Dialogic
Making Innovation Thrive

Dialogic® NaturalAccess™ SCCP Layer Developer's Reference Manual

July 2009 64-0461-01

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.

Windows is a registered trademark of Microsoft Corporation in the United States and/or other countries. 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
B.2.3	July 1997	SCCP release B.2.3
B.3.0	January 1998	SCCP release B.3.0
B.3.1	July 1998	GJG
9000-6467-25	September 1998	GJG
9000-6467-26	March 1999	GJG
9000-6467-27	March 2000	GJG, SS7 3.5 beta
9000-6467-28	November 2000	GJG, SS7 3.6
9000-6467-29	August 2001	GJG, SS7 3.8 beta
9000-6467-30	February 2002	MVH, SS7 3.8
9000-6467-31	November 2003	SRG, SS7 4.0
9000-6467-32	July 2006	LBZ, SS7 4.3
64-0461-01	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

Chapter 1: Introduction 7

Chapter 2: SS7 overview 9
SS7 architecture 9
SCCP task 10

Chapter 3: SCCP programming model 11
Programming model overview 11
SCCP service users 11
Entity and instance IDs 12
NMS SCCP functions 13
Service functions 13
Management functions 14
Queues and contexts 15
Single-context, single-queue model 16
Multiple-context, single-queue model 17
Multiple-context, multiple-queue model 18
Signaling parameters 19
Signaling point and subsystem status procedures 20
Coordinated state change 20
Subsystem state changes 21
Remote signaling point failures 21
SCCP configuration 23
SCCP addressing and routing 24
Routing by point code and subsystem number 25
Routing by global title 25
Global title translation 26
Setting the fields in a TCAP request 26
Setting the SCCP configuration 27
Setting variations in global title translation 27
Connection IDs 29
Application inactivity control 30
Connection auditing 31

Chapter 4: Using the SCCP service 33
SCCP service overview 33
Setting up the Natural Access environment 33
Initializing the Natural Access environment 33
Creating queues and contexts 34
Binding to the SCCP service 34
Transferring connectionless data 36
Undeliverable connectionless messages 36
Establishing connections 37
Transferring connection-oriented data 38
Resetting connections 39
Clearing connections 39
Handling redundancy events 40
Handling congestion events 40
Outbound congestion 40
Inbound congestion 41

Tracing function calls and events 42

Chapter 5: SCCP service function reference 43
SCCP service function summary 43
Using the SCCP service function reference 44
SCCPConnAuditRqst 45
SCCPConnectResp 46
SCCPConnectRqst 47
SCCPCoordResp 48
SCCPCoordRqst 49
SCCPDAckRqst 50
SCCPDataRqst 51
SCCPEDataRqst 52
SCCPGetStats 53
SCCPInactResp 54
SCCPReleaseRqst 55
SCCPResetResp 56
SCCPResetRqst 57
SCCPRetrieveMessage 58
SCCPStateRqst 60
SCCPUDataRqst 61

Chapter 6: SCCP management function reference 63
SCCP management function summary 63
Configuration functions 64
Control functions 65
Statistics functions 65
Status function 65
Using the SCCP management function reference 66
SccpAlarmControl 67
SccpAsciiMaskToBcd 68
SccpAsciiNumToBcd 69
SccpBcdMaskToAscii 70
SccpBcdNumToAscii 71
SccpDelAddrCfg 72
SccpDelRteCfg 73
SccpGetAddrCfg 74
SccpGetGenCfg 75
SccpGetGenStats 76
SccpGetNSapCfg 78
SccpGetNSapStats 79
SccpGetRteCfg 81
SccpGetUSapCfg 82
SccpGetUSapStats 83
SccpGetUSapStatus 85
SccpInitAddrCfg 87
SccpInitGenCfg 91
SccpInitNSapCfg 96
SccpInitRteCfg 99
SccpInitUSapCfg 102
SccpMgmtInit 106
SccpMgmtTerm 107
SccpSetAddrCfg 108

SccpSetGenCfg 109
SccpSetNSapCfg 110
SccpSetRteCfg 112
SccpSetUSapCfg 113
SccpTraceControl 114

Chapter 7: Demonstration programs and utilities 115
Summary of the demonstration programs and utilities 115
Connectionless messages: sccldemo 115
Connection-oriented messages: sccodemo 118
SCCP configuration utility: sccpcfg 120
SCCP layer status: sccpmgr 121

Chapter 8: Common parameters 123
Parameters overview 123
Data types 123
Address parameter 124
Cause value parameter 127
Credit parameter 130
Data parameter 130
End of sequence parameter 130
Expedited data selection parameter 130
Importance parameter 131
Protocol class parameter 131
Receipt confirmation selection parameter 131
Receive sequence number parameter 131
Subsystem multiplicity indicator parameter 132
Subsystem number parameter 132
Subsystem status indicator parameter 132

Chapter 9: Messages 133
Messages overview 133
Data types 133
Connection information message 133
Connect request message 134
Coordination request message 134
Data acknowledge message 134
Data request message 135
Released message 135
Reset request message 135
Unitdata request message 136

Chapter 10: SCCPRetrieveMessage event reference 137
Event reference overview 137
SCCPCONNAUDCFM 138
SCCPCONNCFM 140
SCCPCONNIND 141
SCCPCOORDCFM 142
SCCPCOORDIND 143
SCCPDACKIND 144
SCCPDATIND and SCCPEDATIND 145
SCCPINACTIND 146
SCCPPCSTIND 147

SCCPRELIND 148
SCCPRESETCFM 149
SCCPRESETIND 150
SCCPRUNSTATEIND 151
SCCPSTAIND 152
SCCPSTATEIND 154
SCCPUDATIND 155

1 Introduction

The *Dialogic® NaturalAccess™ SCCP Layer Developer's Reference Manual* explains how to implement the SS7 Signaling Connection Control Part (SCCP) layer using NMS SCCP. This manual explains how to create applications using NaturalAccess™ SCCP and presents a detailed specification of its signaling procedures and functions.

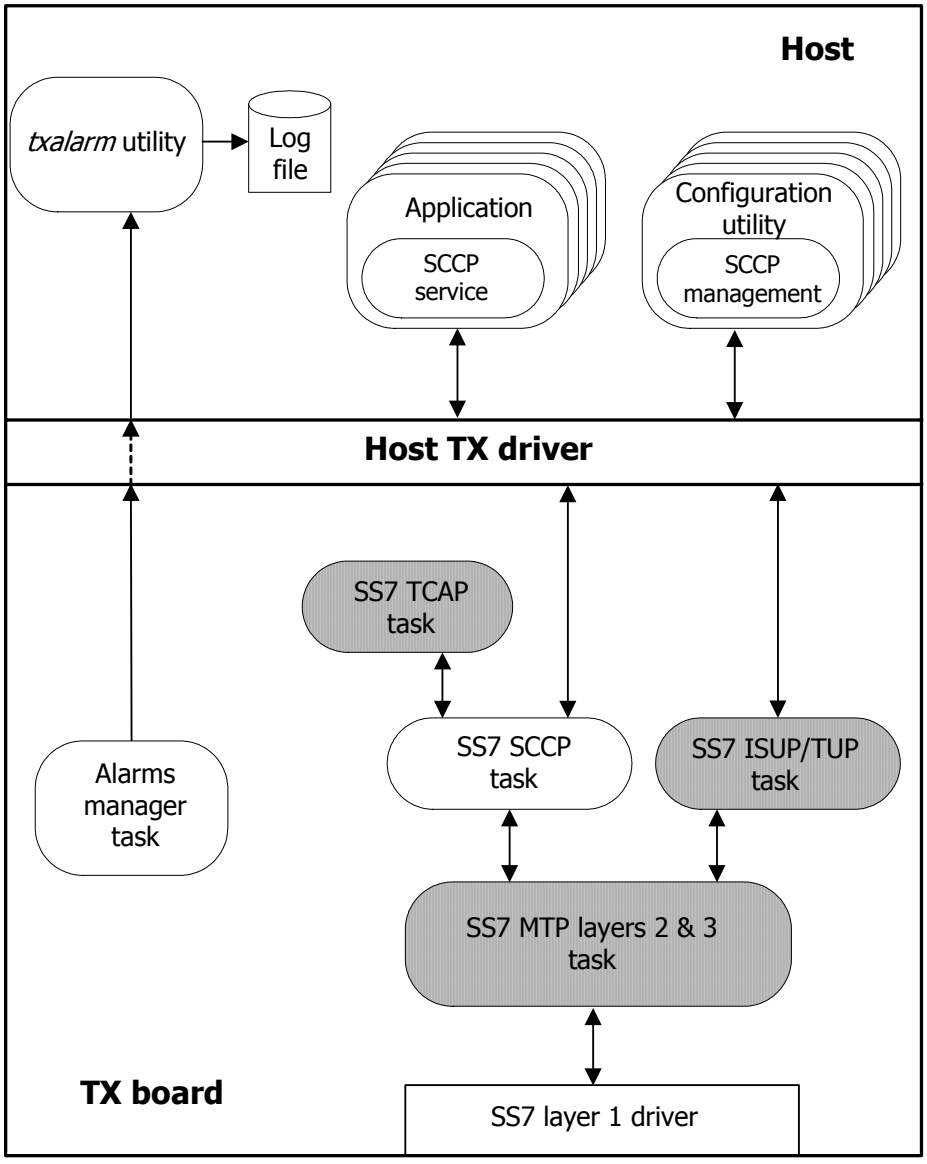
Note: The product(s) to which this document pertains is/are among those sold by NMS Communications Corporation ("NMS") to Dialogic Corporation ("Dialogic") in December 2008. Certain terminology relating to the product(s) has been changed, whereas other terminology has been retained for consistency and 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 SCCP	Dialogic® NaturalAccess™ SCCP Layer

2 SS7 overview

SS7 architecture

The following illustration shows the SS7 software architecture in a typical system with separate host applications handling the data/control (SCCP) interface, system configuration, and system alarms. The system consists of the following components:



The TX board consists of the following components:

- SCCP task that implements the SS7 SCCP layer.
- MTP task that implements the SS7 MTP 2 (data link) layer and the MTP 3 (network) layer.
- Optional ISUP/TUP task that implements the SS7 ISUP/TUP layer.
- Optional TCAP task that implements the SS7 TCAP 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.

The host consists of the following components:

- A TX driver for the native host operating system that provides low-level access to the TX board from the host.
- Functions that provide the application with a high-level interface to the SCCP layer services.
- Functions that provide the application with a high-level interface to the SCCP management layer services.
- An alarm collector process for capturing alarms and saving them to a text file. The alarm collector (*txalarm*) is provided in both executable and source form. The source can be used as an example for developers who want to integrate the TX alarms into their own alarm monitoring system.
- Configuration utilities (one for each SS7 layer) that read the SS7 configuration file(s) and load the configurations to the TX processor tasks at system startup. The SCCP configuration utility (*sccpcfg*) is provided in both executable and source form. The source code can be used as an example for developers who want to integrate the SCCP configuration into their own configuration management system.
- The SCCP manager utility (*sccpmgr*) provides a command line interface from which alarm levels can be set, buffers can be traced, and SCCP statistics can be viewed and reset.

