restart procedure.
StatRestartEnds	Local MTP 3 layer completed its restart. The application can resume sending data messages.
StatPrimary	Local MTP 3 is the primary in a redundant configuration.
StatBackup	Local MTP 3 is the backup in a redundant configuration.
StatStandAlone	MTP 3 is not in a redundant configuration.

Receiving extended status indications

An extended status indication is sent to all applications registered to receive extended status indications (using **MTP3RegXStaReq**) when certain components of the MTP layer change state. MTP 2 links are the only components that support this feature. The following table describes the extended status indications:

Extended status indication	Description
XStatLinkUp	Link finished aligning and is now available to the MTP 3 layer. Traffic can be carried over the available link.
XStatLinkDown	Link is out of service. The MTP 3 layer now uses other links, if available, to carry traffic.
XStatLinkInh	Link is inhibited. Links become inhibited by management requests. The network (MTP layer) never inhibits a link by itself, although it can uninhibit the link. An inhibited link is no longer used by the MTP 3 layer to carry traffic.
XStatLinkInhDen	MTP 3 management entity attempted to inhibit a link, but the inhibition was denied because inhibiting the link causes a destination to become inaccessible.
XStatLinkUninh	Link became uninhibited and is now available to the MTP 3 layer for carrying traffic. Links can be uninhibited by MTP 3 management entities or automatically by the MTP 3 layer.
XStatLinkUninhDen	Uninhibition of a link was denied. This indication occurs when the local MTP 3 layer is not able to negotiate the uninhibition of the link with the remote MTP layer. The link is still unavailable to the MTP 3 layer to carry traffic.
XStatLinkRemBlock	Link is remotely blocked as a result of a processor outage on the remote side of the link. The link is no longer available to carry traffic.
XStatLinkRemUnblock	Link is no longer remotely blocked. The link is now available to carry traffic unless inhibited or locally blocked.
XStatLinkLocBlock	Link is locally blocked as a result of a local processor outage. The link is no longer available to carry traffic.
XStatLinkLocUnblock	Link is no longer locally blocked. The link is now available to carry traffic unless inhibited or remotely blocked.

Controlling congestion

Understanding the MTP congestion control mechanisms and developing an effective application congestion control strategy is a critical step in developing a reliable system.

MTP tracks and controls both inbound and outbound congestion. When outbound congestion occurs, upper layers bound to MTP are notified, enabling them to take some corrective action such as reducing the traffic load that they generate before the congestion becomes severe and impacts the operation of the service. MTP also takes action on its own during both inbound and outbound congestion to assure that application problems or very high network traffic does not overwhelm normal operation.

In most places, MTP uses a four level congestion control strategy, where level 0 indicates that the destination is not congested and level 3 indicates the most congested state. This strategy matches the ANSI standards.

This topic discusses:

- Outbound congestion
- Inbound congestion
- Application flow control

Outbound congestion

In MTP, there are two possible causes of outbound congestion:

- The application generates MTP traffic at a rate greater than the capacity of the SS7 links or downstream network, resulting in network overload.
- The application generates MTP requests faster than the MTP layer can be process them, resulting in the MTP service send queue building up beyond pre-determined thresholds (MTP service congestion).

Network overload

Network overload occurs when the MTP layer outbound queues build up beyond configured limits due to a traffic load exceeding the capacity of the available signaling links or to receipt of a transfer controlled message (TFC) regarding a congested destination.

When outbound traffic exceeds the total link capacity, transmission queues in the MTP 2 layer begin to build up. When the MTP 2 transmission queue for a link becomes full (layer 2 configuration parameter L2_TXQ_THRESH2), the MTP 3 layer ceases sending to the congested link, causing MTP 3 queues to begin building up. There are four configurable levels at which congestion indications of that priority are generated. The following table lists the parameters and defaults from the LINK section in the MTP 3 configuration file:

Parameter	Default
P0QUE_LENGTH	16
P1QUE_LENGTH	32
P2QUE_LENGTH	64
P3QUE_LENGTH	128

Note: These MTP 3 queues are in addition to the L2_TXQ_THRESH2 messages queued at layer 2.

Whether MTP 3 transmit queues are built due to network overload or because a transfer controlled (TFC) was received about a remote destination, the application is notified with an MTP status indication. This indication is the Natural Access event MTP3EVN_DATA with a message code of MTP3_STAT_IND and status of StatCongested. The indication contains the affected pointed code and the current congestion level (0 through 3). The application must reduce its traffic load toward the affected destination until the congestion abates.

In ANSI networks and in other national networks employing multiple congestion levels, the application must not generate any new traffic towards the affected destination with a priority lower than the current destination congestion level or it is discarded at the MTP layer.

For the international signaling network and other ITU-based networks without multiple congestion priorities, the application must reduce the traffic load toward the affected destination, as the MTP layer discards outgoing packets only in cases of excessive queuing of traffic to congested signaling links. If the application fails to reduce its traffic load toward the congested destination, the congestion condition can escalate.

When the network overload condition ceases, the application receives the Natural Access event MTP3EVT_DATA with a message code of MTP3_STAT_IND and status of StatCongestionEnds indicating that the application can resume normal traffic towards the affected destination.

MTP service congestion

MTP 3 service congestion occurs when an application generates traffic faster than the MTP layer can accept it, resulting in the MTP 3 service transmission queue building beyond pre-determined thresholds. This situation applies only to applications that use the MTP 3 service to directly communicate with MTP. The application is notified of congestion with an MTP3EVN_CONGEST Natural Access event that includes the current MTP 3 congestion level (0 through 3, where 0 indicates that congestion ceased). As the MTP 3 congestion level increases, the application is expected to reduce its traffic load proportionately until the congestion ceases.

By default, the MTP 3 service allocates a buffer pool for up to 256 requests to be queued to the MTP layer. If the application fails to reduce its traffic load enough to ease the congestion, eventually the MTP 3 service buffer pool becomes depleted and the MTP 3 send functions fail with a CTAERR_OUT_OF_MEMORY return code. The application can increase the number of buffers in the pool by setting service argument array element six to a number between 128 and 1024 when opening the MTP service. The increased number of buffers allows a larger burst of traffic to be absorbed without triggering congestion at the cost of using more host memory. For more information, refer to *Using ctaOpenServices* on page 38.

Congestion onset and abatement thresholds are always set to a fixed percentage of the buffers in use (queued to the MTP layer) regardless of the total size of the pool, as shown in the following table:

Congestion level	Onset threshold (to reach this level)	Abatement threshold (to next lower level)
1	Greater than 75% of pool in use.	Less than 50% of pool in use.
2	Greater than 85% of pool in use.	Less than 80% of pool in use.
3	Greater than 95% of pool in use.	Less than 90% of pool in use.

Inbound congestion

MTP inbound congestion is caused by the inability of the MTP application to read incoming messages as fast as they are generated by the network, resulting in a build-up of the user SAP queue or a depletion of the layer 1 limited pools used to receive incoming messages on each link.

Unlike outbound congestion, the MTP application is not directly notified of inbound congestion level changes to prevent escalation of the congestion condition. However, an alarm is always generated when a change occurs in the inbound congestion level for an MTP user SAP.

For inbound congestion, the MTP layer cannot rely on the application to reduce its traffic load to ease the congestion, as the source of the traffic bursts is generally other network nodes. The MTP layer acts directly to control inbound congestion in two ways:

- As the SAP queues to upper layers build up, configurable thresholds are crossed that set the SAP congestion priority. For national networks, MTP discards inbound messages with a priority lower than the current SAP congestion level, ensuring that a slow or dead upper layer cannot starve out other upper layers. No such discarding is done for international networks that have no message priorities. The limited pools ensure that all board memory is not used up in that case, but a slow or dead upper layer could starve out other upper layers.
- After the level 1 limited pools reach a defined threshold of allocated buffers, MTP 2 begins generating SIBs (status indication busy) out that link. Enough buffers remain in the limited pool to handle an additional window of 127 messages after SIBs are started. These limited pools disallow a renegade link from using up all of MTP's memory.

There are four configurable levels at which discarding of lower priority national messages occurs. The following table lists the parameters and defaults from the SAP section in the MTP 3 configuration file:

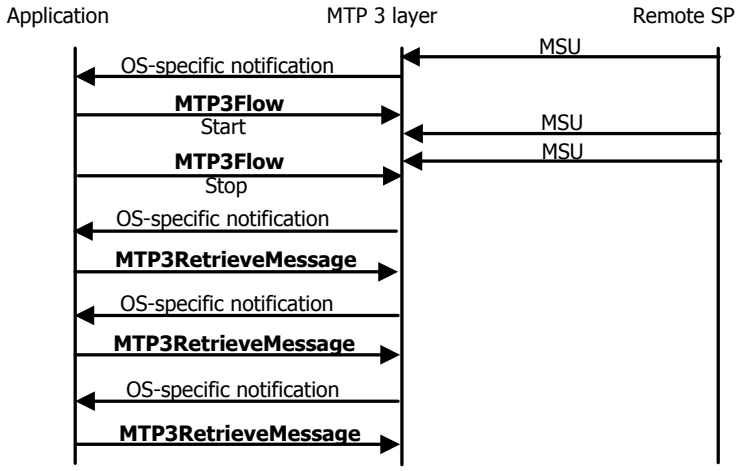
Parameter	Default
P0QUE_LENGTH	0
P1QUE_LENGTH	512
P2QUE_LENGTH	798
P3QUE_LENGTH	896

For example, when there are 512 messages queued to an upper layer, the SAP congestion priority becomes 1 and thereafter national messages of priority 0 are discarded rather than being queued. Similarly, when the queue size reaches 798, messages of priority 0 and 1 are discarded.

The number of inbound messages discarded due to SAP congestion can be determined with **Mtp3NSapStatus**.

Application flow control

An application that cannot keep up with incoming messages can use flow control to force queuing of messages in the lower layers until it can again accept incoming data. The following illustration shows how **MTP3Flow** is used:



When flow control is in effect, messages build up in the MTP 3 SAP queue and are handled as described in *Inbound congestion* on page 33.

Japanese variants

The Japanese variants (JNTT and JTTC) differ from the ANSI and ITU specifications and from each other. This topic describes those differences.

Differences between Japanese and ITU variants

The JNTT and JTTC variants:

- Use 16 bit point codes.
- May use a signaling route test message, acknowledgement, and negative acknowledgement to determine destination availability (SRT/SRA/USN).
- Use the high order two bits of the length indicator byte for message priority.
- Do not support the SIPO message. If a processor outage condition occurs, the link is brought down.

Differences between JNTT and JTTC

The JTCC variant:

- Does not use timer T5, the second changeback timer.
- Uses congestion messages.
- Allows only 40 outstanding unacknowledged packets rather than the full standard window of 127 packets. As a result, JTTC is likely to reach congestion faster than other configurations.
- Ignores 2 of 3 bad BSNs or FIBs.
- Discards packets with a bad FSN and generates a negative acknowledgement.
- Always uses emergency alignment.
- Does not support the user part unavailable message (UPU).
- Allows multiple destinations for transferring and routing set test messages and their cluster variants (TFP, TFR, TFA, TCP, TCR, TCA, RSP, RSR, RCP, RCR).

4 Using the MTP 3 service

Initializing the MTP 3 service under Natural Access

MTP 3 data functions are implemented as a Natural Access service. Natural Access is a development environment for telephony and signaling applications that 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 MTP 3 service.

Before calling any MTP 3 service functions, the application must first:

1. Initialize the Natural Access run-time environment.
2. Create the desired queues and contexts.
3. Open the MTP 3 service to bind itself to the desired MTP 3 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      mtp3Initparms       = {0};
CTA_SERVICE_NAME    mtp3ServiceNames[]  = {{"MTP3", "MTP3MGR"}};
...
mtp3Initparms.size              = sizeof(CTA_INIT_PARMS);
mtp3Initparms.traceflags        = CTA_TRACE_ENABLE;
mtp3Initparms.parmflags         = CTA_PARM_MGMT_SHARED;
mtp3Initparms.ctacompatlevel    = CTA_COMPATLEVEL;

Ret = ctaInitialize(mtp3ServiceNames, 1, &mtp3Initparms);
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 22. 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, "Mtp3SAP-%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 MTP 3 user SAP 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	MTP 3 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 mtp3OpenSvcLst[] = {{{"MTP3","MTP3MGR"}, {0}, {0}, {0}}};

mtp3OpenSvcLst[0].svcargs.args[0] = Board;     /* board number          */
mtp3OpenSvcLst[0].svcargs.args[1] = Sio;       /* Service Information
                                                * Octet                */
mtp3OpenSvcLst[0].svcargs.args[2] = NSapNmb;   /* network SAP number    */
mtp3OpenSvcLst[0].svcargs.args[3] = MyEnt;     /* application entity ID */
mtp3OpenSvcLst[0].svcargs.args[4] = DFLT_INST; /* Inst ID               */
mtp3OpenSvcLst[0].svcargs.args[5] = SuId;      /* Service user ID       */
mtp3OpenSvcLst[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 MTP3 events */
...


Ret = ctaOpenServices( ctaHd, mtp3OpenSvcLst, 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 MTP3 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 **MTP3RetrieveMessage**) 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 MTP 3 user SAP, the application receives a MTP3RUNSTATEIND event indicating the redundancy state of the MTP 3 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	MTP 3 task on this board is currently the primary board in a redundant board pair; normal operation is allowed as long as the board remains the primary.
SN_HAST_BACKUP	MTP 3 task on this board is currently the backup board in a redundant board pair, monitoring the status of the primary; no active traffic passes through this SAP until the board becomes the primary member of the pair.

The MTP3RUNSTATEIND 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 *SS7 Health Management Developer's Reference Manual* for details on writing redundant MTP 3 applications.

5 MTP 3 service function reference

MTP 3 service function summary

The MTP 3 service consists of the following asynchronous functions:

Function	Description
MTP3SendData	Requests data to be transmitted to a specified signaling point.
MTP3DeregXStaReg	Discontinues sending extended status indications to the specified service access point.
MTP3RetrieveMessage	Checks for and retrieves an incoming message from the MTP 3 layer.
MTP3GetApiStats	Retrieves statistics from the service about congestion level activity.
MTP3RegXStaReq	Registers to receive extended status indications.
MTP3Flow	Applies or halts flow control to the specified service access point.

Using the MTP 3 service function reference

This section provides an alphabetical reference to the MTP 3 service functions. A typical function includes:

Prototype	The prototype is followed by a list of the function arguments. NMS Communications data types include: • U8 (8-bit unsigned) • U16 (16-bit unsigned) • S16 (16-bit signed) • U32 (32-bit unsigned) • Bool (8-bit unsigned) If a function argument is a data structure, the complete data structure is defined.
Return values	The return value for a function is either MTP3_SUCCESS or an error code. For asynchronous functions, a return value of MTP3_SUCCESS (zero) indicates the function was initiated; subsequent events indicate the status of the operation.