SCCP task

NMS SCCP provides the interface for host applications to access the routing and network transport services of the SS7 SCCP layer. NMS SCCP supports the following capabilities:

- Connectionless service
  Basic (class 0) and sequenced (class 1) connectionless service.
- Connection-oriented service
  Basic (class 2) and flow-control (class 3) connection-oriented services.
- Global title translation
  Translation of global titles into point codes and subsystem numbers (SSNs).
- Routing based on global title, global title and SSN, or destination point code with global title and/or SSN.
- Compliance with ITU-T Q.711 - Q.717 and ANSI T1.112 recommendations.

3 SCCP programming model

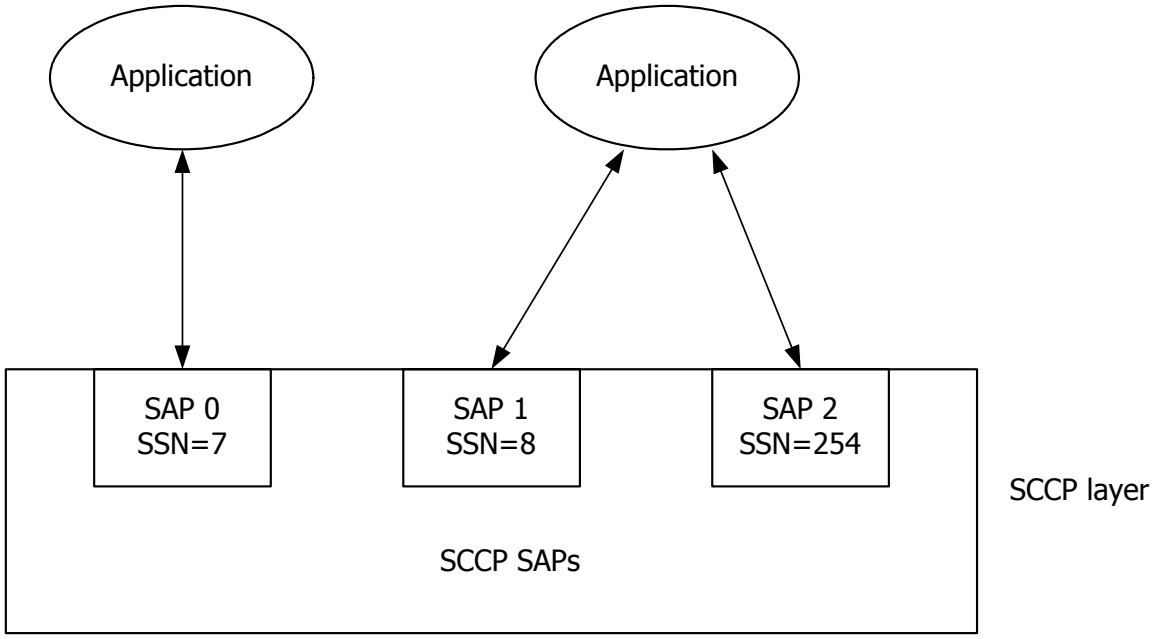
Programming model overview

NMS SCCP consists of a set of functions that provide access to the SCCP layer operations, and a set of events that notify the application of incoming messages, network status, and message delivery errors. NMS SCCP also performs the byte ordering translation, where necessary, between application processor (little endian) byte order and network (big endian) byte order.

NMS SCCP is 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. Understanding the basic Natural Access programming concepts such as services, queues, contexts, and asynchronous events is critical to developing applications that use the SCCP service. Refer to the *Natural Access Developer's Reference Manual* for more information.

SCCP service users

NMS SCCP supports one or more applications with service access points, or SAPs. One SAP is defined for each application that uses the SCCP service. At initialization, an application binds to a particular SAP specifying the SAP ID. Each user SAP is associated with a single SCCP subsystem number. All SCCP messages destined for a particular subsystem number are routed to the application bound to the SAP associated with that subsystem number. SAPs are shown in the following illustration:



An application can implement multiple subsystems by binding to multiple SCCP SAPs, as shown in the previous illustration. If an application supports multiple TX boards, it must bind with each board separately.

Note: The number of SAPs and the characteristics of each SAP are specified at SCCP configuration time. Refer to the *NMS SS7 Configuration Manual* for more information.

Each SAP (and subsystem number) can optionally have a backup point code (a backup node that implements the same subsystem), a set of concerned point codes that are notified by the SCCP task whenever the availability of the subsystem changes, or both.

Entity and instance IDs

Each application must have a unique entity and instance ID for routing messages among the processes in the system. Entity IDs are single byte values in the range of 0x00 - 0xFF, assigned by the application developer. Entity IDs are allocated as follows:

Range	Use
0x00 - 0x1F 0x80 - 0xFF	Reserved for use by system utilities, configuration utilities, and management utilities.
0x20 - 0x7F	Available for use by applications.

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

NMS SCCP functions

NMS SCCP provides two sets of functions:

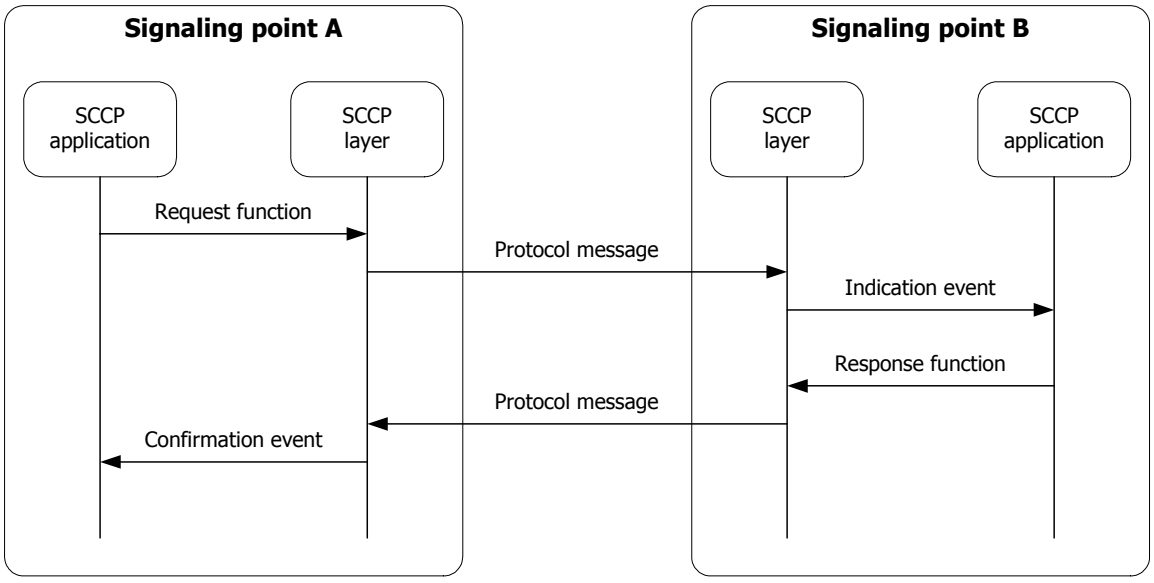
- Service functions
- Management functions

Service functions

The SCCP service functions provide the application access to the SCCP layer services. Applications invoke SCCP services by calling SCCP request functions that send an SCCP message to a remote exchange or signaling point (SP). Request function parameters are converted to messages. The device driver sends these parameters to the SCCP task.

The SCCP requests from the remote signaling points are presented to the application at the receiving side as indications. The receiving application then issues a reply to the originating signaling point by invoking the appropriate SCCP service response function. The response function is typically translated by the SCCP layer into a protocol message back to the originating signaling point. That response is presented back to the application as a confirmation.

This communication model is shown in the following illustration. Some operations, such as sending unit data, include only the request or indication steps. These operations are called unconfirmed operations.



All SCCP service functions are asynchronous. Completion of the function implies only that the function was successfully initiated (a request message was queued to the SCCP task). Errors detected by the SCCP task result in asynchronous status indications being sent to the application. Successfully delivered requests generally result in no notification to the application until the far end takes some corresponding action such as returning a connect confirm message in response to a connection request.

Indication and confirmation messages, as well as status messages from the local SCCP layer, are passed to application processes as asynchronous events. All events for a particular user service access point are delivered through the associated Natural Access queue. For more information about queues, refer to the *Natural Access Developer's Reference Manual*.

Applications detect that an event is pending through an operating system specific mechanism such as **poll** in UNIX or **WaitForMultipleObjects** in Windows. The application retrieves the event data (or message) through a function that also translates the confirmation parameters from SS7 SCCP raw format to API format.

The following table lists the SCCP service connectionless data transfer functions and events:

Request function	Indication event	Response function	Confirmation event
SCCPUDataRqst	SCCPUDATIND	N/A	N/A
N/A	SCCPSTAIND	N/A	N/A

The following table lists the SCCP service connection-oriented data transfer functions and events:

Request function	Indication event	Response function	Confirmation event
SCCPConnectRqst	SCCPCONNIND	**SCCPConnectResp**	SCCPCONNCFM
SCCPDataRqst	SCCPDATIND	N/A	N/A
SCCPEDataRqst	SCCPEDATIND	N/A	N/A
SCCPDAckRqst	SCCPDACKIND	N/A	N/A
SCCPReleaseRqst	SCCPRELIND	N/A	N/A
SCCPResetRqst	SCCPRESETIND	**SCCPResetResp**	SCCPRESETCFM

Management functions

Unlike the SCCP service functions that send and receive messages asynchronously, each SCCP management function generates a request followed immediately by a response from the TX board. SCCP management functions block the calling application waiting for this response (for a maximum of five seconds, but typically a few hundred milliseconds) and return an indication as to whether or not an action completed successfully. For this reason, the SCCP management functions are typically used by one or more management applications. Separate applications use the SCCP service functions. SCCP management is packaged as a separate library with its own interface header files.

The following table lists the SCCP management and status functions and events:

Request function	Indication event	Response function	Confirmation event
SCCPCoordRqst	SCCPCOORDIND	**SCCPCoordResp**	SCCPCOORDCFM
SCCPStateRqst	N/A	N/A	N/A
N/A	SCCPINACTIND	**SCCPInactResp**	N/A
SCCPConnAuditRqst	N/A	N/A	SCCPCONNAUDCFM
N/A	SCCPSTATEIND	N/A	N/A
N/A	SCCPPCSTIND	N/A	N/A
N/A	SCCPRUNSTATEIND	N/A	N/A
SCCPGetStats	N/A	N/A	N/A
N/A	SCCP_EVN_CONGEST	N/A	N/A

Queues and contexts

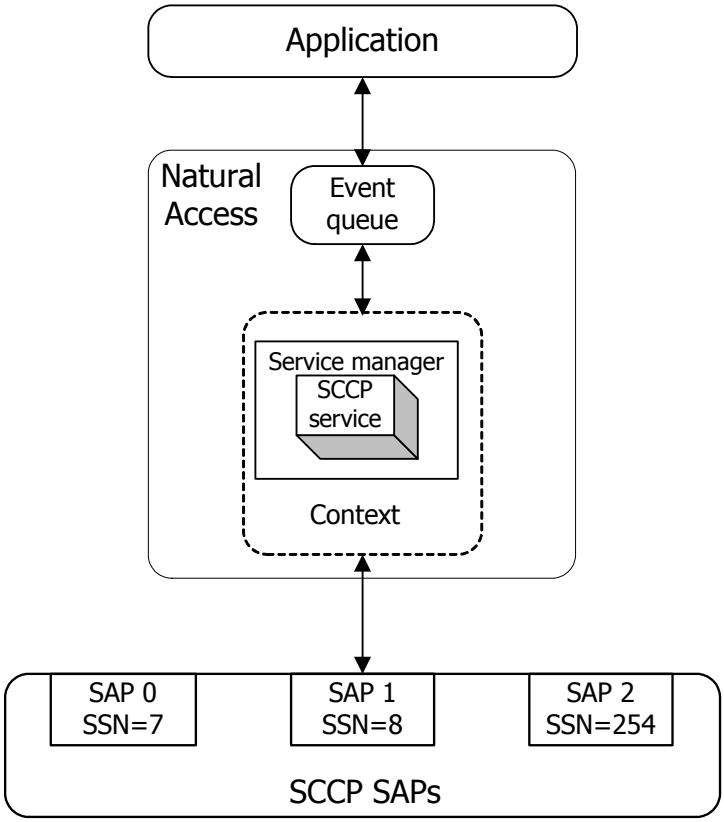
Natural Access organizes services and their associated resources around a processing object known as a context. Each instance of an application binding to an SCCP service access point is a unique Natural Access context. Contexts are created with **ctaCreateContext**.

All events and messages from the SCCP service are delivered to the application through a Natural Access queue object. 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 SCCP service access points (how many SCCP subsystems) are implemented by the application and how the application is organized.

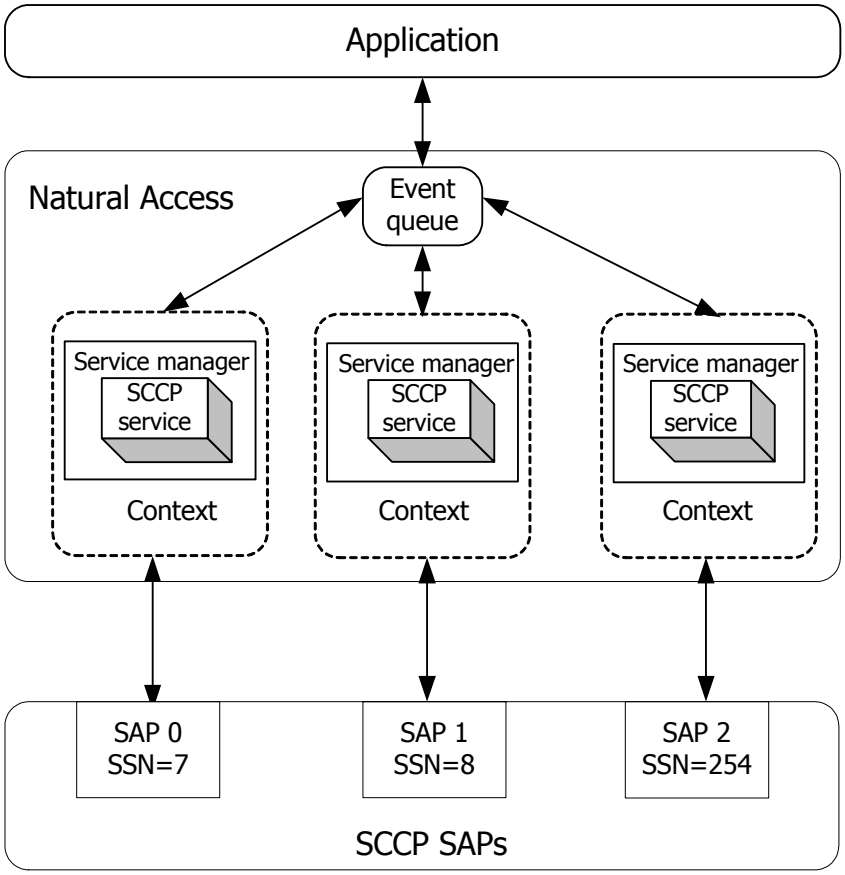
Single-context, single-queue model

An application that uses a single SCCP service access point uses a single-context, single-queue model as shown in the following illustration:



Multiple-context, single-queue model

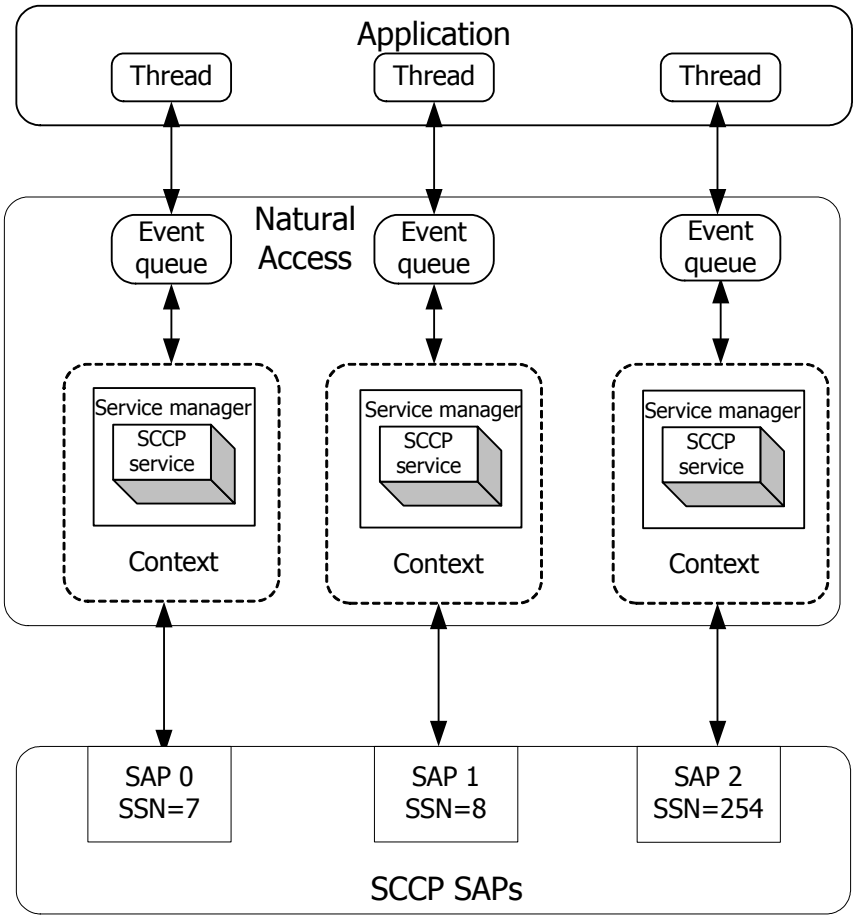
For a single-threaded application that uses multiple service access points (implements multiple subsystems), a multiple-context, single-queue model is recommended (as shown in the following illustration). In this case, the application has a single event loop with events from all service access points delivered through the same queue. The application determines which service access point a particular event is associated with from a service user ID (suID) value returned with each event.



Multiple-context, multiple-queue model

For multiple-threaded applications using multiple SCCP service access points (one per thread), a multiple-context, multiple-queue model is recommended (as shown in the following illustration). In this case, each thread has its own event loop and receives only the events associated with a service access point on its Natural Access queue.

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



Signaling parameters

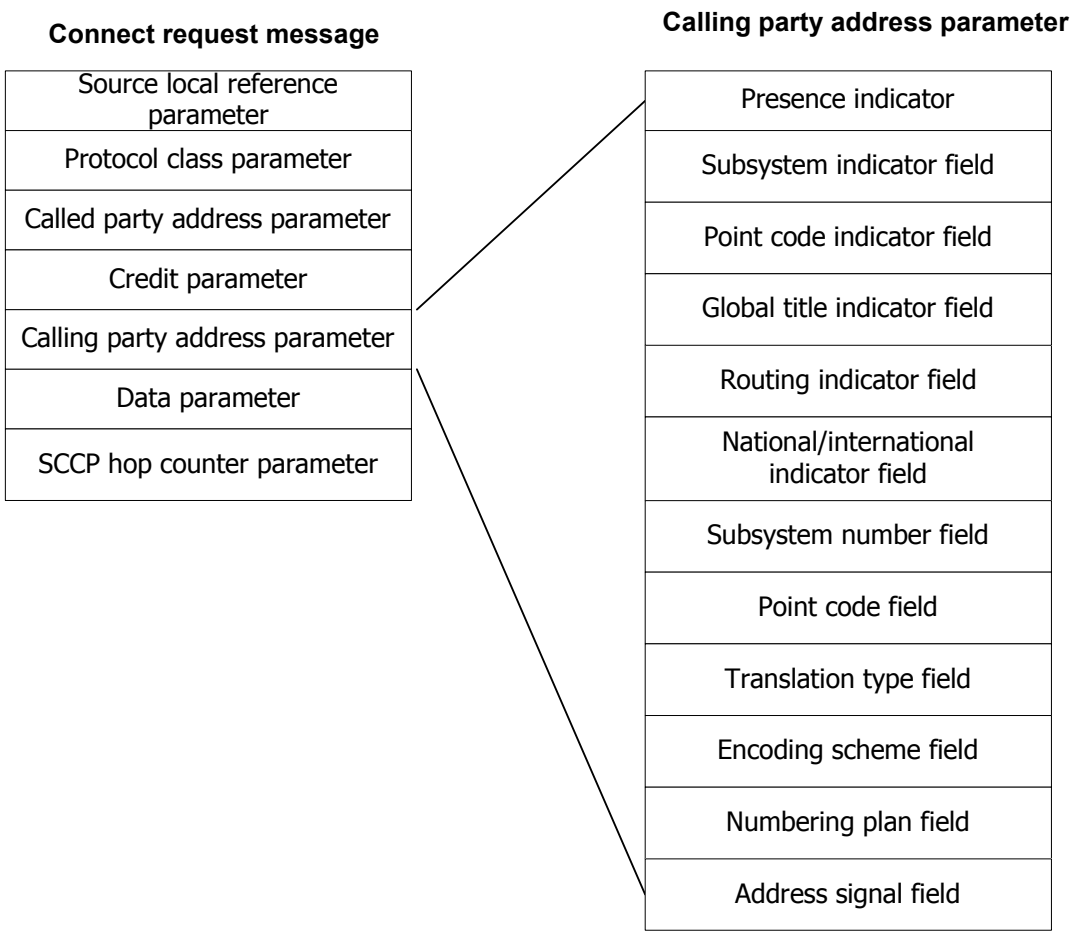
Signaling parameters are passed between the application and the SCCP task in the form of messages, which correlate with actual SCCP messages exchanged across the link. Messages are fixed format structures consisting of one or more parameters (referred to as information elements or IEs).