MTP3DeregXStaReg

Discontinues sending extended status indications to the specified service access point.

Prototype

DWORD NMSAPI **MTP3DeregXStaReg** (CTAHD ***ctahd***, S16 ***sapId***)

Argument	Description
ctahd	Natural Access handle.
sapId	Service access point ID. Specify the same value used for the **ctaOpenServices** request.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_INVBOARD	Invalid board number.
MTP3_NOT_BOUND	**ctaOpenServices** not previously called.
MTP3_OSERROR	Lower-level drivers or task returned an error.

Details

By default, extended status indications are not sent. For more information, refer to *Receiving extended status indications* on page 30.

See also

MTP3RegXStaReq

MTP3Flow

Applies or halts flow control to the specified service access point.

Prototype

DWORD NMSAPI **MTP3Flow** (CTAHD ***ctahd***, S16 ***sapId***, S16 ***action***)

Argument	Description
ctahd	Natural Access handle.
sapId	Service access point ID. Specify the same value used for the **ctaOpenServices** request.
action	Flow control action to take: zero to disable, non-zero to enable.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_INVBOARD	Invalid board number.
MTP3_NOT_BOUND	**ctaOpenServices** not previously called.
MTP3_OSERROR	Lower-level drivers or task returned an error.

Details

If flow control is active, incoming data messages are buffered in the MTP 3 process until flow control is deactivated. For more information, see *Controlling congestion* on page 31.

MTP3GetApiStats

Retrieves statistics from the service about congestion level activity.

Prototype

DWORD **MTP3GetApiStats** (CTAHD ***ctahd***, MTP3APISTATS ****pStats***, U8 ***bReset***)

Argument	Description
ctahd	Natural Access handle.

pStats	Pointer to a buffer where the statistics are returned:
	`typedef struct`
	`{`
	`  U32 qCount;  /* number of API messages currently queued */`
	`  /* to MTP3 layer */`
	`  U32 qPeak;  /* max number of API messages ever queued to */`
	`  /* MTP3 layer */`
	`  U32 txPending;  /* current number of outstanding transmit */`
	`  /* rqsts to MTP3 layer */`
	`  U32 txPendPeak;  /* max number of transmit rqsts ever */`
	`  /* outstanding to MTP3 layer */`
	`  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 MTP3 layer */`
	`  U32 rxFailed;  /* number of receive failure events from */`
	`  /* MTP3 layer */`
	`  U8  apiQCongLvl; /* current outbound queue congestion level */`
	`  /* [0..3] */`
	`  U8  spare1;  /* spare for alignment */`
	`  U16 spare2;  /* spare for alignment */`
	`} MTP3APISTATS;`

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

Return values

Return value	Description
MTP3_SUCCESS	
CTAERR_INVALID_CTAHD	Invalid handle.

MTP3RegXStaReq

Registers to receive extended status indications.

Prototype

DWORD NMSAPI **MTP3RegXStaReq** (CTAHD ***ctahd***, S16 ***sapId***)

Argument	Description
ctahd	Natural Access handle.
sapId	Service access point ID. Specify the same value used for the **ctaOpenServices** request.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_INVBOARD	Invalid board number specified.
MTP3_NOT_BOUND	**ctaOpenServices** not previously called.
MTP3_OSERROR	Lower-level drivers or task returned an error.

Details

By default, extended status indications are not sent. For more information, refer to *Receiving extended status indications* on page 30.

See also

MTP3DeregXStaReg

MTP3RetrieveMessage

Checks for and retrieves an incoming message from the MTP 3 layer.

Prototype

DWORD NMSAPI **MTP3RetrieveMessage** (CTAHD ***ctahd,*** void ****pMsgInd***, short ****Length***)

Argument	Description
ctahd	Natural Access handle.
pMsgInd	Pointer to 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.
Length	Pointer to buffer where the length of the indication is returned.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_INVBOARD	Invalid board number.
MTP3_NOT_BOUND	**ctaOpenServices** not previously called.
MTP3_NO_MSG	No message currently waiting.

Details

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

- Call **MTP3RetrieveMessage** periodically within a polling process. The application must poll regularly to avoid excessive queuing of messages in the TX driver or the MTP 3 task.
- Wait for an asynchronous notification that a message is available, and then call **MTP3RetrieveMessage** to retrieve it.

Applications can receive the following types of messages:

- Data indications
- Status indications
- Extended 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;           /* MTP3_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	Whether an indication is data or status. Always MTP3_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. Possible values are 0 through 0xFFFFFF.
dpc	Destination point code from the routing label of the incoming message. Possible values are 0 through 0xFFFFFF.
srvInfo	Service information octet from the incoming message specifying the upper layer protocol and the network indicator.
lnkSel	Link selector field from the incoming message. Possible values are: 0 through 15 (ITU-T) 0 through 31 (ANSI) 0 through 255 (TUP only)
data	Data received.

Status indications

An application receives a STAT_IND structure when an important status change occurs on a circuit matching the application's service information octet. Status changes are generated by the local MTP 3 layer. For more information, refer to *Receiving status indications* on page 29.

```
typedef struct stat_ind_s
{
  U8   code;      /* MTP3_STAT_IND (0x7A)               */
  U8   spare1;    /* Alignment                          */
  U16  sapId;     /* Service user Id from Open Services */
  U32  pc;        /* Point Code related to the status ind */
  S16  status;    /* Status indicator. See defines above */
  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 MTP3_STAT_IND for status indications.
sapId	Service user identifier passed as suId in the **ctaOpenServices** call.
pc	Point code of the remote node affected by this status indication.
status	Status event that occurred: StatPaused = Data traffic paused StatResumed = Data traffic resumed StatCongested = Data congested StatUsrUnavail = User unavailable StatRestart = Restart in progress StatRestartEnd = Restart completed StatCongestionEnds = Congestion ended StatPrimary = MTP 3 is primary in redundant configuration StatBackup = MTP 3 is backup in redundant configuration StatStandAlone = MTP is not in a redundant configuration
priority	Current congestion level for congestion related indications. Valid range is 0 (lowest) through 3 (highest).

Extended status indications

If the application registered to receive extended status indications (using **MTP3RegXStaReq**), the application receives the XSTAT_IND structure when an important status change occurs on a link. The local MTP 3 layer generates status changes. For more information, refer to *Receiving extended status indications* on page 30.