Parameters are fixed format structures consisting of one or more fields. Parameters that are optional in a message contain a flag indicating their presence or absence from the corresponding SCCP message.

Applications generate SCCP messages by allocating a message structure, populating the values and setting the present indicator for each optional parameter it wants to include in the message, and passing the message to the appropriate SCCP function.

Likewise, once received messages have been decoded into the fixed message format by NMS SCCP, the application can scan each parameter to determine if it was present in the SCCP protocol message and, if so, extract its field values.

The following structure simplifies applications by enabling them to operate on fixed format structures rather than the variable length and variable formats employed by the SCCP protocol. The following illustration shows an example of a message/parameter/field structure:



Signaling point and subsystem status procedures

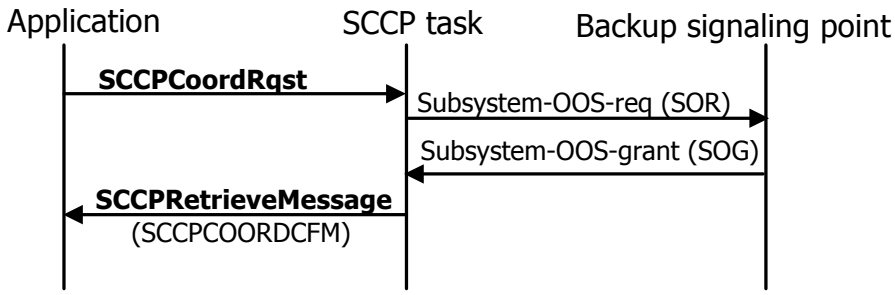
The SCCP service contains functions for maintaining signaling point and subsystem status between the system containing the TX device(s) and backup signaling points or concerned signaling points.

This topic describes the following:

- Coordinated state change
- Subsystem state changes
- Remote signaling point failures

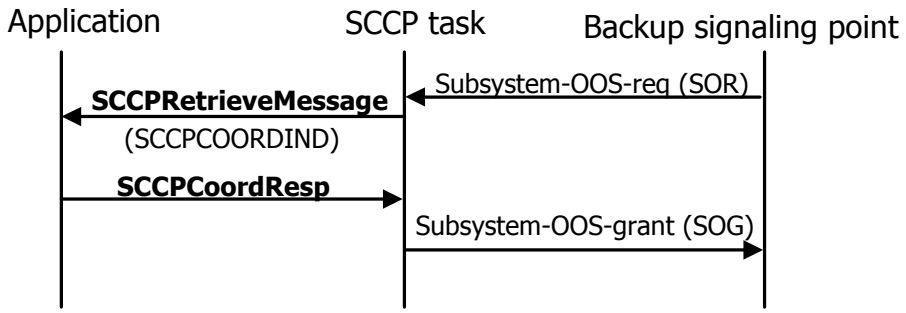
Coordinated state change

An application can request that its subsystem be taken out of service (and have all traffic routed to its backup point code) by invoking **SCCPCoordRqst**. This request generates an SCCP subsystem-out-of-service-request (SOR) to the backup signaling point (as specified in the SAP configuration). The application receives a SCCPCOORDCFM event when the backup signaling point returns a subsystem-out-of-service-grant (SOG) as shown in the following illustration:



If the backup signaling point fails to return a SOG message and the grant request times out, the SCCP layer returns SCCPCOORDCFM confirmation to the application with an indication that the request failed, and the application should not go out of service.

Alternatively, the backup point code can request to go out of service by sending the SOR message, resulting in the application receiving an SCCPCOORDIND event as shown in the following illustration. The application invokes **SCCPCoordResp** to accept the request and return the SOG message.



Subsystem state changes

The application notifies all concerned point codes of a change in its state (in-service or out-of-service) by invoking **SCCPStateRqst**. This request generates a subsystem available (SSA) or subsystem prohibited (SSP) message to all concerned signaling points as specified by the configuration of the application SAP.

Likewise, when the SCCP task receives messages from concerned signaling points indicating that their status has changed, the application receives an unsolicited SCCPSTATEIND indication (subsystem status) or SCCPPCSTIND indication (point code status).

Remote signaling point failures

An application can monitor the status of remote signaling points by specifying a list of concerned point codes in the user SAP configuration corresponding to that application.

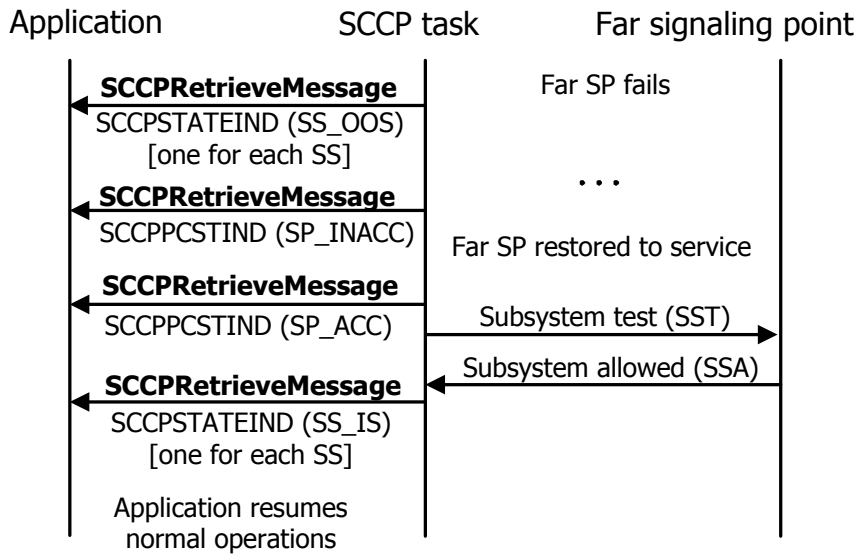
If a concerned point code (CPC) becomes inaccessible, the application receives a SCCPPCSTIND event with the status field set to SP_INACC (signaling point inaccessible). In addition, the application receives a SCCPSTATEIND event with the status field set to SS_OOS (subsystem out-of-service) for each known subsystem at that signaling point.

Similarly, if the MTP layer receives an indication from the remote SP that the SCCP user part is unavailable, the application receives the SCCPSTATEIND (SS_OOS) event for each known subsystem at that signaling point. The application does not receive the SCCPPCSTIND event since only the SCCP user part has failed and not the entire signaling point.

The status field associated with an SCCPPCSTIND event indicates whether active connections have been dropped (status SP_INACC) or retained (SP_INACC_NODROP) when a remote signaling point becomes inaccessible. Dropping or retaining connections during an outage is a configuration option. If connections are dropped, the application does not receive an individual release indication for each active connection. Connections must be re-established when the affected signaling point/subsystem returns to service. If connections are retained across an outage and the remote SP remains inaccessible for a duration longer than the configured receive inactivity timer, those connections are released and must be re-established when the link is restored. In this case, the application is notified with an SCCPRELIND event for each connection when it is released.

When communication with the affected signaling point is restored, the application receives an SCCPPCSTIND event with the status field set to SP_ACC (SP accessible). The SCCP layer initiates subsystem status testing of all known subsystems at the affected SP. When the affected SP returns a subsystem available message, the application receives an SCCPSTATEIND event with the status field set to SS_IS (subsystem in-service). The application can re-establish connections and/or resume connectionless data transfer with the affected SP/subsystem.

The following illustration shows an example of remote signaling point failure and recovery procedures:



A remote signaling point can become congested with too many outbound messages. If this occurs, the application receives an SCCPPCSTIND event with a status field of SP_CONG1, SP_CONG2, or SP_CONG3. The status fields indicate a worsening level of congestion. An application first receives SP_CONG1, followed by SP_CONG2 if congestion increases. Upon receiving these indications, the application reduces traffic to this signaling point until the congestion has eased.

At each congestion level, traffic flow to the signaling point is affected as follows:

Status field	Description
SP_CONG1	No action is taken on outbound messages. This indication is for information only.
SP_CONG2	Further outbound connectionless messages to this signaling point are returned. New connections to this signaling point are refused. Existing connection-oriented traffic is unaffected.
SP_CONG3	Further outbound connectionless messages to this signaling point are discarded. Connection-oriented traffic is treated the same as in SP_CONG2.
SP_CONG_OFF	Congestion has eased. Normal traffic handling is in effect.

The following alarm message is generated as each congestion level is reached:

```
SCCP Route %s Congested, Level = 1
SCCP Route %s Congestion Ended
```

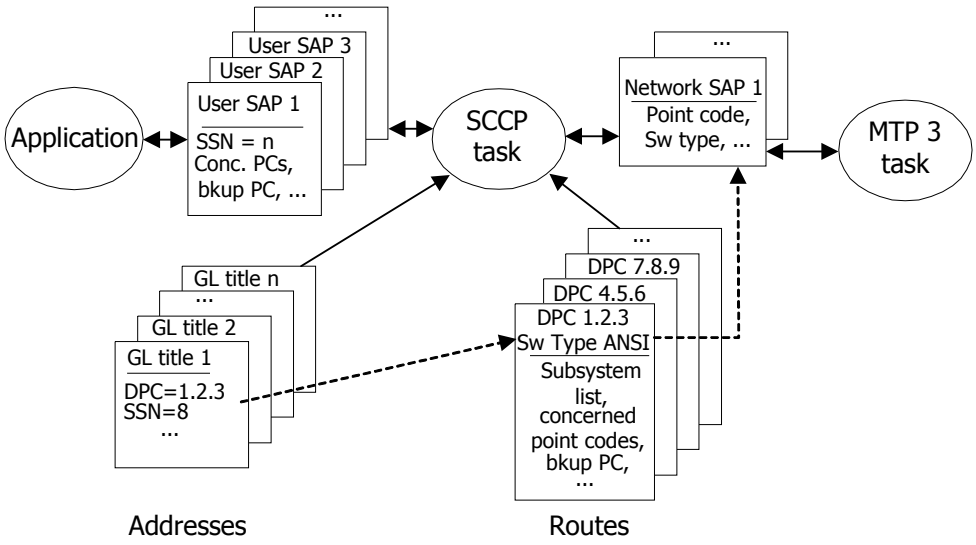
SCCP configuration

NMS SS7 provides a standard program to read the SS7 (including SCCP) configuration from a set of text files and download the configuration to the SS7 tasks running on the TX board. Refer to the *NMS SS7 Configuration Manual* for more information.

The SCCP layer supports the following configuration entities:

Configuration	Description
General	The general configuration defines the operational parameters of the SCCP layer: resource allocations, timer values, and threshold values. The general configuration is loaded only once at system boot time and must be loaded before any other configuration entities.
User SAPs	One user service access point (SAP) is defined for each application using the SCCP layer services. A user SAP is associated with a single subsystem number and switch type (ANSI or ITU-T). The user SAP defines whether the application is replicated on another node for reliability purposes, and lists any concerned point codes (for example, nodes that must be notified of any change in the status of the application). User SAPs are initially loaded at system boot time. Additional user SAPs can be added later, up to the maximum specified in the general configuration.
Network SAPs	One network service access point (NSAP or network SAP) is defined for each MTP 3 layer interface that the SCCP layer uses. Typically the SCCP layer has only a single network SAP, although if the same system supports multiple switch types (ANSI and ITU-T), the SCCP layer would have a separate network SAP for each switch type.
Routes	One route is defined for each destination subsystem that the SCCP layer can access. The route defines the destination point code used to reach that subsystem as well as any backup point code that replicates the subsystem. Routes are initially loaded at system boot time. Additional routes can be added later, up to the maximum specified in the general configuration.
Addresses	Address entries define how the SCCP layer is to translate and/or route between global titles, point codes, and subsystem numbers. Address translations are initially loaded at system boot time. Additional address translations can be added later, up to the maximum specified in the general configuration.

The following illustration shows the relationship between the various configurable entities:



SCCP addressing and routing

All SCCP connectionless data requests and connection establishment requests contain a mandatory called and calling party address. The calling address is optional for the connection request message. These addresses are passed to and from user applications by the SccpAddr data structure, which is a C-language structure representation of the actual address passed in the SCCP protocol message.

SCCP addresses can take several forms, containing various combinations of point code, subsystem number, and global title. The combination of the address and routing indicator constructed by applications (or received from the SS7 network) together with the SCCP configuration allow these messages to be routed to the correct destination or local application.

For outgoing messages from applications, the called party address in the unitdata request or connection request message is used to route the message. The routing method is chosen based on the value specified in the routing indicator field of the called party address. The routing methods are:

- Point code and subsystem number (ROUTE_PC_SSN)
- Global title (ROUTE_GLT)

Routing by point code and subsystem number

When ROUTE_PC_SSN is chosen, the message is routed to the destination point code/subsystem number (DPC/SSN) specified in the called party address. The subsystem number must be present.

If the DPC is present, a route must be configured for the point code and the subsystem must be configured for that route unless the default routing configuration option is selected.

If the DPC is absent, the message is routed to the point code associated with the first (and typically only) route in the SCCP configuration file, but that point code is not included in the outgoing message. This option is typically used only on point-to-point SS7 links, such as the link between a mobile switching center (MSC) and base station controller (BSC) in a wireless network, where the destination point code is not needed for routing.

If a global title is present in the called party address, it is copied to the outgoing message but the routing indicator remains ROUTE_PC_SSN.

Routing by global title

When ROUTE_GLT is chosen, an address translation must be configured that, when combined with the address mask configured for the application SAP, matches the global title specified in the called party address. The message is routed to the point code and subsystem number from the configured address translation entry. If no subsystem number is configured for that address translation but one is supplied by the application, it is copied to the outgoing message.

If the global title must be further translated at another node, the routing indicator configured in the address translation entry specifies ROUTE_GLT and the point code in the address translation entry specifies the next translator node. If this is the final translation, the address translation entry specifies ROUTE_PC_SSN and the DPC in the address translation entry specifies the final destination point code.

When a global title is specified by the user, the global title is translated if possible by the SCCP task, and the global title is included in the outgoing SCCP address along with up to four parameters.

The glTransType, encoding, numPlan, and natAddrInd fields are only used with a global title. Based on the value of the glTitleInd field, some or all of these values are included with a global title in an outgoing SCCP address.

If swType is set to SW_ANSI, two combinations can be included with a global title. The glTitleInd field determines which is selected:

```
global title + translation type (glTitleInd = GLT_TT)
```

```
global title + translation type + numbering plan + encoding (GLT_TT_NP_E)
```

If swType is set to SW_ITU, four combinations can be included with a global title. The glTitleInd field determines which is selected:

```
global title + encoding + nature of address
(glTitleInd = GLT_ITU_FMT1)
```

```
global title + translation type (GLT_ITU_FMT2)
```

```
global title + translation type + numbering plan + encoding (GLT_ITU_FMT3)
```

```
global title + translation type + numbering plan + encoding + nature of address
(GLT_ITU_FMT4)
```

If the user wants the global title to be passed to another node for translation, the ROUTING_IND field can be defined in one of the ADDR sections in the SCCP configuration file. If ROUTING_IND is set to GLT, the global title is passed along with its routing field set to route by global title. The point code field is included, but the subsystem field is not included in the outgoing message.

Refer to *Global title translation* on page 26 for more information.

Global title translation

A global title translation (GTT) translates an array of phone numbers into an associated point code and subsystem for routing to another machine.

This topic presents the following information:

- Setting the fields in a TCAP request
- Setting the SCCP configuration
- Setting variations in global title translation

Setting the fields in a TCAP request

The following example is from a TCAP sample program that uses a global title. This example shows the setting of only the called address SCCP address structure:

```
tInfo.cdAddr.presInd = PRESENT;
tInfo.cdAddr.swType  = SW_ANSI;
tInfo.cdAddr.subsystemInd = SUBSYS_NONE; /* A subsystem does not need to be defined    */
tInfo.cdAddr.pointCodeInd = PTCODE_NONE; /* A point code does not need to be defined   */
tInfo.cdAddr.glTitleInd =  GLT_TT;       /* A global title format MUST be selected     */
tInfo.cdAddr.routingInd =  ROUTE_GLT;    /* The routing flag MUST be set to ROUTE_GLT  */
tInfo.cdAddr.natIntInd =   ADDRIND_INT;
//tInfo.cdAddr.subsystem =   destssn;    /* The subsystem is not required              */
//tInfo.cdAddr.pointCode =   pointCode;  /* The point code is not required             */
tInfo.cdAddr.glTransType = 1;            /* The global title fields MUST be filled     */
tInfo.cdAddr.encoding = ENC_BCD_EVEN;
tInfo.cdAddr.numPlan =  NP_ISDN;
tInfo.cdAddr.natAddrInd = NATIND_NATL;
tInfo.cdAddr.glTitleLen = 5;       /* The global title length is the BCD-encoded length*/
/* BCD encode the phone number */
for ( i = 0; i < 5; i++)
{
ch = gtitle[((i*2)+1)] - 0x30;
tInfo.cdAddr.glTitle[i] = (ch << 4) & 0xF0;
ch = gtitle[(i*2)] - 0x30;
tInfo.cdAddr.glTitle[i] += ch;
}
```

The global title must be BCD-encoded. The global title 8471234567 is BCD encoded as:

```
tInfo.cdAddr.glTitle[0] = 0x48;
tInfo.cdAddr.glTitle[1] = 0x17;
tInfo.cdAddr.glTitle[2] = 0x32;
tInfo.cdAddr.glTitle[3] = 0x54;
tInfo.cdAddr.glTitle[4] = 0x76;
```

The original global title was 10 digits long. When the global title is BCD-encoded, it is 5 bytes in length. This length is used as the global title length:

```
tInfo.cdAddr.glTitleLen = 5;
```

Setting the SCCP configuration

Once the application sends the TCAP message with the route by global title flag set, the SCCP task receives the message and attempts the global title translation. The SCCP task uses three steps to translate a global title:

1. The SCCP task masks the outgoing global title with the ADDR_MASK field in the USER SAP section of the SCCP configuration file.

   In the previous example, the global title was 8471234567. An ADDR_MASK of FFF masks the first three digits of the global title. The result of the masking in the example is the first three digits - 847.

   On incoming messages, the ADDR_MASK in the NSAP section of the SCCP configuration file masks global titles.

2. The SCCP task matches the masked result with the ADDRESS sections in the SCCP configuration file. If a matching section is not found, the SCCP message is returned to the application.

   In the example, an ADDRESS 847 section is configured in the SCCP configuration file and matches the masked global title:

```
ADDRESS           847             # global title matching characters
REPLACE_GLT       FALSE           # do NOT replace the global title
SWITCH_TYPE       ANSI            # one of ITU, ANSI
NI_IND            NATIONAL        # one of NATIONAL, INTERNATIONAL
DPC               1.1.2           # translated destination point code
SSN               254
ROUTING_IND       PC_SSN          # set outgoing routing flag (PC_SSN or GLT)
END                               #
```

3. The SCCP task modifies the SCCP called address of the outgoing message. In the example, the outgoing message has a called address of:

```
cdAddr.pointCode =  1.1.2
cdAddr.subsystem = 254
cdAddr.routingInd = "Route by PC_SSN"
```

   The global title specified by the application is also carried in the outgoing message.

   Note: The point code used in the translated message must be listed as a ROUTE in the SCCP configuration file.

Setting variations in global title translation

The most common variations to global title translation are presented.

Forward a global title translation to another node

A global title translation is forwarded to another node. The following example is a sample global title translation block (ADDRESS section) in the SCCP configuration file. 1.1.2 is the point code of the node that performs the global title translation:

```
ADDRESS           847             # global title matching characters
REPLACE_GLT       FALSE           # do NOT replace the global title
SWITCH_TYPE       ANSI            # one of ITU, ANSI
NI_IND            NATIONAL        # one of NATIONAL, INTERNATIONAL
DPC               1.1.2           # forward translation to this point code
ROUTING_IND       GLT             # set outgoing routing flag(PC_SSN or GLT)
END                               #
```

This translation forwards a message to another node (1.1.2) to do the global title translation.

The ROUTING_IND field is set to GLT (route by global title), which sets the routing flag in the outgoing message. The DPC field contains the point code of the node that receives the message and translates the global title. The SSN field does not need to be defined. The called address of the outgoing message is set to:

```
cdAddr.pointCode =  1.1.2
cdAddr.subsystem = "not encoded"
cdAddr.routingInd = "Route by Global Title"
```

The global title specified by the application is also carried in the outgoing message.

Note: The point code of the forwarded node must be listed as a route in the SCCP configuration file.