```
type def struct xstat_ind_s
{
  U8   code;    /* MTP3_XSTAT_IND (0x3A)              */
  U8   spare1;  /* Alignment                          */
  U16  sapId;   /* Service user Id from Open Services */
  U16  status;  /* Extended Status Type               */
  U16  link;    /* Affected MTP3 link                 */
  U16  spare2;  /* Reserved for future use            */
  U16  spare3;  /* Reserved for future use            */
} XSTAT_IND;
```

XSTAT_IND contains the following fields:

Field	Description
code	Whether an indication is data or status. Always MTP3_XSTAT_IND for status indications.
sapId	Service user identifier passed as suId in the **ctaOpenServices** call. Valid range is 0 through 15.
link	Link number of the affected MTP 3 link. Valid range is 0 to 31.
status	Status event that occurred: XStatLinkUp = Link up XStatLinkDown = Link down XStatLinkInh = Link inhibited XStatLinkInhDen = Link inhibit denied XStatLinkUninh = Link uninhibited XStatLinkUninhDen = Link uninhibit denied XStatLinkRemBlock = Link remotely blocked XStatLinkRemUnblock = Link remotely unblocked XStatLinkLocBlock = Link locally blocked XStatLinkLocUnblock = Link locally unblocked

MTP3SendData

Requests data to be transmitted to a specified signaling point.

Prototype

DWORD NMSAPI **MTP3SendData** (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 used by MTP 3 to choose the link to send data over. Valid ranges are 0 through 15 for ITU-T and 0 through 31 for ANSI.
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) in which the application is interested.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_INVBOARD	Invalid board number.
MTP3_NOT_BOUND	**ctaOpenServices** not previously called.
MTP3_OSERROR	Lower-level drivers or task returned an error.
MTP3_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.

For TUP only, all 8 bits are available to the application with no ***lnkSel*** range checking. For more information, refer to *Outbound messages* on page 19.

The user data sent is unique. The first byte corresponds to the first byte following the routing label in the underlying SS7 data message. For example, when constructing an ISUP message, the first byte of user data is the first byte of the circuit identification code (CIC). The MTP layer 3 constructs the routing label and SIO values from other parameters and bind information. 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. Only the service indicator field is used for routing of incoming messages.

If you receive an MTP3_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 MTP3EVN_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 31.
- Reduce other host to board traffic, including management function traffic.

For more information, see *Transferring data* on page 28.

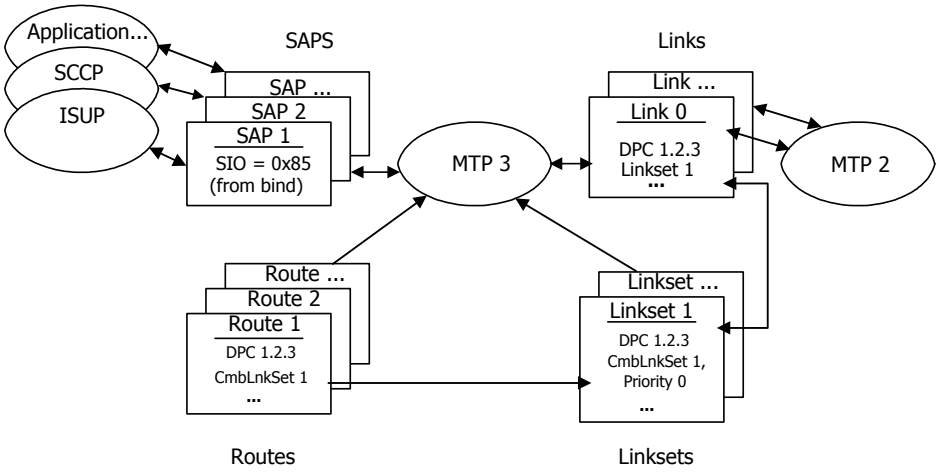
6 Managing the MTP 3 task

Configuring MTP 3

Define the following configurations for the MTP 3 layer:

- General configuration
- Service access points (SAPs) configuration
- Link configuration
- Linkset configuration
- Route configuration

The following illustration shows MTP 3 configurable entities:



General configuration

General configuration parameters define and control the general operation of the signaling point (SP) implemented by the SS7 software. General configuration parameters include the type of signaling point being constructed (SP or STP), the point code assigned to the signaling point, the values for various SP-level timers, and the maximum number of other configurable elements (SAPs, links, linksets, routes) to control memory allocation. General parameters are configured once at board download time, before the other entities are configured. After the general configuration is performed, only the restart required attribute, timers, and routing masks can be reconfigured using **Mtp3SetGenCfg**. The board must be downloaded again to change any of the other general configuration parameters.

Service access point configuration

Service access points (SAPs) define the SS7 applications that use MTP 3. The configurable attributes of SAPs include the:

- Protocol variant.
- Point code length supported by the application associated with the SAP.
- Maximum number of application messages to be queued at each of the four possible message priority levels when flow control between MTP 3 and the application is in effect.

SAPs can be defined any time after the general configuration parameters are defined, up to the maximum number of SAPs allowed by the general configuration parameters definition. After the SAP is defined, all parameters can be changed by calling the SAP configuration function, **Mtp3SetNSapCfg**.

Link configuration

Links define physical signaling links between the TX board and the adjacent signaling points. The configurable attributes of a link include:

- The point code of the adjacent signaling point.
- The protocol variant employed on the link.
- The point code length.
- The maximum packet length.
- Various timer values.
- Membership in a linkset.

Links can be defined any time after the general configuration parameters are defined, up to the maximum number of links allowed by the general configuration parameters definition. After a link is defined, some of its attributes can be changed by calling the link configuration request, **Mtp3SetLinkCfg**. Links can also be deleted with the remove link control request.

Linkset configuration

Linksets are groups of 1 to 32 links that directly connect two signaling points. The configurable attributes of a linkset include:

- The point code of the adjacent signaling point.
- The number of links to attempt to keep active.

Linksets can be defined any time after the general configuration parameters are defined, up to the maximum number of linksets allowed by the general configuration parameters definition. After a linkset is defined, the target number of active links and the defined combined linksets can be changed by calling the linkset configuration request, **Mtp3SetLinkSetCfg**. The board must be downloaded again to change any of the other linkset configuration parameters.

Route configuration

Routes specify the destination signaling points (subnetworks or clusters) accessible from the node being configured. Each route is assigned a direction, either up or down. Up routes are used to identify incoming messages to be routed up to the applications. One down route is required for each remote signaling point/network/cluster that is to be accessible from the SP being configured. Down routes route outgoing messages across the appropriate signaling links.

Other configurable attributes of routes include the destination point code and the protocol variant in use at the destination SP/cluster/network. Routes can be defined any time after the general configuration parameters are defined, up to the maximum number of routes allowed by the general configuration parameters definition. After a route is defined, some of its attributes can be changed by calling the route configuration request, **Mtp3SetRouteCfg**.

Retrieving status information

Use MTP 3 status requests to enable the application to retrieve the following status information:

Entity	Request	Status returned includes
Signaling point	**Mtp3GenStatus**	Restart in progress or not High availability state (primary | backup | stand alone)
SAP	**Mtp3NSapStatus**	State (bound | unbound) Flow control status (on | off) Congestion discard count
Link	**Mtp3LinkStatus**	Current state (active | inactive) Remotely blocked (yes | no) Locally blocked (yes | no) Remotely inhibited (yes | no) Locally inhibited (yes | no) Congested (yes | no) Emergency state (yes | no)
Linkset	**Mtp3LinkSetStatus**	Current state (active | inactive) Congested (yes | no) Number of active links Number of congested links
Route	**Mtp3RouteStatus**	Current state (accessible | inaccessible) Congested (yes | no) Number of active linksets Number of congested linksets Destination restart in progress (yes | no)

Controlling MTP 3 entities

Use MTP 3 control requests to enable the application to control MTP 3 entities in the following ways:

Entity	Request	Control request type
Signaling point	**Mtp3MgmtCtrl**	Enable, disable alarms Enable, disable flow control Enable, disable data tracing
Link		Enable, disable link Inhibit, uninhibit link Delete a link Start, end local processor outage Start, end emergency condition Enable, disable data tracing
Linkset		Activate, deactivate linkset

Retrieving statistics

Use MTP 3 statistics requests to enable the application to retrieve and optionally reset the following statistics:

Entity	Request	Statistics returned include
Signaling point	**Mtp3GenStats**	Transmit and receive counts for user part unavailable messages, traffic restarts, and packets dropped.
Links	**Mtp3LinkStats**	Counts for: • The various message types and their acknowledgments both transmitted and received. • Queue levels and high water marks. • The number of times each congestion level was reached.
Linksets	**Mtp3LinkSetStats**	Number of times a linkset has failed and recovered and the total duration of linkset unavailability.
Routes	**Mtp3RouteStats**	Transmit and receive counts for messages such as test, congestion, transfer prohibited, transfer restricted, and transfer allowed, as well as number of routes unavailable and duration of route unavailability.

7 MTP 3 management function reference

MTP 3 management function summary

NMS MTP 3 consists of the following synchronous management functions in which the action is completed before control is returned to the application:

- Configuration functions
- Control functions
- Statistics functions
- Status functions

Configuration functions

Function	Description
Mtp3GetGenCfg	Obtains the current values of the general configuration parameters in the MTP 3 task.
Mtp3GetLinkCfg	Obtains the current data link configuration values of the specified link number.
Mtp3GetLinkSetCfg	Obtains the current configuration values of the specified linkset.
Mtp3GetNSapCfg	Obtains the current NSAP configuration values from the MTP 3 task.
Mtp3GetRouteCfg	Obtains the current route configuration values for the specified destination point code.
Mtp3InitGenCfg	Initializes the general configuration structure with default values and the provided originating point code.
Mtp3InitLinkCfg	Initializes the provided data link configuration structure with default values and link number, link type, and destination point code.
Mtp3InitLinkSetCfg	Initializes the provided linkset configuration structure with default values, the specified linkset number, and the destination point code.
Mtp3InitNSapCfg	Initializes the provided NSAP configuration structure with default values and the specified link type.
Mtp3InitRouteCfg	Initializes the provided route configuration structure with default values, the specified route number, the DPC, and the switch type.
Mtp3SetGenCfg	Sets the general configuration values in the MTP 3 task.
Mtp3SetLinkCfg	Configures the MTP 3 task with the data link configuration values contained in the provided MTP3LinkCfg structure.
Mtp3SetLinkSetCfg	Configures the MTP 3 task with the linkset configuration values contained in the provided MTP3LinkSetCfg structure.
Mtp3SetNSapCfg	Configures the MTP 3 task with the NSAP configuration values contained in the provided MTP3NSapCfg structure.
Mtp3SetRouteCfg	Configures the MTP 3 task with the route configuration values contained in the provided MTP3RouteCfg structure.

Control functions

Function	Description
Mtp3MgmtCtrl	Sends a control request to the MTP 3 task.
Mtp3MgmtInit	Initializes internal structures and opens communication with the MTP 3 process on the TX board.
Mtp3MgmtTerm	Terminates the dual port RAM channel binding specified in **Mtp3GenStatus** for this application.

Statistics functions

Function	Description
Mtp3GenStats	Obtains and potentially resets general or global statistics for the MTP 3 signaling point.
Mtp3LinkSetStats	Obtains (and potentially resets) linkset statistical information about the specified linkset number.
Mtp3LinkStats	Obtains and potentially resets data link statistical information about the specified link number.
Mtp3RouteStats	Obtains and potentially resets routing statistics for the route associated with the specified destination point code.

Status functions

Function	Description
Mtp3GenStatus	Obtains general status information about the MTP 3 signaling point.
Mtp3LinkSetStatus	Obtains linkset status information about the specified linkset number.
Mtp3LinkStatus	Obtains data link status information about the specified link number, including link state, flow control state, queue sizes, and blocked, inhibited, and congestion status.
Mtp3NSapStatus	Obtains NSAP status information from the MTP 3 task, including the NSAP state (bound or unbound) and the flow control state (on or off).
Mtp3RouteStatus	Obtains routing status information about the route associated with the specified destination point code.
Mtp3RouteTest	Initiates a route test by transmitting an SRT message (JNTT only).

Using the MTP 3 management function reference

This section provides an alphabetical reference to the MTP 3 management functions. A typical function includes:

Prototype	The prototype is followed by a list of the function arguments. NMS Communications data types include: • U8 (8-bit unsigned) • S16 (16-bit signed) • U16 (16-bit unsigned) • U32 (32-bit unsigned) • Bool (8-bit unsigned) If a function argument is a data structure, the complete data structure is defined.
Return values	The return value for a function is either MTP3_SUCCESS or an error code.

Mtp3GenStats

Obtains and potentially resets general or global statistics for the MTP 3 signaling point.

Prototype

MTP3_STATUS **Mtp3GenStats** (U8 ***board***, MTP3GenStats ****pStats***, BOOL ***bReset***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through ***maxboard*** (currently 8).

pStats	Pointer to a buffer provided by the application where the requested statistics are returned:
	`typedef struct _Mtp3GenStats`
	`/* MTP Level 3 signaling point statistics */`
	`{`
	`U32 usrUnavailRx; /* User part unavailable received */`
	`U32 usrUnavailTx; /* User part unavailable transmitted */`
	`U32 traTx; /* Traffic restart allowed transmitted */`
	`U32 traRx; /* Traffic restart allowed received */`
	`U32 trwTx; /* Traffic restart waiting transmitted */`
	`U32 trwRx; /* Traffic restart waiting received */`
	`U32 msuDropRteErr; /* MSU dropped due to a routing data error */`
	`} MTP3GenStats;`

bReset	Optionally resets statistics.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_HANDLE	**Mtp3MgmtInit** not called for the specified board.

Details

General statistics include transmit and receive counts for user part unavailable messages, traffic restarts, and packets dropped.

If the ***bReset*** value is a non-zero integer, the statistics are reset after returning the current values. A value of zero disables the reset function.

See also

Mtp3GenStatus, **Mtp3MgmtInit**

Mtp3GenStatus

Obtains general status information about the MTP 3 signaling point.

Prototype

MTP3_STATUS **Mtp3GenStatus** (U8 ***board***, MTP3GenStatus ****pStats***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through ***maxboard*** (currently 8).

pStats	Pointer to a buffer provided by the application where the requested statistics are returned:
	`typedef struct _Mtp3GenStatus`
	`/* MTP Level 3 - signaling point status */`
	`{`
	`Bool spRst; /* SP restarting flag */`
	`U8 haState; /* Current high-availability state */`
	`} MTP3GenStatus;`
	See the Details section for field descriptions.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_HANDLE	**Mtp3MgmtInit** not called for the specified board.

Details

The MTP3GenStatus structure includes a Boolean flag specifying whether or not the signaling point is in the process of restarting, and the current high-availability state. The following table describes the fields in the structure:

Field	Description
spRst	If TRUE, indicates the signaling point is restarting.
haState	Must be one of the following: SN_HAST_STANDALONE = None (redundant state). SN_HAST_STARTING = Initial redundant state. SN_HAST_PRIMARY = MTP 3 is the primary in redundant state. SN_HAST_BACKUP = MTP 3 is the backup in redundant state.

See also

Mtp3MgmtInit

Mtp3GetGenCfg

Obtains the current values of the general configuration parameters in the MTP 3 task.

Prototype

MTP3_STATUS **Mtp3GetGenCfg** (U8 ***board***, MTP3GenCfg ****pGenCfg***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through ***maxboard*** (currently 8).
pGenCfg	Pointer to an MTP3GenCfg structure to be filled in by the MTP 3 task.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_HANDLE	**Mtp3MgmtInit** not called for the specified board.
MTP3_TIMEOUT	No response from the board.

Details

This function can be called any time after **Mtp3MgmtInit**. The application must provide a pointer to a buffer large enough to hold the MTP3GenCfg structure. Refer to *Mtp3InitGenCfg* on page 68 for more information about the MTP3GenCfg structure.

See also

Mtp3GenStatus, **Mtp3InitGenCfg**, **Mtp3MgmtInit**

Mtp3GetLinkCfg

Obtains the current data link configuration values of the specified link number.

Prototype

MTP3_STATUS **Mtp3GetLinkCfg** (U8 ***board***, MTP3LinkCfg ****pLinkCfg***, S16 ***linkNo***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through ***maxboard*** (currently 8).
pLinkCfg	Pointer to an MTP3LinkCfg structure where the data link configuration values are filled.
linkNo	Link number for which to obtain configuration information.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_HANDLE	**Mtp3MgmtInit** not called for the specified board.
MTP3_TIMEOUT	No response from the board.

Details

This function can be called any time after **Mtp3MgmtInit**. An application must provide a pointer to a buffer large enough for the MTP3LinkCfg structure. Refer to *Mtp3InitLinkCfg* on page 71 for more information about the MTP3LinkCfg structure.

See also

Mtp3GenStatus, **Mtp3MgmtInit**

Mtp3GetLinkSetCfg

Obtains the current configuration values of the specified linkset.

Prototype

MTP3_STATUS **Mtp3GetLinkSetCfg** (U8 ***board***, MTP3LinkSetCfg ****pLinkSetCfg***, S16 ***linkSetNo***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through ***maxboard*** (currently 8).
pLinkSetCfg	Pointer to an MTP3LinkSetCfg structure where the linkset configuration values are filled.
linkSetNo	Linkset number from which to obtain configuration information.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_HANDLE	**Mtp3MgmtInit** not called for the specified board.
MTP3_TIMEOUT	No response from the board.

Details

This function can be called any time after **Mtp3MgmtInit**. An application must provide a pointer to a buffer large enough to hold the MTP3LinkSetCfg structure. Refer to *Mtp3InitLinkSetCfg* on page 77 for more information about the MTP3LinkSetCfg structure.

See also

Mtp3GenStatus, **Mtp3MgmtInit**

Mtp3GetNSapCfg

Obtains the current NSAP configuration values from the MTP 3 task.

Prototype

MTP3_STATUS **Mtp3GetNSapCfg** (U8 ***board***, MTP3NSapCfg ****pNSapCfg***, S16 ***nSapNo***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through ***maxboard*** (currently 8).
pNSapCfg	Pointer to an MTP3NSapCfg structure where the network SAP configuration values are filled.
nSapNo	Network SAP number from which to retrieve the configuration information. Valid range is 0 through the maximum number of NSAPs minus 1 as specified in **Mtp3SetGenCfg**.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_HANDLE	**Mtp3MgmtInit** not called for the specified board.
MTP3_TIMEOUT	No response from the board.

Details

This function can be called any time after **Mtp3MgmtInit**. An application must provide a pointer to a buffer large enough to hold the MTP3NSapCfg structure. Refer to *Mtp3InitNSapCfg* on page 79 for more information about the MTP3NSapCfg structure.

See also

Mtp3GenStatus, **Mtp3MgmtInit**

Mtp3GetRouteCfg

Obtains the current route configuration values for the specified destination point code.

Prototype

MTP3_STATUS **Mtp3GetRouteCfg** (U8 ***board***, MTP3RouteCfg ****pRouteCfg***, U32 ***dpc***, U32 ***opc***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. This value must be from 1 through ***maxboard*** (currently 8).
pRouteCfg	Pointer to an MTP3RouteCfg structure where the current route configuration values are filled.
dpc	Destination point code associated with the route for which configuration information is obtained.
opc	Originating point code associated with the route for which configuration information is obtained. If set to zero, MTP uses the OPC specified in the general configuration.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_HANDLE	**Mtp3MgmtInit** not called for the specified board.
MTP3_TIMEOUT	No response from the board.

Details

This function can be called any time after **Mtp3MgmtInit**. An application must provide a pointer to a buffer large enough to hold the MTP3RouteCfg structure. Refer to *Mtp3InitRouteCfg* on page 81 for more information about the MTP3RouteCfg structure.

See also

Mtp3GenStatus, **Mtp3MgmtInit**

Mtp3InitGenCfg

Initializes the general configuration structure MTP3GenCfg with default values and the provided originating point code.

Prototype

MTP3_STATUS **Mtp3InitGenCfg** (MTP3GenCfg ****pGenCfg***, U32 ***opc***)

Argument	Description

pGenCfg	Pointer to the MTP3GenCfg structure to be initialized:
	`typedef struct _MTP3GenCfg`
	`{`
	`U32 spCode1; /* Our DPC for CCITT version */`
	`U32 spCode2; /* Our DPC for ANSI or CHINA version */`
	`U8 typeSP; /* Type of signaling point */`
	`Bool disableUPU; /* Disable sending of User Part */`
	`/* unavailable messages */`
	`Bool ssfValid; /* Ssf validation required */`
	`Bool rstReq; /* Restarting procedure required */`
	`U16 nmbLinks; /* Number of MTP data links */`
	`U16 nmbNSap; /* Number of Upper Layer Saps */`
	`U16 nmbRouts; /* Maximum number of routing entries */`
	`U16 nmbLnkSets; /* Number of link sets */`
	`U16 nmbRteInst; /* Number of simultaneous Rte instances */`
	`U8 nmbMasks; /* Number of masks */`
	`U8 pcDispFmt; /* Point code display format */`
	`U32 masks[MTP3MAXMASKS]; /* Route masks */`
	`S16 cbTimeRes; /* Time resolution */`
	`S16 icbTimeRes; /* Time resolution */`
	`S16 spTimeRes; /* Time resolution */`
	`S16 rteTimeRes; /* Time resolution */`
	`MTP3GenTimerCfg tmr; /* General timer configuration */`
	`PDesc stkmgr; /* Stack manager */`
	`U8 traceData; /* Trace data flag */`
	`Bool opcRouting; /* Outbound routing by OPC? */`
	`Bool transparentMode;/* Transparent mode */`
	`U8 spare2; /* Make an even number of longs */`
	`} MTP3GenCfg;`
	See the Details section for field descriptions.

opc	Point code of the local signaling point.

Return values

MTP3_SUCCESS

Details

This function enables an application to initialize an MTP3GenCfg structure before calling **Mtp3SetGenCfg** to set the general configuration parameters. **Mtp3InitGenCfg** sets the ***opc*** as specified and initializes all other fields in the MTP3GenCfg structure to their defaults.

A pointer to an MTP3GenCfg structure is passed to **Mtp3InitGenCfg** where default values are set. After initialization, an application can override zero to all of these values and then pass the pointer to **Mtp3SetGenCfg**, which sets the configuration. The default values provided by **Mtp3InitGenCfg** are adequate for most applications.

The spCode1 and spCode2 fields in the MTP3SetGenCfg structure store the local point code. spCode1 is used by MTP 3 if the link type is CCITT or a variant of CCITT. spCode2 is used by MTP 3 if the link type is ANSI or a variant of ANSI. Both fields

are set to the same value (the ***opc*** parameter) by **Mtp3InitGenCfg**. If a signaling point is acting as a gateway between CCITT and ANSI networks, or otherwise uses both types of point codes (14 and 24 bit), the application can override the defaults and specify different values for spCode1 and spCode2.

The following table lists the fields in the MTP3GenCfg structure that can be modified. A field name in **bold** indicates that subsequent calls to **Mtp3SetGenCfg** can update the field. Fields that do not appear in the following table are either unused or for internal use, and must not be modified.

Field	Description
typeSp	Signaling point type. Valid values are: MTP3TYPE_SP = Signaling endpoint MTP3TYPE_STP = Signaling transfer point. This is the default.
spCode1	Local point code for CCITT version. Must be specified with ***opc*** parameter. Valid values are 0 through 0xFFFFFF.
spCode2	Local point code for ANSI or CHINA version. Must be specified with ***opc*** parameter. Valid values are 0 through 0xFFFFFF.
rstReq	If TRUE, restart procedures are implemented as defined by the ANSI or ITU-T specification. Valid values are TRUE (default) or FALSE.
nmbLinks	Maximum number of configurable data links. Valid range is 1 through 32. Default is 4.
nmbNSap	Maximum number of configurable network SAPs. Valid range is 1 through 64. Default is 2.
nmbLnkSets	Maximum number of configurable linksets. Valid range is 1 through 32. Default is 1.
nmbRouts	Maximum number of configurable routes. Valid range is 1 through 32767. Default is 32.
nmbRteInst	Maximum number of simultaneous route instances. Valid range is 1 through 32767. Default is 1024.
nmbMasks	Number of route masks. Valid range is 0 through 8. Default is 0.
masks	Route masks. The entire structure contains zeros by default.
tmr	General timers configuration. Refer to the *MTP3GenTimerCfg structure* on page 70.
tracedata	TRUE starts tracing of all data traffic between MTP 3 and MTP 2. Default is FALSE.
opcRouting	If TRUE, outbound routing is performed by both OPC and DPC. For use with multiple OPC emulation. Default is FALSE meaning outbound routing is performed by DPC only. Note that you can still use multiple OPC emulation (which is for inbound routing) and set opcRouting to FALSE.
transparentMode	If TRUE, all inbound traffic regardless of DPC and OPC is passed to the bound upper layer that matches the SIO value and protocol variant of the message. This behavior does not include MTP messages that MTP still handles internally. All outbound traffic is spread across all links regardless of DPC and OPC. Default is FALSE, meaning normal routing is used.

MTP3GenTimerCfg structure