Configure a single ADDRESS section that matches any global title

Set the ADDR_MASK in the USER SAP section of the SCCP configuration file to 0 (zero). Create an ADDRESS 0 translation block. This block now matches all global titles.

Do not encode a point code in the translated outgoing message

Remove the DPC field from the ADDRESS translation block. The called address DPC is not encoded in the outgoing message. Use the default routing option to provide an MTP header DPC.

Copy the application specified subsystem in the translated outgoing message

Remove the SSN field from the ADDRESS translation block. The subsystem field specified by the application in the SCCP called address (if it exists) is inserted in the SCCP called address of the outgoing message.

Connection IDs

For connection-oriented services, the application and the SCCP task use connection IDs to reference the SAP (subsystem) and to reference the connection to which a particular message applies. The connection IDs reference both requests and responses from the application to the SCCP layer as well as indications and confirmations from the SCCP task to the application. The structure of a connection ID is as follows:

```
typedef struct connectID
{
  S16   suId;      /* service user ID               */
  S16   spId;      /* service provider ID           */
  U32   suConnId;  /* user's connection ref number  */
  U32   spConnId;  /* SCCP's connection ref number  */
} ConnectID;
```

Field	Description
suId	The application reference number for the SAP that is used. This reference number is the same as the suId passed by the application in the bind request. The application that implements multiple SAPs can use this field on incoming messages to identify the associated SAP.
spId	The SCCP SAP number to which the message belongs. The application sets this field on all messages from the application to the SCCP layer.
suConnId	An arbitrary number that the application uses to identify the connection. The application sets this value on the first message associated with a connection - a connect request for outgoing connections or a connect response for incoming connections. The SCCP layer returns this value on all subsequent messages belonging to this connection.
spConnId	SCCP number for identifying the connection. The application saves this value from the first message received associated with a connection - a connect confirm for outgoing connections or a connect indication for incoming connections - and passes it on all subsequent messages belonging to this connection.

For connectionless functions, only suId and spId are used to enable the application and the SCCP layer, respectively, to identify the SAP associated with a connectionless request.

Application inactivity control

The SCCP layer performs application inactivity control timing. Application inactivity control timing detects connections maintained by the SCCP layer of which the application is no longer aware. These undetected connections can occur in a failover scenario in a redundant configuration when the connection data maintained by both the application and the SCCP layer can get out of synchronization. Failure to detect and release these stranded connections reduces the number of subsequent connections that can be created.

Note: Application inactivity control timing is over and above the SCCP protocol-level connection inactivity timing (as defined in the ITU-T and ANSI SCCP specifications) that is intended to detect mismatched connection data between the two endpoints. This function is still performed by the SCCP layer and is transparent to the application.

When application inactivity control is enabled for a particular application (SCCP user SAP), the SCCP layer starts a timer (AIC_TIMER) when a connection is established in either direction by that application. Each time the application issues a data request for that connection, the AIC_TIMER is restarted. Whenever the timer expires, the application is notified with a SCCPINACTIND event for that connection and a second timer (AIC_RESP_TIMER) is started. If the application recognizes the connection ID in the SCCPINACTIND event as belonging to a valid (active) connection, it responds by calling **SCCPInactResp** for that connection. Otherwise, it should call **SCCPReleaseRqst** to release the connection. If the application fails to respond to the SCCPINACTIND event before the AIC_RESP_TIMER expires, the SCCP layer releases the connection to the far end and returns a SCCPRELIND to the application for that connection ID.

The duration of the AIC_TIMER (8 minutes by default) and AIC_RESP_TIMER (10 seconds by default) are configured with the *sccpcfg* utility or by **SccpSetGenCfg**. In addition, application inactivity timing must be enabled for each SCCP user SAP by setting the INACT_CONTROL parameter to TRUE (or 1 or YES) in the SCCP configuration file or by calling **SccpSetUSapCfg** with the aicEnabled attribute set to 1.

Application inactivity control should not be enabled for an application until it can explicitly handle the SCCPINACTIND event and respond by calling **SCCPInactResp**.

Connection auditing

An application can audit the status of all in-progress connections maintained by the SCCP layer by calling **SCCPConnAuditRqst**. **SCCPConnAuditRqst** can be invoked to either the primary or the backup SCCP layer in a redundant configuration. Use connection auditing to:

- Resynchronize a backup application that has failed and restarted
- Ensure that the application and the SCCP layer have consistent connection data
- Troubleshoot an application

Each call to **SCCPConnAuditRqst** generates a SCCPCONNAUDCFM event to the application containing connection data (connection ID, calling and called addresses, connection state) for one connection. On the first call, the application specifies a connection ID (the spInstId attribute of the connection ID) of zero. On all subsequent calls, the application specifies the connection ID returned in the previous SCCPCONNAUDCFM event that returned a connection. When all active connections are exhausted, the SCCP layer responds to a call to **SCCPConnAuditRqst** with a SCCPCONNAUDCFM event with an event type of SCCP_CONNAUDEOF, indicating there are no more active connections. The following example demonstrates the initiation of the connection audit:

```
SccpConnId      connId;
DWORD           ret;

...

/* populate connection ID to kick off audit */
connId.suId = mySuid;
connId.spId = mySpid;
connId.suConnId = 0;            /* not used */
connId.spConnId = 0;            /* start at first connection */

/* issue the command to the SCCP service API */
ret = SCCPConnAuditRqst( myCtaHd, &connId );

...
```

Connection audit requests return data on all active connections for all SCCP user SAPs, not just those associated with the calling application. If more than one user application uses connection-oriented services, the application associated with a connection returned in a SCCPCONNAUDCFM event can be determined from the service provider ID (SCCP user SAP number) attribute of the connection ID or from the subsystem number field in the calling or called address parameter.

The following example demonstrates the handling of a connection audit confirmation:

```
    SccpAllMsgs     sccpMsg;
    SccpRcvInfoBlk  infoBlk;

      ...

    /* get SCCP message from API */
    if( (retval = SCCPRetrieveMessage( myCtaHd, &sccpMsg, &infoBlk, 0 )) != SCCP_SUCCESS
)
    {
      ...
    }

    switch (infoBlk.indType)
    {

      ...

       case SCCPCONNAUDCFM:
       {

           /* check if connection info returned or end of connections */
           if( infoBlk.evntType == SCCP_CONNAUDOK )
           {
               /* display returned connection info */
               printf( ... );

               /* request next connection audit info, using connection ID
                  just returned */
               ret = SCCPConnAuditRqst( myCtaHd, &infoBlk.connId );
           }
           else if( infoBlk.evntType == SCCP_CONNAUDEOF )
           {
               printf( "Connection audit complete\n" );
           }

          ...

           break;
       }

     ...
```

4 Using the SCCP service

SCCP service overview

The SCCP interface for connectionless services consists of two phases:

- Initialization and binding
- Data transfer

The SCCP interface for connection-oriented services consists of four phases:

- Initialization and binding
- Establishing connections
- Data transfer
- Clearing connections

Setting up the Natural Access environment

Before calling any SCCP service functions, the application must:

- Initialize Natural Access
- Create queues and contexts
- Bind to the SCCP service

Refer to the *Natural Access Developer's Reference Manual* for more information about Natural Access.

Initializing the Natural Access environment

The Natural Access environment is initialized by calling **ctaInitialize**. Initialize Natural Access only once per application, regardless of the number of queues and contexts created.

```
CTA_INIT_PARMS      sccpInitparms       = {0};
CTA_SERVICE_NAME    sccpServiceNames[]  = {{"SCCP", "SCCPMGR"}};
...
sccpInitparms.size              = sizeof(CTA_INIT_PARMS);
sccpInitparms.traceflags        = CTA_TRACE_ENABLE;
sccpInitparms.parmflags         = CTA_PARM_MGMT_SHARED;
sccpInitparms.ctacompatlevel    = CTA_COMPATLEVEL;

Ret = ctaInitialize(sccpServiceNames, 1, &sccpInitparms);
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. The queue must always be created before any associated context is created.

```
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, "SccpSAP-%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 );
     ...
}
```

Binding to the SCCP service

Once the queues and contexts are created, the application must bind to each desired SCCP user service access point by calling **ctaOpenServices** once for each binding. The binding operation specifies the following parameters:

Parameter	Description
board	TX board number.
srcEnt	Calling application entity ID.
srcInst	Calling application instance ID.
suId	Calling application service user ID.
spId	SCCP service access point ID on which to bind.
ssn	SCCP subsystem number associated with the service access point.
poolsize	Number of messages allowed to be queued to the TX board. Default value is 128. Maximum value of 1024.

In 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 sccpOpenSvcLst[] = {{{"SCCP", "SCCPMGR"}, {0}, {0}, {0}}};

sccpOpenSvcLst[0].svcargs.args[0] = board;   /* board number           */
sccpOpenSvcLst[0].svcargs.args[1] = INST_ID; /* srcInst                */
sccpOpenSvcLst[0].svcargs.args[2] = ENT_ID;  /* srcEnt                 */
sccpOpenSvcLst[0].svcargs.args[3] = 0;       /* reserved for future use */
sccpOpenSvcLst[0].svcargs.args[4] = SAP_ID;  /* spId                   */
sccpOpenSvcLst[0].svcargs.args[5] = SAP_ID;  /* suId                   */
sccpOpenSvcLst[0].svcargs.args[6] = DST_SSN; /* subsystem number       */
sccpOpenSvcLst[0].svcargs.args[7] = 0;       /* reserved for future use */
sccpOpenSvcLst[0].svcargs.args[7] = poolsize /* number of queued msgs.  */
```

ctaOpenServices is an asynchronous function. The return from the function indicates that the bind operation 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 the 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 SCCP events */
...