```
typedef struct _MTP3GenTimerCfg
{
  TimerCfg t15;      /* t15 - waiting to start route set       */
                     /* congestion test                        */
  TimerCfg t16;      /* t16 - waiting for route set congestion */
                     /* status update                          */
  /* ITU restart timers */
  TimerCfg t18_itu;  /* ITU only - time at restarting STP to   */
                     /* wait for TFP, TFR, TFA until enough    */
                     /* TRA's received.  Then send TFP, TFR's. */
  TimerCfg t20_itu;  /* ITU only - overall restart time at     */
                     /* restarting SP/STP                      */

  /* ANSI restart timers */
  TimerCfg t22_ansi; /* ANSI only - time at restarting SP to   */
                     /* wait for links to become available     */
  TimerCfg t23_ansi; /* ANSI only - time at restarting SP,     */
                     /* after T22, to wait for all TRA's       */
  TimerCfg t24_ansi; /* ANSI only - time at restarting STP,    */
                     /* after T23, waiting to send all TRA's   */
  TimerCfg t26_ansi; /* ANSI only - time at restarting SP      */
                     /* waiting to repeat TRW                  */
  TimerCfg t27_ansi; /* ANSI only - minimum duration of        */
                     /* unavailability for full restart        */
  TimerCfg tRteInst; /* Route instance life timer              */
  TimerCfg tResync;  /* Internal timer for re-syncing          */
                     /* checkpointing                          */
} MTP3GenTimerCfg;
```

The following table lists the MTP3GenTimerCfg fields that can be modified. Unless otherwise specified, the timer names correspond to the CCITT specification. Subsequent calls to **Mtp3SetGenCfg** can update any field in this table. Fields that do not appear in the table are either unused or for internal use, and must not be modified. Valid values for all fields in this table range from 1 through 65535 tenths of a second.

Field	Description
t15	Waiting to start route set congestion test. Default value is 30.
t16	Waiting for route set congestion status update. Default value is 20.
t18_itu	ITU restart timer for an STP during which links are restarted and TFA, TFR, and TFP messages are received. Default value is 300.
t20_itu	ITU overall restart timer. Default value is 600.
t22_ansi	ANSI restart timer at restarting SP waiting for links to become available. Default value is 300.
t23_ansi	ANSI restart timer at restarting SP waiting for TRA messages. Default value is 300.
t26_ansi	ANSI restart timer at restarting SP waiting to repeat TRW message. Default value is 130.
t27_ansi	Minimum duration of unavailability for full restart. Default value is 30.
trteinst	Route instance timer (internally used, not applicable to ANSI or CCITT specifications). Default value is 18000.

See also

Mtp3MgmtInit, **Mtp3SetGenCfg**

Mtp3InitLinkCfg

Initializes the provided data link configuration structure MTP3LinkCfg with default values and link number, link type, and destination point code.

Prototype

MTP3_STATUS **Mtp3InitLinkCfg** (U8 ***board***, MTP3LinkCfg ****pLinkCfg***, S16 ***linkNo***, U8 ***linkType***, U32 ***dpc***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through ***maxboard*** (currently 8).

pLinkCfg	Pointer to the MTP3LinkCfg structure to be initialized:
	`typedef struct _MTP3LinkCfg`
	`{`
	`  U16 lnkSetId;  /* Link set ID */`
	`  U16 lnkId;  /* Signalling link identity */`
	`  U32 opc;  /* Originating point code */`
	`  U32 adjDpc;  /* Adjacent Destination Point Code */`
	`  U8  lnkPrior;  /* Link priority within the link set */`
	`  U8  usePrior;  /* Use message priority, or force zero for */`
	`  /*  all priorities */`
	`  U16 msgSize;  /* Maximum message length */`
	`  U8  msgPrior;  /* Management message priority */`
	`  U8  lnkType;  /* Link type ANSI or CCITT */`
	`  U8  maxSLTtry;  /* Maximum times to retry SLTM */`
	`  U8  hsbit;  /* Alignment formerly spare1 */`
	`  S16 p0QLen;  /* Size of the priority 0 Q */`
	`  S16 p1QLen;  /* Size of the priority 1 Q */`
	`  S16 p2QLen;  /* Size of the priority 2 Q */`
	`  S16 p3QLen;  /* Size of the priority 3 Q */`
	`  U8  discPrior;  /* Discard priority */`
	`  U8  ssf;  /* Sub service field */`
	`  U8  lnkTstSLC;  /* Link selection code for link test */`
	`  U8  tstLen;  /* Link test pattern length */`
	`  U8  tst[MTP3LNKTSTMAX];  /* Link test pattern */`
	`  MTP3LinkTimerCfg tmr;  /* Timer configuration */`
	`  U16 dstProcId;  /* Destination processor id */`
	`  U8  dstEnt;  /* Entity */`
	`  U8  dstInst;  /* Instance */`
	`  U8  prior;  /* Priority */`
	`  U8  route;  /* Route */`
	`  U8  selector;  /* Lower layer selector */`
	`  U8  spare2;  /* Alignment */`
	`  MemoryId mem;  /* Memory region and pool id */`
	`  U8  dpcLen;  /* dpc or opc length */`
	`  U8  lnkIndex;  /* Index into link array.  0 to n-1 in order */`
	`  /*  defined in config */`
	`  S16 spId;  /* Service provider id */`
	`  U8 dis;  /* Initial link state (enabled/disabled) JMK */`
	`  U8 portType;  /* Port type - T1/E1 or Serial - TEK */`
	`  U8 traceData;  /* Trace data flag */`
	`  U8 spare3;  /* End structure on even 4-byte boundardy */`
	`} MTP3LinkCfg;`
	Refer to the Details section for field descriptions.

linkNo	Link number to assign to this data link. Valid range is 0 through the maximum number of links minus 1 (configured with **Mtp3SetGenCfg**).

Argument	Description
linkType	Type of link. Valid values are: MTP3LNK_ANSI MTP3LNK_ANSI88 MTP3LNK_CCITT MTP3LNK_JNTT MTP3LNK_JTTC
dpc	Point code of the remote end point of this link.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_LNKTYPE	Invalid link type.

Details

This function enables an application to initialize an MTP3LinkCfg structure before calling **Mtp3SetLinkCfg** to set the data link configuration parameters. **Mtp3InitLinkCfg** sets the link number, link type, and destination point code as specified, and initializes all other fields in the MTP3LinkCfg structure to their defaults.

Typically a pointer to an MTP3LinkCfg structure is passed to **Mtp3InitLinkCfg** where default values are set. After initialization, an application can override zero to all of these values and then pass the pointer to **Mtp3SetLinkCfg**, which sets configuration parameters. The default values provided by **Mtp3InitLinkCfg** are adequate for most applications.

For each link, a single transmit queue delivers data to the next lower layer (MTP 2). This queue builds up if the application is sending faster than MTP 2 can transmit data over the link. The pxQlen fields represent the queue length thresholds at which the internal transmit congestion priority is raised to the next level. For example, the default for p1Qlen is 32. This value means that when the transmit queue size reaches 32 messages, the internal transmit congestion priority is raised from 0 to 1. 0 is the lowest priority; 3 is the highest. The congestion priority in combination with the discard priority (discPrior field) are used to determine what to do with new messages from an application when the stack is backed up. If a new message has a priority less than the congestion priority and less than the discard priority, the message is discarded. Otherwise, MTP 3 attempts to queue the message for transmission to MTP 2, assuming buffer space exists. A discard priority of 0 means that all messages attempt to be queued regardless of the internal congestion priority.

The following table lists the MTP3LinkCfg fields that can be modified. A field name in **bold** indicates that subsequent calls to **Mtp3SetLinkCfg** with the same linkNo can update the field. Fields that do not appear in the following table are either unused or for internal use, and must not be modified.

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>lnkSetId</td><td>Linkset number of which this link is a member. Valid range is 1 through 32. Default is 1.</td></tr>
<tr><td>opc</td><td>Originating point code. This point code can be set for multiple OPC emulation. By default, it is set to zero and indicates to the MTP task that the general OPC must be used instead.</td></tr>
<tr><td>adjDpc</td><td>Adjacent destination point code. Must be specified with the dpc parameter.</td></tr>
<tr><td>lnkPrior</td><td>Link priority within the linkset.
It is crucial that priorities are not skipped. A single link must always be zero. Two links in the same linkset can be assigned priorities 0 and 0 or 0 and 1 but not 0 and 2 or 1 and 2. Default value is 0, the highest priority.</td></tr>
<tr><td>msgSize</td><td>Maximum message length across this link. Default value is 272 octets (recommended).</td></tr>
<tr><td>msgPrior</td><td>Management message priority on this link. Default value is 3 (highest).</td></tr>
<tr><td>lnkType</td><td>Protocol variant used on this link. Must be specified with the linkType parameter.</td></tr>
<tr><td>maxSLTtry</td><td>Maximum times to retry the SLT message before disabling the link. Default value is 2. Set this to 0 for infinite SLT retries.</td></tr>
<tr><td>hsBits</td><td>High speed link bits valid values:
<table>
<tr><th>Bit</th><th>Value</th><th>Description</th></tr>
<tr><td>HS_LINK</td><td>0x01</td><td>High speed link is in effect.</td></tr>
<tr><td>HS_EXT_SEQ</td><td>0x02</td><td>Extended sequence numbers are in effect.</td></tr>
</table>
The following table describes these high speed link bits:
<table>
<tr><th>HSL bit</th><th>Description</th></tr>
<tr><td>HS_LINK</td><td>Setting this bit to TRUE, notifies MTP that high speed links are in effect and automatically sets HS_EXT_SEQ to TRUE.
Set HS_EXT_SEQ to FALSE for high speed links with normal sequence numbers.</td></tr>
<tr><td>HS_EXT_SEQ</td><td>Setting this bit to TRUE, notifies MTP that extended sequence numbers are in effect and changes the size of:
• FSN, BSN, and LI fields in MTP 2 packets
• The last FSN field of COO and COA messages at layer 3
Sequence numbers increase from 7 to 12 bits and the length indicator increases from 6 to 8 bits.
Setting HS_EXT_SEQ to TRUE automatically sets HS_LINK to TRUE. Normal speed links with extended sequence numbers are not supported.</td></tr>
</table>
A combination of high and normal speed links is not supported.</td></tr>
<tr><td>p0Qlen</td><td>Transmit queue length threshold at which the congestion priority is raised to 0. Default value is 16.</td></tr>
</table>

Field	Description
p1Qlen	Transmit queue length threshold at which the congestion priority is raised to 1. Default value is 32.
p2Qlen	Transmit queue length threshold at which the congestion priority is raised to 2. Default value is 64.
p3Qlen	Transmit queue length threshold at which the congestion priority is raised to 3.
discPrior	Discard priority. Default value is 0. Refer to the Details section for more information.
lnkTstSLC	Link selection code for link test. This code must match the configured value at the adjacent SP. Default value is spId (linkNo) minus1.
tstLen	Link test pattern length. Default value is 3.
tst	Link test pattern.
ssf	Subservice field. Default based on ***lnkType***: MTP3SSF_NAT = LNK_ANSI or LNK_ANSI88. MTP3SSF_INTL = LNK_CCITT, LNK_JNTT, or LNK_JTTC. MTP3SSF_RESERVE = Never defaults to this value. MTP3SSF_SPARE = Never defaults to this value.
tmr	Timer configuration structure. Refer to the *MTP3LinkTimerCfg structure* on page 75.
dpcLen	DPC and OPC length. Defaults based on ***lnkType***: 24 = LNK_ANSI or LNK_ANSI88. 14 = LNK_CCITT. 16 = LNK_JNTT or LNK_JTTC.
dis	Initial link state. Non-zero is disabled, zero is enabled. If disabled, the link will not attempt to align after the board loads and therefore manual enabling will be required. Default value is 0 (enabled).
portType	Physical port type. Valid values are: PORT_TYPE_REMOTE = Link is physically present on the redundant board PORT_TYPE_TDM = T1/E1/H.100/H.110 links
traceData	TRUE starts tracing of all data between MTP 2 and MTP 3 on this link. Default value is FALSE.

MTP3LinkTimerCfg structure

```
typedef struct _MTP3LinkTimerCfg
{
  TimerCfg t1;   /* t1 - delay to avoid missequencing on      */
                 /* changeover                                */
  TimerCfg t2;   /* t2 - waiting for changeover ack           */
  TimerCfg t3;   /* t3 - delay to avoid missequencing on      */
                 /* changeback                                */
  TimerCfg t4;   /* t4 - waiting for first changeback ack     */
  TimerCfg t5;   /* t5 - waiting for second changeback ack    */
  TimerCfg t6;   /* t6 - delay to avoid missequencing on      */
                 /* rerouting                                 */
  TimerCfg t7;   /* t7 - waiting for link connection ack      */
  TimerCfg t11;  /* t11 - transfer restricted timer           */
  TimerCfg t12;  /* t12 - waiting for uninhibit ack           */
  TimerCfg t13;  /* t13 - waiting for forced uninhibit        */
  TimerCfg t14;  /* t14 - waiting for inhibition ack          */
  TimerCfg t17;  /* t17 - delay to avoid oscillation of       */
                 /* initial alignment failure                 */
  TimerCfg t22;  /* t22 - local inhibit test timer            */
  TimerCfg t23;  /* t23 - remote inhibit test timer           */
  TimerCfg t24;  /* t24 - stabilizing timer                   */
  TimerCfg t31;  /* t31 - BSN requested timer                 */
  TimerCfg t32;  /* t32 - SLT timer                           */
  TimerCfg t33;  /* t33 - connecting timer                    */
  TimerCfg t34;  /* t34 - periodic signalling link test timer */
  TimerCfg t40;  /* Redundancy bind timer                     */
  TimerCfg t41;  /* Redundancy disconnect request timer       */
  TimerCfg t42;  /* Redundancy flow control request timer     */
  TimerCfg t43;  /* Redundancy local processor status timer   */
  TimerCfg t44;  /* Redundancy unbind timer                   */
} MTP3LinkTimerCfg;
```

The following table lists the MTP3LinkTimerCfg fields that can be modified. Unless otherwise specified, the timer names correspond to the CCITT specification. Subsequent calls to **Mtp3SetLinkCfg** with the same linkNo can update any field in this table. Valid values for all fields in this table range from 1 through 65535 tenths of a second.

Field	Description
t1	Delay to avoid an out-of-sequence condition on changeover. Default value is 10.
t2	Waiting for changeover acknowledgement. Default value is 10.
t3	Time controlled diversion: delay to avoid an out-of-sequence condition on changeback. Default value is 10.
t4	Waiting for first changeback acknowledgement. Default value is 10.
t5	Waiting for second changeback acknowledgement. Default value is 10.
t6	Delay to avoid an out-of-sequence condition on controlled rerouting. Default value is 10.
t7	Waiting for data link connection acknowledgement. Default value is 20.
t11	Transfer restricted timer. Default value is 600.
t12	Waiting for uninhibit acknowledgement. Default value is 12.
t13	Waiting for forced uninhibit. Default value is 10.
t14	Waiting for inhibition acknowledgement. Default value is 30.
t17	Delay to avoid oscillation of initial alignment failure. Default value is 10.

Field	Description
t22	Local inhibit test timer (ANSI timer T20). Default value is 1100.
t23	Remote inhibit test timer (ANSI timer T21). Default value is 1100.
t24	Stabilizing timer (ANSI timer not available). Default value is 40.
t31	BSN requested timer (internal timer, not applicable to ANSI or CCITT specifications). Default value is 50.
t32	SLT timer (internal timer, not applicable to ANSI or CCITT specifications). Default value is 100.
t33	Connecting timer (internal timer, not applicable to ANSI or CCITT specifications). Default value is 200.
t34	Periodic signaling link test timer (internal timer, not applicable to ANSI or CCITT specifications). Default value is 600.
t40	Time to wait for a bind confirm from MTP 2 before sending another bind request. Default value is 30.
t41	Time to wait for a disconnect confirm from MTP 2 before sending another disconnect request. Default value is 30.
t42	Time to wait for a flow control confirm from MTP 2 before sending another flow control request. Default value is 30.
t43	Time to wait for a status confirm from MTP 2 before sending another status request. Default value is 30.
t44	Time to wait for an unbind confirm from MTP 2 before sending another unbind request. Default value is 30.

See also

Mtp3MgmtInit, **Mtp3SetGenCfg**, **Mtp3SetLinkCfg**

Mtp3InitLinkSetCfg

Initializes the provided linkset configuration structure MTP3LinkSetCfg with default values, the specified linkset number, and the destination point code.

Prototype

MTP3_STATUS **Mtp3InitLinkSetCfg** (MTP3LinkSetCfg ****pLinkSetCfg***, S16 ***linkSetNo***, U32 ***dpc***)

Argument	Description

pLinkSetCfg	Pointer to the MTP3LinkSetCfg structure to be initialized:
	`typedef struct _MTP3LnkSetCfg`
	`/* MTP Level 3 linkset configuration */`
	`{`
	`U16 lnkSetId; /* Linkset ID */`
	`U16 spare1; /* Alignment */`
	`U32 opc; /* Originating point code */`
	`U32 adjDpc; /* Adjacent destination point code */`
	`Bool loadShar; /* Load sharing indication between links */`
	`U8 spare2; /* Alignment */`
	`U16 minNmbActLnk; /* MAXIMUM number of active links */`
	`U16 spare3; /* Alignment */`
	`U16 nmbCmbLnkSet; /* Number of combined linksets */`
	`MTP3CmbLnkSet cmbLnkSet[MTP3MAXCMBLNK]; /* Combined linksets */`
	`} MTP3LinkSetCfg;`
	Refer to the Details section for field descriptions.

linkSetNo	Number to assign to this linkset. Valid range is 1 through the maximum number of linksets defined in the call to **Mtp3SetGenCfg**.
dpc	Destination point code associated with this linkset.

Return values

MTP3_SUCCESS

Details

This function enables an application to initialize an MTP3LinkSetCfg structure before calling **Mtp3SetLinkSetCfg** to set the linkset configuration parameters. **Mtp3InitLinkSetCfg** sets the linkset number and destination point code as specified, and initializes all other fields in the MTP3LinkSetCfg structure to the defaults.

A pointer to an MTP3LinkSetCfg structure is passed to **Mtp3InitLinkSetCfg** where default values are set. After initialization, an application can override zero to all of these values and then pass the pointer to **Mtp3SetLinkSetCfg**, which sets the configuration. The default values provided by **Mtp3InitLinkSetCfg** are adequate for most applications.

The following table lists the MTP3LinkSetCfg fields that can be modified. A field name in **bold** indicates that subsequent calls to **Mtp3SetLinkSetCfg** with the same lnkSetId (lnkSetNo) can update the field. Fields that do not appear in the following table are either unused or for internal use, and must not be modified.

Field	Description
lnkSetId	Linkset number. Must be specified with the ***lnkSetNo*** parameter.
opc	Originating point code. This field defaults to zero and indicates to the MTP task that it must use the general OPC instead. It can be overridden for multiple OPC emulation.
adjDpc	Adjacent destination point code of all links in this linkset. Must be specified with the ***dpc*** parameter. Valid range is 0 through 0xFFFFFF.
minNmbActLnk	Target number of links to keep active at any given time. Valid range is 0 through 16. Default is 16.
nmbCmbLnkSet	Number of combined linksets. Valid range is 0 through 256. Default is 0.
cmbLnkSet	Array of combined linkset structures. The entire structure contains zeros by default. Refer to the MTP3CmbLnkSet structure.

MTP3CmbLnkSet structure

```
typedef struct _Mtp3CmbLnkSet
                    /* MTP Level 3 combined linkset cfg  */
{
  U16 cmbLnkSetId;   /* Combined linkset ID               */
  U8 lnkSetPrior;    /* Linkset priority                  */
  U8 spare1;         /* Alignment                         */
} MTP3CmbLnkSet;
```

The following table lists the MTP3CmbLnkSet fields that can be modified. Both fields in this table can be updated on subsequent calls to **Mtp3SetLinkSetCfg** with the same lnkSetId (lnkSetNo).

Field	Description
cmbLnkSetId	Combined linkset number (route number). Valid range is 1 through 256.
lnkSetPrior	Priority of this linkset relative to other link sets containing this route. Valid range is 0 through 3. Priorities must not be skipped. A single linkset must always be zero. For example, two linksets containing the same route can be assigned priorities 0 and 0 or 0 and 1 but not 0 and 2 or 1 and 2. Default value is 0, the highest priority.

See also

Mtp3MgmtInit, **Mtp3SetLinkCfg**, **Mtp3SetLinkSetCfg**

Mtp3InitNSapCfg

Initializes the provided NSAP configuration structure MTP3NSapCfg with default values and the specified link type.

Prototype

MTP3_STATUS **Mtp3InitNSapCfg** (MTP3NSapCfg ***pNSapCfg***, U8 ***linkType***)

Argument	Description

pNSapCfg	Pointer to the MTP3NSapCfg structure to be initialized:
	`typedef struct _MTP3NSapCfg`
	`{`
	`  S16 p0QLen;  /* Size of the priority 0 Q */`
	`  S16 p1QLen;  /* Size of the priority 1 Q */`
	`  S16 p2QLen;  /* Size of the priority 2 Q */`
	`  S16 p3QLen;  /* Size of the priority 3 Q */`
	`  U8  discPrior;  /* Discard priority */`
	`  U8  lnkType;  /* Link type ANSI or CCITT */`
	`  U8  selector;  /* Upper layer selector */`
	`  U8  spare1;  /* Alignment */`
	`  MemoryId mem;  /* Memory region and pool id */`
	`  U8  dpcLen;  /* dpc or opc length */`
	`  U8  prior;  /* Priority */`
	`  U8  route;  /* Route */`
	`  U8  spare2;  /* Alignment */`
	`  U32 opc;  /* Default opc for UDatReq from user layer */`
	`} MTP3NSapCfg;`
	See the Details section for field descriptions.

linkType	Type of link. Valid values are: MTP3LNK_ANSI MTP3LNK_ANSI88 MTP3LNK_CCITT MTP3LNK_JNTT MTP3LNK_JTTC

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_LNKTYPE	Invalid link type.

Details

This function enables an application to initialize an MTP3NSapCfg structure before calling **Mtp3SetNSapCfg** to set the network SAP configuration parameters. **Mtp3InitNSapCfg** sets the link type as specified and initializes all other fields in the MTP3NSapCfg structure to their defaults.

The p***x***Qlen fields represent the queue length thresholds at which the internal congestion priority is raised to the next level. For example, the default for p1Qlen is 16. This value means that when the receive queue size reaches 16 messages, the internal congestion priority is raised from 0 to 1. Zero (0) is the lowest priority and 3 is the highest priority. The congestion priority in combination with the discard priority (discPrior field) are used to determine what to do with newly received messages when flow control is on.

If a newly received message has a priority less than the congestion priority and less than the discard priority, the message is discarded. Otherwise, MTP 3 attempts to queue the message, assuming buffer space exists. A discard priority of 0 means that all messages will attempt to be queued regardless of the internal congestion priority. The congestion priority is also used to notify adjacent signaling points of the level of congestion being experienced by this node so that they can re-route future messages or take other corrective actions.

Typically a pointer to an MTP3NSapCfg structure is passed to **Mtp3InitNSapCfg** where default values are set. After initialization, an application can override zero to all of these values and then pass the pointer to **Mtp3SetNSapCfg**, which sets the configuration. The default values provided by **Mtp3InitNSapCfg** are adequate for most applications.

The following table lists the MTP3NSapCfg fields that can be modified. All the fields in this table can be updated on subsequent calls to **Mtp3SetNSapCfg**. Fields that do not appear in the table are either unused or for internal use, and must not be modified.

Field	Description
p0Qlen	Receive queue length threshold at which the congestion priority is raised to 0. Valid range is 2 through 1024. Default is 0.
p1Qlen	Receive queue length threshold at which the congestion priority is raised to 1. Valid range is (p0Qlen + 2) through 1024. Default is 512.
p2Qlen	Receive queue length threshold at which the congestion priority is raised to 2. Valid range is (p1Qlen + 2) through 1024. Default is 768.
p3Qlen	Receive queue length threshold at which the congestion priority is raised to 3. Valid range is (p2Qlen + 2) through 1024. Default is 896.
discPrior	Discard priority. Valid range is 0 through 3. Default is 0. See the Details section for more information.
lnkType	Protocol variant used by this NSAP. Must be specified with the ***linkType*** parameter. Valid values are: MTP3LNK_ANSI MTP3LNK_ANSI88 MTP3LNK_CCITT MTP3LNK_JNTT MTP3LNK_JTTC
dpcLen	DPC and OPC length. Defaults based on ***linkType***: 24 = LNK_ANSI or LNK_ANSI88 14 = LNK_CCITT 16 = LNK_JNTT or LNK_JTTC

Mtp3InitRouteCfg

Initializes the provided route configuration structure MTP3RouteCfg with default values, the specified route number, the DPC, and the switch type.

Prototype

MTP3_STATUS **Mtp3InitRouteCfg** (MTP3RouteCfg ***pRouteCfg***, S16 ***routeNo***, U32 ***dpc***, U8 ***swType***)

Argument	Description

pRouteCfg	Pointer to the MTP3RouteCfg structure to be initialized:
	`typedef struct _MTP3RoutCfg`
	`/* MTP Level 3 routing configuration */`
	`{`
	`U32 dpc; /* Destination point code */`
	`U32 opc; /* Originating point code */`
	`U8 spType; /* Signaling point type */`
	`U8 swtchType; /* Switch type */`
	`U16 cmbLnkSetId; /* Combined link set ID */`
	`Bool loadShar; /* Load sharing indication between */`
	`/* linksets */`
	`U8 dir; /* Direction */`
	`Bool rteToAdjSp; /* Flag indicating this route to */`
	`/* adjacent SP */`
	`Bool rteToCluster; /* Flag indicating this route to */`
	`/* a cluster */`
	`U8 ssf; /* Sub service field */`
	`U8 spare1; /* Alignment */`
	`U16 spare2; /* Alignment */`
	`MTP3RteTimerCfg tmr; /* Route timer configuration */`
	`} MTP3RouteCfg;`
	See the Details section for field descriptions.

routeNo	Number to assign to this route. Valid range is 0 through the maximum number of routes defined in the call to **Mtp3SetGenCfg**.
dpc	Destination point code associated with this route.
swType	Switch type of the target signaling point. Must be one of the following types: MTP3LNK_ANSI MTP3LNK_CCITT MTP3LNK_JNTT MTP3LNK_JTTC

Return values

MTP3_SUCCESS

Details

This function enables an application to initialize an MTP3RouteCfg structure before calling **Mtp3SetRouteCfg** to set the route configuration parameters. **Mtp3InitRouteCfg** sets the linkset number, dpc, and switch type as specified, and initializes all other fields in the MTP3RouteCfg structure to their defaults.

A pointer to an MTP3RouteCfg structure is passed to **Mtp3InitRouteCfg** where default values are set. After initialization, an application can override zero to all of these values and then pass the pointer to **Mtp3SetRouteCfg**, which sets the

configuration. The default values provided by **Mtp3InitRouteCfg** are adequate for most applications.

The following table lists the MTP3RouteCfg fields that can be modified. A field name in **bold** indicates that subsequent calls to **Mtp3SetRouteCfg** with the same dpc or cmbLnkSetId can update the field. Fields that do not appear in the table are either unused or for internal use, and must not be modified.

Field	Description
dpc	Destination point code. Must be specified with the ***dpc*** parameter. Valid values are 0 through 0xFFFFFF.
opc	Originating point code. This field defaults to zero and indicates to the MTP task that it must use the general OPC instead. It can be overridden for multiple OPC emulation.
spType	Signaling point type of the destination of this route. Valid values are: MTP3TYPE_SP MTP3TYPE_STP (default)
swType	Switch type of the destination of this route. Must be specified with the ***swType*** parameter. Valid values are: MTP3LNK_ANSI MTP3LNK_CCITT MTP3LNK_JNTT MTP3LNK_JTTC
cmbLnkSetId	Route ID. Must be specified with the ***routeNo*** parameter. Valid values are 1 through 256.
dir	Route direction. Valid values are: MTP3_RTE_UP (inbound) MTP3_RTE_DN (outbound)
MTP3RteTimerCfg	Route timer configuration. Refer to the *MTP3RteTimerCfg structure* on page 83.
rteToAdjSp	Flag indicating whether or not this route is to an adjacent signaling point. Valid values are True or False.
rteToCluster	Flag indicating whether this route is to a cluster (True) or a single signaling point (False). This value affects only STP nodes.
ssf	Subservice field. Used to determine what to put in the ssf field of MTP 3 originated management messages. Default based on ***swType***: MTP3SSF_NAT = LNK_ANSI or LNK_ANSI88. MTP3SSF_INTL = LNK_CCITT, LNK_JNTT, or LNK_JTTC.

MTP3RteTimerCfg structure

```
typedef struct _MTP3RteTimerCfg
{
  TimerCfg t8;       /* t8 - transfer prohibited inhibition    */
                     /* timer                                  */
  TimerCfg t10;      /* t10 - waiting to repeat route set test */
   /* ITU restart timers */
  TimerCfg t19_itu;  /* ITU only - time after sending a TRA to */
                     /* ignore rcvd TRA's                      */
  TimerCfg t21_itu;  /* ITU only - overall restart time at     */
                     /* adjacent SP/STP                        */
   /* ANSI restart timers */
  TimerCfg t25_ansi; /* ANSI only - time at adjacent SP        */
                     /* waiting for TRA                        */
  TimerCfg t28_ansi; /* ANSI only - time at adjacent SP        */
                     /* waiting for TRW                        */
  TimerCfg t29_ansi; /* ANSI only - timer started when TRA     */
                     /* sent in response to unexpected TRW     */
                     /* or TRA                                 */
  TimerCfg t30_ansi; /* ANSI only - timer to limit sending     */
                     /* of TFP's and TFR's in response to      */
                     /* unexpected TRW or TRA                  */
} MTP3RteTimerCfg;
```

The following table lists the MTP3RteTimerCfg fields that can be modified. Unless otherwise specified, the timer names correspond to the CCITT specification. Subsequent calls to **Mtp3SetRouteCfg** with the same dpc or cmbLnkSetId can update any field in this table. Valid values for all fields in this table range from 1 through 65535 tenths of a second.

Field	Default	Description
t8	1	Transfer prohibited inhibition timer.
t10	45	Timer waiting to repeat route set test message.
t19_itu	680	ITU restart timer to avoid ping-pong of TFP, TFR, or TRA messages.
t21_itu	640	Overall ITU restart timer at adjacent SP.
t25_ansi	320	ANSI restart timer at adjacent SP waiting for a TRA message.
t28_ansi	300	ANSI restart timer at adjacent SP waiting for a TRW message.
t29_ansi	630	ANSI restart timer started when a TRA is sent in response to an unexpected TRA or TRW.
t30_ansi	320	ANSI restart timer to limit sending of TFPs and TFRs in response to unexpected TRA or TRW.

See also

Mtp3MgmtInit, **Mtp3SetGenCfg**, **Mtp3SetRouteCfg**

Mtp3LinkSetStats

Obtains (and potentially resets) linkset statistical information about the specified linkset number.

Prototype

MTP3_STATUS **Mtp3LinkSetStats** (U8 ***board***, S16 ***linkSetNo***, MTP3LinkSetStats ****pStats***, BOOL ***bReset***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through ***maxboard*** (currently 8).
linkSetNo	Linkset number about which to obtain status information. Must have been previously defined with **Mtp3SetLinkSetCfg**.

pStats	Pointer to a buffer provided by the application where the requested statistical information is returned:
	`typedef struct _Mtp3LinkSetStats /* MTP Level 3 link set statistics */`
	`{`
	`  U32  strtLnkSetFail;  /* Start of linkset failure */`
	`  U32  stopLnkSetFail;  /* Stop of linkset failure */`
	`  U32  durLnkSetUnav;  /* Duration of linkset unavailability */`
	`} MTP3LinkSetStats;`

bReset	Optionally resets statistics.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_HANDLE	**Mtp3MgmtInit** not called for the specified board.

Details

Linkset statistics include the number of times a linkset has failed and recovered and the total duration of linkset unavailability.

If the ***bReset*** value is a non-zero integer, the statistics are reset after returning the current values. A value of zero disables the reset function.

See also

Mtp3MgmtInit, **Mtp3SetLinkSetCfg**

Mtp3LinkSetStatus

Obtains linkset status information about the specified linkset number.

Prototype

MTP3_STATUS **Mtp3LinkSetStatus** (U8 ***board***, S16 ***linkSetNo***, MTP3LinkSetStatus ****pStatus***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through ***maxboard*** (currently 8).
linkSetNo	Linkset number about which to obtain status information. Must have been previously defined with **Mtp3SetLinkSetCfg**.

pStatus	Pointer to a buffer provided by the application where the requested status information is returned:
	`typedef struct _Mtp3LinkSetStatus /* MTP Level 3 - link set status */`
	`{`
	`  Bool cfgFlg; /* Configured flag */`
	`  U8 spare1; /* Alignment */`
	`  U16 nmbActLnks; /* Number of active links */`
	`  U16 nmbCongLnks; /* Number of congested links */`
	`  Bool cong; /* Linkset is congested */`
	`  U8 state; /* Linkset state */`
	`} Mtp3LinkSetStatus;`
	See the Details section for field descriptions.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_HANDLE	**Mtp3MgmtInit** not called for the specified board.

Details

A linkset is a group of links that, once defined as a linkset, can be controlled or queried as a group. This function obtains linkset status information about the specified linkset number, including the linkset state, and the number of active and congested links.

The following table lists the Mtp3LinkSetStatus fields:

Field	Description
nmbActLinks	Number of active links in this linkset. Valid values are 0 through minNmbActLnk.
nmbCongLnks	Number of congested links in this linkset. Valid values are 0 through number of links in linkset.
cong	Linkset congestion state. Valid values are: True = linkset is congested. False = linkset is not congested.
state	Linkset state. Valid values are: 0 = linkset is active. 1 = linkset is disabled.

See also

Mtp3MgmtInit, **Mtp3SetLinkSetCfg**

Mtp3LinkStats

Obtains and potentially resets data link statistical information about the specified link number.

Prototype

MTP3_STATUS **Mtp3LinkStats** (U8 ***board***, S16 ***linkNo***, MTP3LinkStats ****pStats***, BOOL ***bReset***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through ***maxboard*** (currently 8).
linkNo	Link number for which statistics are obtained. Must have been previously defined with **Mtp3SetLinkCfg**.
pStats	Pointer to a buffer provided by the application where the requested statistical information is returned. Refer to the *MTP3LinkStats structure* on page 88.
bReset	Optionally resets statistics.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_HANDLE	**Mtp3MgmtInit** not called for the specified board.

Details

If the ***bReset*** value is a non-zero integer, the statistics are reset after returning the current values. A value of zero disables the reset function.

Link statistics include counts for the various message types and their acknowledgments both transmitted and received.

MTP3LinkStats structure

```
typedef struct _Mtp3LinkStats   /* MTP Level 3 link statistics   */
{
  U32  changeOverTx;      /* Changeover order transmitted       */
  U32  changeOverRx;      /* Changeover order received          */
  U32  changeOverAckTx;   /* Changeover ack transmitted         */
  U32  changeOverAckRx;   /* Changeover ack transmitted         */
  U32  changeBackTx;      /* Changeback declaration             */
                          /*   transmitted                      */
  U32  changeBackRx;      /* Changeback declaration received    */
  U32  changeBackAckTx;   /* Changeback ack transmitted         */
  U32  changeBackAckRx;   /* Changeback ack received            */
  U32  emChangeOverTx;    /* Emergency changeover transmitted   */
  U32  emChangeOverRx;    /* Emergency changeover received      */
  U32  emChangeOverAckTx; /* Emergency changeover ack           */
                          /*   transmitted                      */
  U32  emChangeOverAckRx; /* Emergency changeover ack received  */
  U32  lnkInhTx;          /* Link inhibit transmitted           */
  U32  lnkInhRx;          /* Lnk inhibit received               */
  U32  lnkUninhTx;        /* Lnk uninhibit transmitted          */
  U32  lnkUninhRx;        /* Link uninhibit received            */
  U32  lnkInhAckTx;       /* Link inhibited ack transmitted     */
  U32  lnkInhAckRx;       /* Link inhibited ack received        */
  U32  lnkUninhAckTx;     /* Link uninhibited ack transmitted   */
  U32  lnkUninhAckRx;     /* Link uninhibited ack received      */
  U32  lnkInhDenTx;       /* Link inhibit denied transmitted    */
  U32  lnkInhDenRx;       /* Link inhibit denied received       */
  U32  lnkForceUninhTx;   /* Force link uninhibit transmitted   */
  U32  lnkForceUninhRx;   /* Force link uninhibit received      */
  U32  lnkLocInhTstTx;    /* Local link inhibit test            */
                          /*   transmitted                      */
  U32  lnkLocInhTstRx;    /* Local link inhibit test received   */
  U32  lnkRmtInhTstTx;    /* Remote link inhibit test           */
                          /*   transmitted                      */
  U32  lnkRmtInhTstRx;    /* Remote link inhibit test received  */
  U32  lnkConOrdTx;       /* Link connection order transmitted  */
  U32  lnkConOrdRx;       /* Link connection order received     */
  U32  lnkConAckTx;       /* Link connection ack transmitted    */
  U32  lnkConAckRx;       /* Link connection ack received       */
  U32  txCntrlRx;         /* Transfer controlled received       */
  U32  txCntrlTx;         /* Transfer controlled transmitted    */
  U32  lnkTstRx;          /* Link test received                 */
  U32  lnkTstTx;          /* Link test transmitted              */
  U32  lnkTstAckRx;       /* Link test ack received             */
  U32  lnkTstAckTx;       /* Link test ack transmitted          */
  U32  txDrop;            /* Transmit messages dropped          */
  U32  txCongDrop;        /* MSUs dropped due to link congestion */
  U32  sifOctTx;          /* Number of SIF octets transmitted   */
  U32  sifOctRx;          /* Number of SIF octets received      */
  U32  sioOctTx;          /* Number of SIO octets transmitted   */
  U32  sioOctRx;          /* Number of SIO octets received      */
  U32  msuTx;             /* Number of MSU transmitted          */
  U32  msuRx;             /* Number of MSU received             */
  U32  cong0;             /* Link congestion at threshold 0     */
  U32  cong1;             /* Link congestion at threshold 1     */
  U32  cong2;             /* Link congestion at threshold 2     */
  U32  cong3;             /* Link congestion at threshold 3     */
  U16  tqCnt;             /* Message count in transmit queue    */
  U16  rtqCnt;            /* Message count in retransmit queue  */
  U16  hiTqCnt;           /* High water message count in transmit*/
                          /*   queue                            */
  U16  hiRtqCnt;          /* High water message count in the    */
                          /*   retransmit queue                 */
  U32  durLnkUnav;        /* Duration of link unavailability    */
  U32  durLnkCong;        /* Duration of link congestion        */
} MTP3LinkStats;
```

See also

Mtp3MgmtInit, **Mtp3SetLinkSetCfg**

Mtp3LinkStatus

Obtains data link status information about the specified link number, including the link state, flow control state, queue sizes, and blocked, inhibited, and congestion status.

Prototype

MTP3_STATUS **Mtp3LinkStatus** (U8 ***board***, S16 ***linkNo***, MTP3LinkStatus *****pStatus***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through ***maxboard*** (currently 8).
linkNo	Link number for which to obtain status information. Must have been previously defined with **Mtp3SetLinkCfg**.

pStatus	Pointer to a buffer provided by the application where the requested status information is returned:
	`typedef struct _MTP3LinkStatus  /* MTP Level 3 - signalling link status */`
	`{`
	`  DateTime DT;  /* Date / timestamp */`
	`  U8 state;  /* Link state */`
	`  U8 flcSt;  /* Flow control state */`
	`  Bool blkd;  /* Link is blocked */`
	`  Bool locBlkd;  /* Link is blocked locally */`
	`  Bool rmtBlkd;  /* Link is blocked remotely */`
	`  Bool inhbt;  /* Link is inhibited */`
	`  Bool locInhbt;  /* Local inhibit indication */`
	`  Bool rmtInhbt;  /* Remote inhibit indication */`
	`  Bool uninhbt;  /* Uninhibit signaling indication */`
	`  Bool cong;  /* Link is congested */`
	`  Bool emerg;  /* Emergency indication */`
	`  U8  spare1;  /* End structures on 4-byte boundaries */`
	`}  MTP3LinkStatus;`
	See the Details section for field descriptions.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_HANDLE	**Mtp3MgmtInit** not called for the specified board.

Details

The following table describes the MTP3LinkStatus fields:

Field	Description
linkState	State of the data link: MTP3_LNK_INACTIVE = Link is inactive. MTP3_LNK_CONN = Link is connecting. MTP3_LNK_ACTIVE = Link is active. Data flow possible. MTP3_LNK_FAILED = Link failed. MTP3_LNK_WAITCON = Link is waiting for a remote connect.
flcSt	Flow control state: True = Flow control for link is on. False = Flow control for link is off.
blkd	Blocking state: True = Link is blocked. False = Link is not blocked.
locBlkd	Local blocking state: True = Link is locally blocked. False = Link is not locally blocked.
rmtBlkd	Remote blocking state: True = Link is remotely blocked. False = Link is not remotely blocked.
inhbt	Inhibited state: True = Link is inhibited. False = Link is not inhibited.
locInhbt	Local inhibited state: True = Link is locally inhibited. False = Link is not locally inhibited.
rmtInhbt	Remote inhibited state: True = Link is remotely inhibited. False = Link is not remotely inhibited.
uninhbt	Uninhibit signaling indication: True = Uninhibit signal received. False = Uninhibit signal not received.
cong	Congestion state: True = Link is congested. False = Link is not congested.
emerg	Emergency indication: True = Emergency indication present. False = Emergency indication not present.

See also

Mtp3MgmtInit, **Mtp3SetLinkSetCfg**

Mtp3MgmtCtrl

Sends a control request to the MTP 3 task.

Prototype

MTP3_STATUS **Mtp3MgmtCtrl** (U8 ***board***, S16 ***entity***, U8 ***action***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through ***maxboard*** (currently 8).
entity	Link number for link control operations or the linkset number for linkset control operations. The link or linkset number must have been previously defined with the appropriate configuration function call, either **Mtp3SetLinkCfg** or **Mtp3SetLinkSetCfg**. Not used for general control operations.
action	Action to take on the specified entity. See the **Details** section for valid actions.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_HANDLE	**Mtp3MgmtInit** not called for the specified board.
MTP3_PARAM	Invalid action. Must be an action defined in the following General control actions table.

Details

Three classifications of control requests exist: general, link, and linkset. Use this function to activate and deactivate network layer resources. The combination of ***entity*** and ***action*** tells MTP 3 what action to take and what to act upon.

The ***action*** field must be one of the following:

General control actions (***entity*** is not used)

Action	Description
MTP3_CTRL_ALARM_ENA	Enables alarms.
MTP3_CTRL_ALARM_DIS	Disables alarms.
MTP3_CTRL_FLOWCTL_ON	Starts flow control and disables transmission of all except critical MTP 3 management messages.
MTP3_CTRL_FLOWCTL_OFF	Ends flow control and enables transmission of all MTP 3 management messages.
MTP3_CTRL_TRACE_ON	Enables all data tracing in MTP 3.
MTP3_CTRL_TRACE_OFF	Disables all data tracing in MTP 3.

Link control actions (***entity*** specifies link number)

Action	Description
MTP3_CTRL_LINK_ENA	Enables the specified link.
MTP3_CTRL_LINK_DIS	Disables the specified link.
MTP3_CTRL_LINK_INH	Inhibits transmission of data messages to and from specified link.
MTP3_CTRL_LINK_UNINH	Uninhibits transmission of data messages to and from specified link.
MTP3_CTRL_LINK_DEL	Deletes the specified link.
MTP3_CTRL_LINK_LPO	Sends specified link into local processor outage condition. Layer 2 sends SIPOs rather than FISUs.
MTP3_CTRL_LINK_LPR	Clears local processor outage condition. Layer 2 resumes sending FISUs.
MTP3_CTRL_LINK_EMG	Sends specified link into emergency alignment condition. Layer 2 sends SIE rather than SIN.
MTP3_CTRL_LINK_NRM	Sends specified link into normal alignment condition. Layer 2 sends SIN rather than SIE.
MTP3_CTRL_LINK_TRCON	Enables data tracing on specified link.
MTP3_CTRL_LINK_TRCOFF	Disables data tracing on specified link.

Linkset control actions (***entity*** specifies linkset number)

Action	Description
MTP3_CTRL_LINKSET_ENA	Enables all links in specified linkset.
MTP3_CTRL_LINKSET_DIS	Disables all links in specified linkset.

See also

Mtp3MgmtInit, **Mtp3SetLinkCfg**, **Mtp3SetLinkSetCfg**

Mtp3MgmtInit

Initializes internal structures and opens communication with the MTP 3 process on the TX board.

Prototype

MTP3_STATUS **Mtp3MgmtInit** (U8 ***board***, U8 ***srcEnt***, U8 ***srcInst***)

Argument	Description
board	TX board number with which to communicate. Valid range is 1 through ***maxboard*** (currently 8).
srcEnt	Source entity ID.
srcInst	Source instance ID.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_DRIVER	Low level driver returned an error trying to open the specified dual port RAM channel.

Details

Call this function before calling any other management functions. The source entity ID must be from 0x20 through 0x31 and unique for each application accessing the MTP 3 (or other) task.

Note: An application that opens the management and data functions must use different entity IDs for **Mtp3GenStatus** calls.

See also

Mtp3MgmtTerm

Mtp3MgmtTerm

Terminates communication with the MTP3 task for this application.

Prototype

MTP3_STATUS **Mtp3MgmtTerm** (U8 ***board***)

Argument	Description
board	TX board number with which to terminate communication. Valid range is 1 through ***maxboard*** (currently 8).

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_HANDLE	**Mtp3MgmtInit** not called for the specified board.

Details

Call this function to free up resources when an application terminates or finishes communication with the MTP 3 task.

See also

Mtp3GenStatus, **Mtp3MgmtInit**

Mtp3NSapStatus

Obtains network service access point status information from the MTP 3 task, including the NSAP state (bound or unbound) and the flow control state (on or off).

Prototype

MTP3_STATUS **Mtp3NSapStatus** (U8 ***board***, S16 ***nsapNo***, MTP3NSapStatus ****pStatus***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. This value must be from 1 through ***maxboard*** (currently 8).
nsapNo	NSAP number for which to obtain status information. Must have been previously defined with **Mtp3SetNSapCfg**.

pStatus	Pointer to a buffer provided by the application where the requested status information is returned:
	`typedef struct _Mtp3NSapStatus`
	`/* MTP Level 3 - SAP status */`
	`{`
	`U8 sapState; /* SAP state */`
	`U8 flcSt; /* Flow control state */`
	`U8 spare1; /* Alignment */`
	`U8 spare2; /* Alignment */`
	`U32 congDiscard; /* Messages discarded due to congestion */`
	`} MTP3NSapStatus;`
	See the Details section for field descriptions.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_HANDLE	**Mtp3MgmtInit** not called for the specified board.

Details

The following table describes the Mtp3NSapStatus fields:

Field	Description
sapState	State of the service access point. Valid values: SN_UNBND = No application has bound to this SAP. SN_BND = An application has bound to this SAP.
flcSt	Flow control state. Valid values: TRUE = Flow control is on. FALSE = Flow control is off.
congDiscard	Count of inbound messages discarded due to congestion. For more information, see *Controlling congestion* on page 31.

See also

Mtp3MgmtInit, **Mtp3SetNSapCfg**

Mtp3RouteStats

Obtains and potentially resets statistics for the route associated with the specified destination point code.

Prototype

MTP3_STATUS **Mtp3RouteStats** (U8 ***board***, U32 ***dpc***, U32 ***opc***, MTP3RouteStats ****pStats***, BOOL ***bReset***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through ***maxboard*** (currently 8).
dpc	Destination point code associated with the desired route.
opc	Originating point code associated with the desired route. If set to zero, MTP will use the OPC specified in the general configuration.

pStats	Pointer to a buffer provided by the application where the requested statistical information is returned:
	`typedef struct _Mtp3RouteStats /* MTP Level 3 route statistics */`
	`{`
	`U32 dpc; /* Destination point code of this route */`
	`U32 opc; /* Originating point code of this route */`
	`U32 routeTstTx; /* Route set test transmitted */`
	`U32 routeTstRx; /* Route set test received */`
	`U32 congTstRx; /* Route set congestion test received */`
	`U32 congTstTx; /* Route set congestion test transmitted */`
	`U32 txProhibTx; /* Transfer prohibited transmitted */`
	`U32 txProhibRx; /* Transfer prohibited received */`
	`U32 txRestrictRx; /* Transfer restricted received */`
	`U32 txAllowTx; /* Transfer allowed transmitted */`
	`U32 txAllowRx; /* Transfer allowed received */`
	`U32 rteUnavCnt; /* Route unavailable */`
	`U32 sifOctTx; /* Number of SIF octets transmitted */`
	`U32 sioOctTx; /* Number of SIO octets transmitted */`
	`U32 durRteUnav; /* Duration of route unavailability */`
	`} MTP3RouteStats;`

bReset	If true (non-zero), the statistics are reset (set to zero) after returning the current values. If false (zero), no reset is performed.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_HANDLE	**Mtp3MgmtInit** not called for the specified board.

Details

Route statistics include transmit and receive counts for messages such as route test, route congestion test, transfer prohibited, transfer restricted, and transfer allowed, as well as the number of times a route was unavailable and the total duration of route unavailability. All counts are U32 fields.

See also

Mtp3MgmtInit

Mtp3RouteStatus

Obtains routing status information about the route associated with the specified destination point code.

Prototype

MTP3_STATUS **Mtp3RouteStatus** (U8 ***board***, U32 ***dpc***, U32 ***opc***, MTP3RouteStatus ****pStatus***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through ***maxboard*** (currently 8).
dpc	Destination point code associated with the desired route.
opc	Originating point code associated with the desired route. If set to zero, MTP will use the OPC specified in the general configuration.

pStatus	Pointer to a buffer provided by the application where the requested status information is returned:
	`typedef struct _Mtp3RouteStatus /* MTP Level 3 - route status */`
	`{`
	`  U32 dpc; /* Destination point code */`
	`  U32 opc; /* Originating point code */`
	`  U16 nmbActvLnkSets; /* Number of active link sets */`
	`  U16 nmbCongLnkSets; /* Number of congested link sets */`
	`  U8 state; /* Route state */`
	`  Bool cong; /* Route set is congested */`
	`  Bool rstFlg; /* SP restarting flag */`
	`} MTP3RouteStatus;`
	See the Details section for field descriptions.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_HANDLE	**Mtp3MgmtInit** not called for the specified board.

Details

The following table describes the Mtp3RouteStatus fields:

Field	Description
dpc	Destination point code associated with this route. Valid values are 0 through 0xFFFFFF.
opc	Originating point code associated with this route. Valid values are 0 through 0xFFFFFF.
nmbActvLnkSets	Number of active link sets. Valid values are 0 through 32.
nmbCongLnkSets	Number of congested link sets. Valid values are 0 through 32.
state	Route state: 0 = Route unavailable. 1 = Route available.
cong	Route congestion state: True = Route congested. False = Route not congested.
rstFlg	Adjacent signaling point restarting flag: True = Adjacent signaling point is restarting. False = Adjacent signaling point is not restarting.

See also

Mtp3MgmtInit

Mtp3RouteTest

Initiates a route test by transmitting an SRT message (JNTT and JTTC only).

Prototype

MTP3_STATUS **Mtp3RouteTest** (U8 ***board***, U32 ***dpc***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through ***maxboard*** (currently 8).
dpc	Destination point code associated with the desired route.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_HANDLE	**Mtp3MgmtInit** not called for the specified board.

See also

Mtp3GetRouteCfg, **Mtp3InitRouteCfg**, **Mtp3SetRouteCfg**

Mtp3SetGenCfg

Sets the general configuration values in the MTP 3 task.

Prototype

MTP3_STATUS **Mtp3SetGenCfg** (U8 ***board***, MTP3GenCfg ****pGenCfg***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through ***maxboard*** (currently 8).
pGenCfg	Pointer to the MTP3GenCfg structure containing the general configuration values to set.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_HANDLE	**Mtp3MgmtInit** not called for the specified board.
MTP3_TIMEOUT	No response from the board.

Details

An application must set the field values in the MTP3GenCfg structure before calling this function. Set the values by calling **Mtp3InitGenCfg**, by setting each field from within the application, or by a combination of the two (calling **Mtp3InitGenCfg** and then overriding specific fields before passing the MTP3GenCfg structure to this function).

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

See also

Mtp3InitGenCfg, **Mtp3MgmtInit**

Mtp3SetLinkCfg

Configures the MTP 3 task with the data link configuration values contained in the provided MTP3LinkCfg structure.

Prototype

MTP3_STATUS **Mtp3SetLinkCfg** (U8 ***board***, MTP3LinkCfg ****pLinkCfg***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through ***maxboard*** (currently 8).
pLinkCfg	Pointer to the MTP3LinkCfg structure containing the data link configuration values.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_HANDLE	**Mtp3MgmtInit** not called for the specified board.
MTP3_TIMEOUT	No response from the board.

Details

An application must set the field values in the MTP3LinkCfg structure before calling this function. Set the values by calling **Mtp3InitLinkCfg**, by setting each field from within the application, or with a combination of the two (calling **Mtp3InitLinkCfg** and then overriding specific fields before passing the MTP3LinkCfg structure to this function).

This function is typically called once for each configured link.

See also

Mtp3InitLinkCfg, **Mtp3MgmtInit**

Mtp3SetLinkSetCfg

Configures the MTP 3 task with the linkset configuration values contained in the provided MTP3LinkSetCfg structure.

Prototype

MTP3_STATUS **Mtp3SetLinkSetCfg** (U8 ***board***, MTP3LinkSetCfg ****pLinkSetCfg***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through ***maxboard*** (currently 8).
pLinkSetCfg	Pointer to the MTP3LinkSetCfg structure containing the linkset configuration values.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_HANDLE	**Mtp3MgmtInit** not called for the specified board.
MTP3_TIMEOUT	No response from the board.

Details

An application must set the field values in the MTP3LinkSetCfg structure before calling this function. Set the values by calling **Mtp3InitLinkSetCfg**, by setting each field from within the application, or with a combination of the two (calling **Mtp3InitLinkSetCfg** and then overriding specific fields before passing the MTP3LinkSetCfg structure to this function).

This function is typically called once for each configured linkset.

See also

Mtp3InitLinkSetCfg, **Mtp3MgmtInit**

Mtp3SetNSapCfg

Configures the MTP 3 task with the network SAP configuration values contained in the provided MTP3NSapCfg structure.

Prototype

MTP3_STATUS **Mtp3SetNSapCfg** (U8 ***board***, MTP3NSapCfg ****pNSapCfg***, S16 ***nSapNo***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through ***maxboard*** (currently 8).
pNSapCfg	Pointer to the MTP3NSapCfg structure containing the network SAP configuration values.
nSapNo	Network SAP number to assign to this NSAP. Valid range is 0 through the maximum number of NSAPs minus 1, as specified in **Mtp3SetGenCfg**.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_HANDLE	**Mtp3MgmtInit** not called for the specified board.
MTP3_TIMEOUT	No response from the board.

Details

An application must set the field values in the MTP3NSapCfg structure before calling this function. Set the values by calling **Mtp3InitNSapCfg**, by setting each field from within the application, or with a combination of the two (calling **Mtp3InitNSapCfg** and overriding specific fields before passing the MTP3NSapCfg structure to this function).

This function is typically called once for each configured NSAP.

See also

Mtp3InitNSapCfg, **Mtp3MgmtInit**, **Mtp3SetGenCfg**

Mtp3SetRouteCfg

Configures the MTP 3 task with the route configuration values contained in the provided MTP3RouteCfg structure.

Prototype

MTP3_STATUS **Mtp3SetRouteCfg** (U8 ***board***, MTP3RouteCfg ****pRouteCfg***)

Argument	Description
board	TX board number on which the desired MTP 3 task resides. Valid range is 1 through ***maxboard*** (currently 8).
pRouteCfg	Pointer to the MTP3RouteCfg structure containing the route configuration values.

Return values

Return value	Description
MTP3_SUCCESS	
MTP3_BOARD	Invalid board number.
MTP3_HANDLE	**Mtp3MgmtInit** not called for the specified board.
MTP3_TIMEOUT	No response from the board.

Details

An application must set the field values in the MTP3RouteCfg structure before calling this function. Set the values by calling **Mtp3InitRouteCfg**, by setting each field from within the application, or with a combination of the two (calling **Mtp3InitRouteCfg** and overriding specific fields before passing the MTP3RouteCfg structure to this function).

This function is typically called once for each configured route.

See also

Mtp3InitRouteCfg, **Mtp3MgmtInit**

8 mtpmgr utility

mtpmgr overview

mtpmgr is an SS7 MTP status utility. Use *mtpmgr w*hen monitoring or troubleshooting SS7 signaling to check the status of:

- SS7 signaling links.
- SS7 linksets.
- SS7 routes.
- An MTP task including whether it is primary or backup.

Procedure

Complete the following steps to use *mtpmgr*:

Step	Action

1	Start *mtpmgr* by entering:
	`mtpmgr`
	The following information displays:
	`mtpmgr: sample MTP3 management application version 7.2 Jul 15 2002`
	`mtpmgr[1]>`

2	Enter *mtpmgr* commands at the command prompt.
3	End an *mtpmgr* session by typing **q** and pressing **Enter**.

mtpmgr commands

The following table describes the available *mtpmgr* commands:

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td>help or ?</td><td>Displays a list of supported commands. To display the command syntax, precede the command with either help or ?. For example:
<code>Mtpmgr[1]> ? status</code>
Usage:
<code>status entity [number | *]</code>
where:
<table><tr><th>Argument</th><th>Valid values</th></tr><tr><td><entity></td><td>LINK | LINKSET | NSAP | ROUTE | MTP3</td></tr><tr><td><number></td><td>Not used for STATUS MTP3.
Use an asterisk (*) only with status link.</td></tr></table></td></tr>
<tr><td>status</td><td>Displays status information about links, linksets, routes, NSAPs, and MTP. The status information includes current configuration values.
<table><tr><th>Argument</th><th>Valid values</th></tr><tr><td>link n</td><td>Shows the current status of link n including link state and whether a link is blocked or inhibited.</td></tr><tr><td>linkset n</td><td>Shows the current status of linkset n including linkset state and congestion status.</td></tr><tr><td>route pointcode</td><td>Shows the current status of the route to the signaling point specified by pointcode. The signaling points can be specified in dotted notation (for example, 0.0.2), as a decimal number (for example, 2), or as a hexadecimal number (for example, 0x2). Information includes the route state and congestion status.</td></tr><tr><td>nsap</td><td>Shows the current status of an NSAP (upper layer interface) including whether or not an application is bound to MTP through the NSAP.</td></tr><tr><td>mtp3</td><td>Shows general status information for the MTP 3 layer including whether MTP3 is primary, backup, or standalone.</td></tr></table>For more information, see *status command examples* on page 111.</td></tr>
</table>

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td>stats</td><td>The status information displays and optionally clears current statistics for the MTP layer or on a link, link set, or route. For example:
<code>stats object [reset]</code>
where object is one of the following arguments:
<table><tr><th>Argument</th><th>Valid values</th></tr><tr><td>link n</td><td>Lists current statistics for link n.</td></tr><tr><td>linkset n</td><td>Lists current statistics for linkset n.</td></tr><tr><td>route pointcode</td><td>Lists the current statistics for the route to the signaling point specified by pointcode. The signaling point can be specified in dotted notation (for example, 0.0.2), as a decimal number (for example, 2), or as a hexadecimal number (for example, 0x2).</td></tr><tr><td>mtp3</td><td>Lists general statistics for the MTP 3 layer.</td></tr><tr><td>reset</td><td>Optional. Sets all statistics for the specified object to zero (0) immediately after the current statistics are displayed.</td></tr></table>
For more information, see stats command examples on page 114.</td></tr>
<tr><td>link</td><td>Performs link control functions such as enabling and disabling links, inhibiting and uninhibiting links, and enabling and disabling tracing.
To display the link control options, type:
<code>? link</code>
There is no mtpmgr output for the control operations. To control behavior of a specified link, enter:
<code>link n 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 link n.</td></tr><tr><td>dis</td><td>Disables link n.</td></tr><tr><td>inh</td><td>Inhibits link n.</td></tr><tr><td>uni</td><td>Uninhibits link n.</td></tr><tr><td>del</td><td>Deletes link n.</td></tr><tr><td>lpo</td><td>Forces link n into local processor outage condition (typically only used during conformance testing).</td></tr><tr><td>lpr</td><td>Restores link n from local processor outage condition.</td></tr><tr><td>emg</td><td>Forces link n to use emergency alignment procedure.</td></tr><tr><td>nrm</td><td>Forces link n to use normal alignment procedure.</td></tr><tr><td>tre</td><td>Enables packet tracing on link n. For more information, see the TX Utilities Manual.</td></tr><tr><td>trd</td><td>Disables packet tracing on link n. For more information, see the TX Utilities Manual.</td></tr></table></td></tr>
</table>

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td>linkset</td><td>Performs link set control functions such as enabling or disabling entire link sets.
Enables or disables the specified link set. For example:
<code>linkset n 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 linkset n.</td></tr><tr><td>dis</td><td>Disables linkset n.</td></tr></table></td></tr>
<tr><td>trace</td><td>Enables or disables packet tracing on all configured links. For example:
<code>trace [on | off]</code>
where:
<table><tr><th>Argument</th><th>Valid values</th></tr><tr><td>on</td><td>Enables packet tracing on all links.</td></tr><tr><td>off</td><td>Disables packet tracing on all links.</td></tr></table>For more information, see the TX Utilities Manual.</td></tr>
<tr><td>srt</td><td>Initiates a signaling route test on a specified point code. This command is valid only for the JNTT and JTTC protocol variants.
<code>srt pointcode</code>
where pointcode can be specified in dotted notation (for example, 0.0.2), as a decimal number (for example, 2), or as a hexadecimal number (for example, 0x2). The upper byte can also be used to select the link or linkset over which to send the SRT. Refer to MTP3RouteTest on page 100 for full details.</td></tr>
<tr><td>Q</td><td>Quits the mtpmgr application.</td></tr>
</table>

status command examples

This topic provides examples of the following status commands:

- status link command
- status linkset command
- status route command
- status mtp3 command

status link command

The status link * command displays an overview of all the links in the system as shown in the following sample:

```
status link *
Num    Name   MTP3 State     MTP2 Hi State   Low State           Block   Inhbt
---                          -----
0      T1     ACTIVE          ENABLED         IN_SERVICE
1      R      ACTIVE          REMOTE          REMOTE
```

Use this command to quickly determine if all links have come up after the board is loaded. In this sample, the first link is physically on the board where the query was performed. The second link is on the redundant mate board in the other signaling server. The Block and Inhbt fields would show L, R, or both L and R if a link is locally or remotely blocked or inhibited.

The status link ***link_number*** command displays more detailed information for a specified link, including the current configuration settings. Use this command to determine if configuration values were set as expected. MTP3 values are displayed first. Press **Enter** to bring up the MTP2 values. For remote links, MTP2 values are not displayed. They can be displayed by running *mtpmgr* on the other signaling server in the node.

```
mtpmgr[1]>status link 0

MTP3 Board 1, Link 0:
  State  = ACTIVE

---- Current Link (0) Layer 3 configuration ----
bandwidth= LSL        seqNum    = Normal     opc       = 0.0.2 (0x2)
lnkSetId = 1          lnkType   = ANSI       adjDpc    = 0.0.1 (0x1)
ssf      = NATL       dpcLen    = 24         lnkPrior  = 0
msgSize  = 272        msgPrior  = 3          maxSLTtry = 2
p0QLen   = 16         p1QLen    = 32         p2QLen    = 64
p3QLen   = 128        discPrior = 0          disable   = FALSE
lnkTstSLC= 0          tstString = 54 53 54
         ---- Link Timers (in tenths/sec) ----
t1       = 10         t2        = 10         t3        = 10
t4       = 10         t5        = 10         t6        = 10
t7       = 20         t11       = 600        t12       = 10
t13      = 10         t14       = 30         t17       = 10
t22      = 1100       t23       = 1100       t24       = 40
t31      = 10         t32       = 100        t33       = 30
t34      = 600        t40       = 30         t41       = 30
t42      = 30         t43       = 30         t44       = 30

Press [Enter] for MTP2 status
Board 1, Link 0 MTP2 Status:

High lvl state = ENABLED    Frames out    = 0     Frames dropped = 0
Low  lvl state = IN_SERVICE

---- Current Link (0) Layer 2 configuration ----

Link Name  = T1         bandwidth   = LSL        seqNum      = Normal
maxFrmLen  = 272        congDisc    = FALSE
errType    = NORMAL     lssuLen     = 2          SUERMD rate = 256
SUERM Thrsh= 64         AERM Thrsh N= 4          AERM Thrsh E= 1
Max RTB Msg= 127        Max RTB Oct = 34544      Max Prov Abt= 5

IsoThresh  = 1000       TxqThresh1  = 50         TxqThresh1A = 20
TxqThresh2 = 200        TxqThresh2A = 100        SapThresh   = 500
SapThreshA = 100        IdleFreq    = 0          RtFreq      = 0


       ----- Layer 2 Timers (in tenths/sec) -----
L2 t1      = 130        L2 t2       = 115        L2 t3       = 115
L2 t4 norm = 23         L2 t4 emrg  = 6          L2 t5       = 1
L2 t6      = 60         L2 t7       = 20         L2 t10      = 30
L2 t11     = 20         L2 t12      = 20         L2 t13      = 20
```

status linkset command

The status linkset ***linkset_number*** command displays status information for the specified linkset. Use this command to display the number of active links and the linkset state. The number of active links contains both local and remote links to the same destination.

```
mtpmgr[1]>status linkset 1

Board 1, Linkset 1:
  Active Links: 2
  Congested Links: 0
  State: Active, Not Congested

---- Current Linkset (1) configuration ----
                          opc          = 0.0.2 (0x2)
lnkSetId     = 1          adjDpc       = 0.0.1 (0x1)
minNmbActLnk = 2          nmbCmbLnkSet = 1
      ---- Combined Linksets ----
cmbLnkSetId  = 1          lnkSetPrior  = 0
```

status route command

The status route ***point_code*** command displays status information for the specified destination point code. Use this command to check that the configuration values are set as expected. Other useful information includes the number of active linksets and the route state. You cannot get status information for an up route associated with incoming messages.

```
mtpmgr[1]>status route 0.0.1

Board 1, Route DPC 0.0.1 (0x1), OPC 0.0.2 (0x2):
  Active Linksets: 1
  Congested Linksets: 0
  State: Available, Not Congested

---- Current Route (0x1, 0x2) configuration ----
RouteId     = 1         spType       = STP        dpc      = 0.0.1 (0x1)
swtchType   = ANSI      direction    = DOWN       opc      = 0.0.2 (0x2)
rteToAdjSp  = TRUE      rteToCluster = FALSE      ssf      = NATL
              ---- Route Timers (in tenths/sec) ----
t8          = 10        t10          = 450
t19_itu     = 680       t21_itu      = 640
t25_ansi    = 320       t28_ansi     = 300       t29_ansi     = 630
```

status mtp3 command

The status mtp3 command shows general configuration information and the high-availability state of the server. Use this command to check that the configuration values are set as expected. The HA State value returned by this command indicates whether the server is the primary or backup in a redundant configuration or running in standalone mode.

```
mtpmgr[1]>status mtp3

Board 1 SP State: Normal          HA State: STAND_ALONE

---------------- Current general configuration ---------------
typeSP      = SP          PcDispFmt    = DEFAULT
spCode1     = 0.0.2 (0x2)
spCode2     = 0.0.2 (0x2)
rstReq      = FALSE       nmbLinks     = 2          nmbNSap      = 2
nmbLnkSets = 2           nmbRouts     = 64         nmbRteInst   = 32
nmbMasks    = 1
  Mask[0 ] = ffffffff

    ----- Signalling Point Timers (in tenths/sec) -----
t15         = 30        t16          = 20         tRteInst     = 18000
t18_itu     = 300       t20_itu      = 600
t22_ansi    = 300       t23_ansi     = 300        t24_ansi     = 300
t26_ansi    = 130       t27_ansi     = 30
```

stats command examples

The following code samples show examples of displayed statistics in the following scenarios:

- mtpmgr[1]> stats link 0
- mtpmgr[1]> stats linkset 1
- mtpmgr[1]> stats route 0.0.2
- mtpmgr[1]> stats mtp3

stats link command

The stats link ***n*** command lists message counts and other statistics for link ***n***. The first page shows MTP3 level statistics.

```
mtpmgr[1]> stats link 0

Board 1, Link 0 MTP3 Stats:
Msg:       Rx        Tx          Msg:       Rx        Tx
----       --        --          ----       --        --
COO        0         0           COA        0         0
CBD        0         0           CBA        0         0
ECO        0         0           ECA        0         0
LIN        0         0           LIA        0         0
LUN        0         0           LUA        0         0
LID        0         0           LFU        0         0
LLI        0         0           LRI        0         0
DLC        0         0           CSS        0         0
SLTM       0         0           SLTA       0         0
TFC        0         0
MSU        0         0
SIF        0         0           Octets
SIO        0         0           Octets
MSUs Dropped:      Congestion     0    Total      0
Transmit Queue:       Current     0     High      0
Retransmit Queue:     Current     0     High      0
Link Unavailable:              0 secs   Link Congested:           0 secs
```

Press **Enter** to display MTP2 statistics:

```
Board 1, Link 0 MTP2 Stats:
inService  = 11166769  lclBusy     = 0         slFailAll   = 5
slFailAb   = 0         slFailAck   = 0         slFailErr   = 5
slFailCong = 0         slFailAlign = 6         slNSUErr    = 0
slNegAck   = 0         nRetrans    = 0
Msg:       Rx        Tx          Msg:       Rx        Tx
----                                    --
SIF/SIO    12        13          MSU        72135
LSSU*      56        68          FISU*      196680    71598
* LSSU & FISU counts do not include those filtered by the firmware
             Current    High
             -------    ----
Tx Queue:      0          1
Rtb Queue:     0          4
lTx Queue:     0          5
Sap Queue:     0          0
```

stats linkset command

The stats linkset ***n*** command lists current statistics for linkset ***n***.

```
mtpmgr[1]> stats linkset 1

Board 1, Linkset 1 Stats:
Linkset Failure  Start: 0  End: 0
Linkset Unavailable Duration: 0
```

stats route command

The stats route ***pointcode*** command lists the current statistics for the route to the signaling point specified by ***pointcode***.

```
mtpmgr[1]> stats route 0.0.2

Board 1, Route 0.0.2 (0x2) Stats:
Msg:       Rx        Tx          Msg:       Rx        Tx
----       --        --          ----       --        --
RST        0         0           RCT        0         0
TFP        0         0           TFA        0         0
TFC        0
SIF Octets Tx: 0
SIO Octets Tx: 0
Route Unavailable Count:  0
```

stats mtp3 command

The stats mtp3 command lists general statistics for the MTP 3 layer.

```
mtpmgr[1]> stats mtp3

Board 1 Stats
Msg:       Rx        Tx
----       --        --
UPU        0         0
TRA        0         0
TRW        0         0
Packets dropped due to routing errors: 0
```

Index