Ret = ctaOpenServices( ctaHd, sccpOpenSvcLst, 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 SCCP service [%s]\n", sErr );
    ctaDestroyQueue( pSap->ctaQueue );    /* destroys context too */
    return( ... );
}
```

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

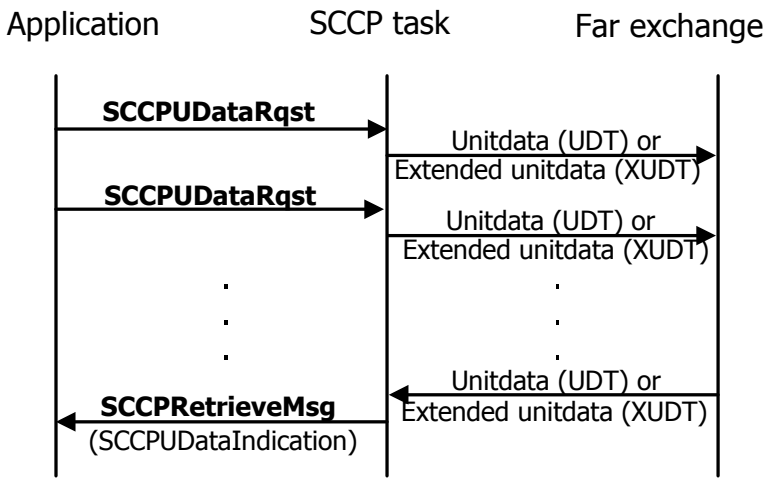
It is also recommended that the application call **SCCPStateRqst** and mark the subsystem in service as soon as it is ready to handle data traffic, in case the subsystem was previously left out of service by an application unbinding from the same SCCP service access point.

Transferring connectionless data

An application using the connectionless service sends data to the far end by invoking **SCCPUDataRqst**. Depending on the size of the data and the class of service employed, the SCCP task generates a unitdata (UDT) or extended unitdata (XUDT) message to the far end as shown in the following illustration. There is no response from the far end to these messages.

If the far end SCCP layer cannot process the received data message and returns a unitdata service message (UDTS) or extended unitdata service message (XUDTS) to the SCCP task, the application is notified with a SCCPStatusIndication.

Incoming unitdata (UDT and/or XUDT) messages are presented to the application with a SCCPUDATIND event.

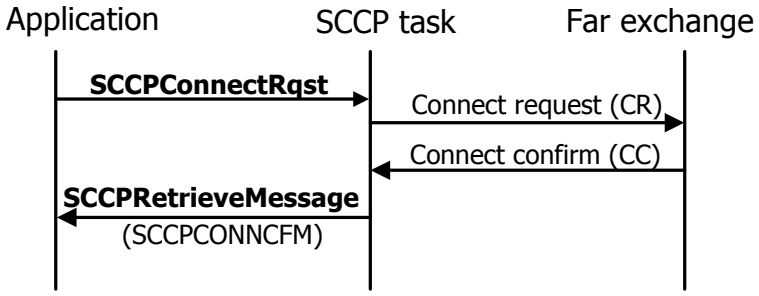


Undeliverable connectionless messages

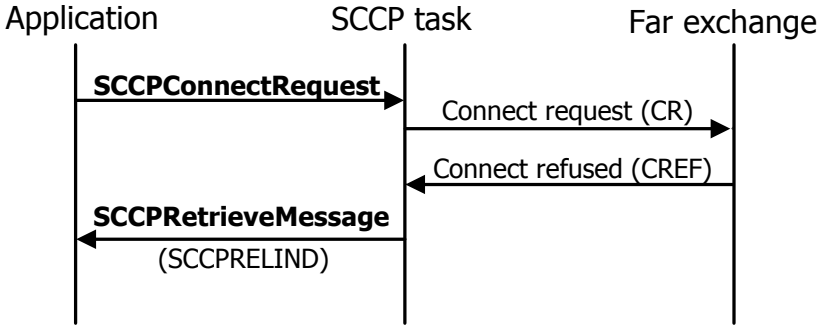
If the application has selected the return on error option on a unitdata request and either the local SCCP layer is unable to deliver the message or the remote SCCP returns the message as undeliverable (through the UDTS or XUDTS message), the application is notified with a SCCPSTAIND event. This event contains the reason for the delivery failure and the original content of the undeliverable message.

Establishing connections

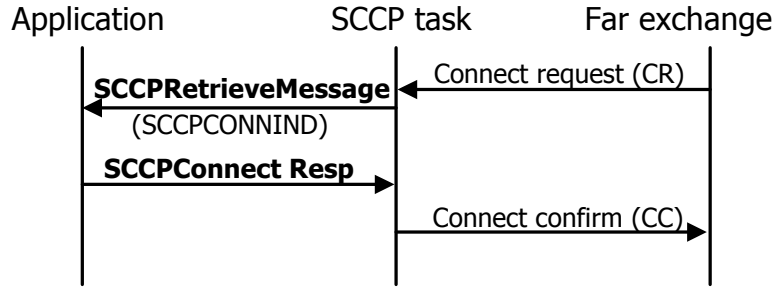
The application initiates an SCCP connection by invoking **SCCPConnectRqst**, resulting in the generation of an SCCP connect request message (CR) to the far exchange, as shown in the following illustration:



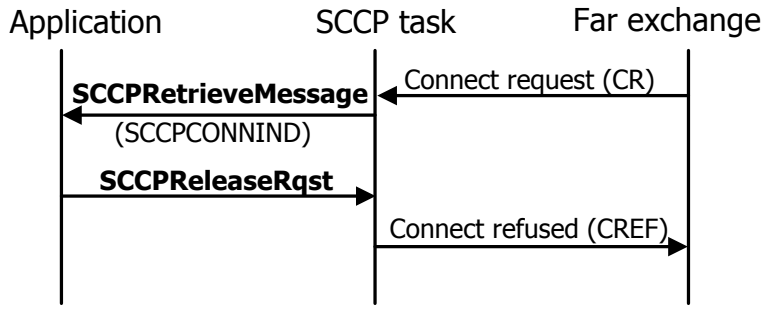
The connection establishment phase ends when the application receives the SCCPCONNCFM message (far exchange sent connection confirm). If the far end refuses the connection request, the application receives a SCCPRELIND indication instead of a connect confirmation, as shown in the following illustration:



Alternatively, the far exchange can initiate the connection by sending the CR message, resulting in the application receiving an SCCPCONNIND event, as shown in the following illustration:



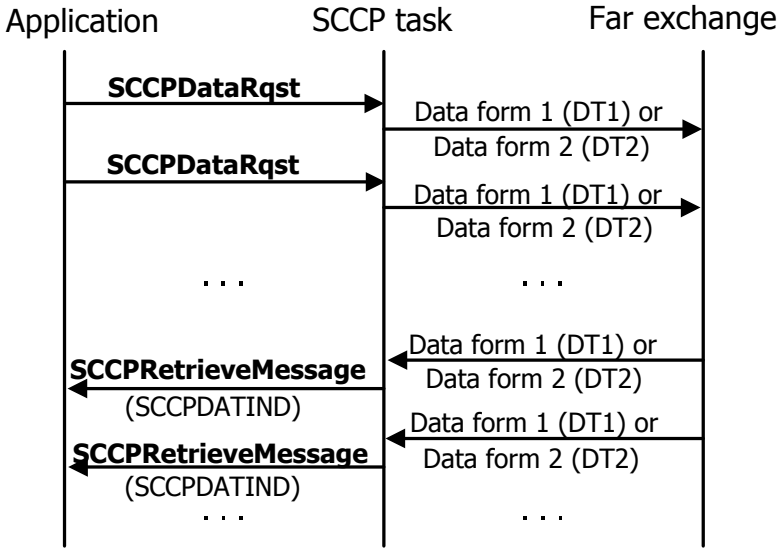
For an incoming connection, the application invokes **SCCPConnectResp** to signal to the far end that the connection is accepted. If the application refuses the connection, it calls **SCCPReleaseRqst**, as shown in the following illustration:



After delivering a SCCPCONNIND event to the application, the SCCP layer on the TX board starts a timer. Timing for the response prevents SCCP layer memory and connection resources from being stranded if the application fails to respond. When an application fails to respond to an incoming connection indication before the application connection response timer expires, the SCCP layer automatically refuses the connection to the originator and notifies the application with a SCCPRELIND event for that connection. The time allowed for the application connection response is configured by the ACR_TIMER configuration parameter. This feature can be disabled by configuring the ACR_TIMER value to 0 (zero).

Transferring connection-oriented data

During the time that the call is connected, the application can exchange data with its peer in the far end point by invoking **SCCPDataRqst** or **SCCPEDataRqst** (expedited data). Incoming messages are signaled to the application with the SCCPDATIND or SCCPEDATIND indication events, as shown in the following illustration:



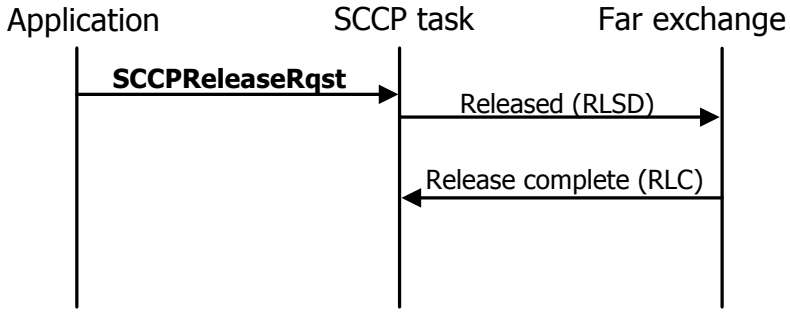
Resetting connections

The application can request that an active (class 3) connection be reset by invoking **SCCPResetRqst**. When the far end acknowledges the reset with a reset confirmation message, the application receives a SCCPRESETCFM event.

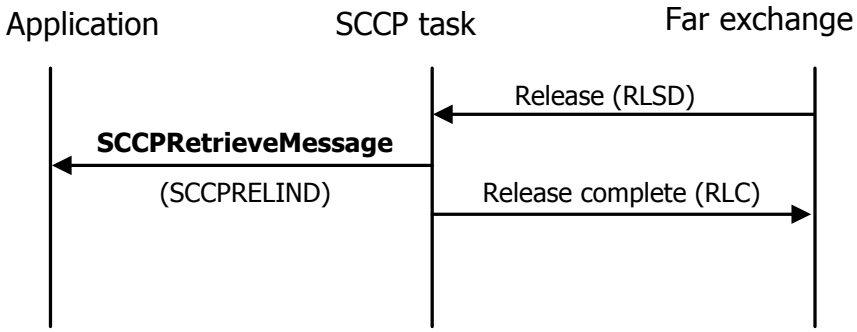
Likewise, if the far end initiates the connection reset, the application receives a SCCPRESETIND event and is expected to respond by invoking **SCCPResetResp**.

Clearing connections

The application requests clearing a connection by invoking **SCCPReleaseRqst**, which results in an SCCP released (RLSD) message to the far end. No confirmation is returned to the application, as shown in the following illustration:



When a connection is released first by the far party, the application receives a SCCPRELIND indication, as shown in the following illustration. There is no response from the application to this indication.



Handling redundancy events

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

Event type	Description
SPRS_STANDALONE	Application is in a non-redundant configuration. Normal operation can begin.
SPRS_PRIMARY	SCCP 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.
SPRS_BACKUP	SCCP 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 SCCPRUNSTATEIND event is the first message posted to the application's 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.

Refer to the *SS7 Health Management Developer's Reference Manual* for information on writing redundant SCCP applications.

Handling congestion events

Outbound and inbound congestion control features are available for the SCCP service.

Outbound congestion

After binding to an SCCP user SAP, an application can receive a SCCPEVN_CONGEST event. The Event.value parameter contains the congestion level. When this event is received, one of the following events has occurred:

- The SCCP service has queued a number of events that have not been read by the SCCP task on the board.
- Memory available on the board has become very low.

The application can determine which event occurred by calling **SCCPGetStats**.

When this congestion event is received, the application must reduce traffic sent to the board.

If the congestion is due to the SCCP service, the only effect on traffic flow is that at some point after congestion level 3 occurs, requests sent to the board can fail with CTAERR_DRIVER_SEND_FAILED or SCCPERR_RESOURCES errors. The application must reduce traffic flow to avoid this possibility.

If the congestion is due to low available memory on the board, the SCCP service implements the following congestion procedure:

Congestion level	Action taken
1	No action is taken on outbound or inbound messages. This indication is for information only.
2	Further outbound connectionless messages are returned. All new connections (inbound and outbound) are refused. Existing connection-oriented traffic is unaffected.
3	Further inbound and outbound connectionless messages are discarded. Connection-oriented traffic is treated the same as in level 2.
0	Congestion has eased. Normal traffic handling is in effect.

The levels used to determine memory congestion are configured in the SCCP general configuration and can be altered by the user as required. Refer to **SccpInitGenCfg** for more information.

The following alarm message is generated as each congestion level is reached:

```
SCCP Memory Congestion Level = 1
```

Inbound congestion

Although not reported to the application, congestion can also occur due to excessive inbound traffic to a specific user SAP. If an application cannot read traffic from the board quickly enough, these messages build on a queue. As each congestion level is reached, procedures are implemented to limit further inbound traffic to that user SAP. Traffic to other user SAPs is unaffected.

Congestion level	Action taken
1	Half of new inbound or outbound connections are refused. Traffic for existing connections is unaffected.
2	All new inbound or outbound connections are refused. Traffic for existing connections is unaffected. Inbound connectionless messages are returned.
3	All new inbound or outbound connections are refused. Traffic for existing connections is unaffected. Inbound connectionless messages are discarded.

The levels used to determine inbound congestion are configured in the SCCP user SAP configuration and can be altered by the user as required. Refer to **SccpInitUSapCfg** for more information.

The following alarm message is generated as each congestion level is reached:

```
SCCP User Sap 0 Queue Congestion Level = 1
```

Tracing function calls and events

Natural Access provides a mechanism for tracing function calls and events issued or received by an application. To capture trace messages, the Natural Access Server (*ctdaemon*) must be running, and the SCCP service must be included in the [ctasys] section of the *cta.cfg* file, as shown:

```
#=============================================================================
# Natural Access System Configuration (ctasys)
#
# Valid options are:
#   Service = name, dll   - tells the daemon about available "services"
#                           - tells the Natural Access server what "services"
#                          to export
#
# Note: NCC should always precede ADI when both services are listed.
#=============================================================================
[ctasys]
Service = ncc, adimgr
Service = adi, adimgr
Service = ais, aismgr
Service = dtm, adimgr
Service = sccp, sccpmgr
Service = ppx, ppxmgr
Service = swi, swimgr
Service = vce, vcemgr
Service = oam, oammgr
```

In addition, the application must enable tracing when Natural Access is initialized:

```
sccpInitparms.size              = sizeof(CTA_INIT_PARMS);
sccpInitparms.traceflags        = CTA_TRACE_ENABLE;
sccpInitparms.parmflags         = CTA_PARM_MGMT_SHARED;
sccpInitparms.ctacompatlevel    = CTA_COMPATLEVEL;

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

For more information about tracing, refer to the *Natural Access Developer's Reference Manual.*

5 SCCP service function reference

SCCP service function summary

The SCCP service consists of the following service functions:

Function	Description
SCCPConnAuditRqst	Requests an audit of connection data for the next connection following the connection specified by the spInstId field of the connID parameter.
SCCPConnectResp	Accepts an incoming connection request.
SCCPConnectRqst	Requests establishment of an SCCP connection.
SCCPCoordResp	Accepts a request that an associated signaling point subsystem be taken out of service.
SCCPCoordRqst	Requests the subsystem be taken out of service and have all traffic routed to its backup point code.
SCCPDAckRqst	Reserved for future implementation.
SCCPDataRqst	Sends user data associated with an established connection to the far end point.
SCCPEDataRqst	Sends an expedited data message associated with an established connection to the far end point.
SCCPGetStats	Retrieves congestion level activity statistics from the SCCP service.
SCCPInactResp	Called in response to an SCCPINACTIND event received from the SCCP layer to indicate that the specified connection is still active.
SCCPReleaseRqst	Clears an SCCP connection by generating a released message to the far exchange.
SCCPResetResp	Acknowledges that a reset request indication was received from the far end point of an SCCP connection.
SCCPResetRqst	Sends a reset message to the far end point and starts the reset timer.
SCCPRetrieveMessage	Retrieves the next message from the SCCP layer.
SCCPStateRqst	Notifies all concerned point codes of a change in its subsystem state.
SCCPUDataRqst	Sends a datagram to the destination specified in the calledPty parameter of the SCCPUDataRqst message.

Using the SCCP service function reference

This section provides an alphabetical reference to the SCCP service functions. A prototype of each function is shown with the function description and details of all arguments and return values. A typical function includes:

Prototype	The prototype is followed by a listing of the function arguments. NMS data types include: • DWORD (8-bit unsigned) • S16 (16-bit signed) • U32 (32-bit unsigned) • Bool (8-bit unsigned) • U8 (8-bit unsigned) If a function argument is a data structure, the complete data structure is shown. **Note:** Not all parameters are applicable to both ANSI and ITU-T (CCITT) networks.
Return values	The return value for a function is either SCCP_SUCCESS or an error code. For asynchronous functions, a return value of SCCP_SUCCESS (zero) indicates the function was initiated; subsequent events indicate the status of the operation.

SCCPConnAuditRqst

Requests an audit of connection data for the next connection following the connection specified by the spInstId field of the connID parameter.

Prototype

DWORD **SCCPConnAuditRqst** (CTAHD ***ctahd***, SccpConnId ****connId***)

Argument	Description
ctahd	Natural Access handle returned by **ctaCreateContext**.
connId	Pointer to the connection ID.

Return values

Return value	Description
SCCP_SUCCESS	
SCCPERR_RESOURCES	Number of available buffers is exhausted.
CTAERR_DRIVER_SEND_FAILED	Error occurred accessing the CPI driver. Call **SCCPGetStats** to obtain the error code. The error code is in the failTxErr field of the SccpStats structure. The CPI error codes are listed in the *txcpi.h* include file.
CTAERR_INVALID_CTAHD	Natural Access handle is invalid.

Details

A successful call to this function results in a SCCPCONNAUDCFM event generated to the calling application, containing either connection data (connection ID, calling and called addresses, connection state) or an indication that there are no more connections to audit.

This function is typically used to sequentially retrieve data on all in-progress connections. On the first call, the connId->spInstId field is set to zero. This setting causes the SCCP layer to return audit data on the first in-progress connection in its internal connection tables. On subsequent calls, the application specifies the connection ID returned with the previous SCCPCONNAUDCFM event containing connection data. This sequence continues until the SCCP layer returns a SCCPCONNAUDCFM event with an event type indicating that there are no more connections to audit.

Refer to *Connection auditing* on page 31 for more information.

SCCPConnectResp

Accepts an incoming connection request.

Prototype

DWORD **SCCPConnectResp** (CTAHD ***ctahd***, SccpConnId ****connId***, SccpConnRqst ****connRqst***)

Argument	Description
ctahd	Natural Access handle returned by **ctaCreateContext**.
connId	Pointer to the connection ID.
connRqst	Pointer to the caller's connect confirm message structure containing all parameters to be included in the SCCP connect confirmation.

Return values

Return value	Description
SCCP_SUCCESS	
SCCPERR_OVERRUN	Message generated by this request exceeds the maximum allowed size.
SCCPERR_RESOURCES	Number of available buffers is exhausted.
CTAERR_BAD_ARGUMENT	One or more arguments are invalid.
CTAERR_DRIVER_SEND_FAILED	Error occurred accessing the CPI driver. Call **SCCPGetStats** to obtain the error code. The error code is in the failTxErr field of the SccpStats structure. The CPI error codes are listed in the *txcpi.h* include file.
CTAERR_INVALID_CTAHD	Natural Access handle is invalid.

Details

To refuse an incoming connection request, call **SCCPReleaseRqst**.

Refer to *Establishing connections* on page 37 for more information.

SCCPConnectRqst

Requests establishment of an SCCP connection.

Prototype

DWORD **SCCPConnectRqst** (CTAHD ***ctahd***, SccpConnId ****connId***, SccpConnRqst ****connRqst***)

Argument	Description
ctahd	Natural Access handle returned by **ctaCreateContext**.
connId	Pointer to the connection ID.
connRqst	Pointer to the caller's connect event structure containing all parameters relevant to establishing this connection.

Return values

Return value	Description
SCCP_SUCCESS	
SCCPERR_OVERRUN	Message generated by this request exceeds the maximum allowed size.
SCCPERR_RESOURCES	Number of available buffers is exhausted.
CTAERR_BAD_ARGUMENT	One or more arguments are invalid.
CTAERR_DRIVER_SEND_FAILED	Error occurred accessing the CPI driver. Call **SCCPGetStats** to obtain the error code. The error code is in the failTxErr field of the SccpStats structure. The CPI error codes are listed in the *txcpi.h* include file.
CTAERR_INVALID_CTAHD	Natural Access handle is invalid.

Details

If the SCCP layer cannot successfully initiate the outgoing connection request (for example, due to network congestion), it returns an asynchronous released indication message to the application with the cause value coded with the reason for the failure.

Refer to *Establishing connections* on page 37 for more information.

SCCPCoordResp

Accepts a request that an associated signaling point subsystem be taken out of service.

Prototype

DWORD **SCCPCoordResp** (CTAHD ***ctahd***, S16 ***spId***, SccpCoordRqst ****coordRqst***)

Argument	Description
ctahd	Natural Access handle returned by **ctaCreateContext**.
spId	SCCP service access point.
coordRqst	Pointer to the caller's coordrqst message structure containing the affected subsystem number.

Return values

Return value	Description
SCCP_SUCCESS	
SCCPERR_RESOURCES	Number of available buffers is exhausted.
CTAERR_BAD_ARGUMENT	One or more arguments are invalid.
CTAERR_DRIVER_SEND_FAILED	Error occurred accessing the CPI driver. Call **SCCPGetStats** to obtain the error code. The error code is in the failTxErr field of the SccpStats structure. The CPI error codes are listed in the *txcpi.h* include file.
CTAERR_INVALID_CTAHD	Natural Access handle is invalid.

Details

This function generates an SCCP subsystem-out-of-service-grant (SOG) to the backup signaling point as specified in the SAP configuration.

Refer to *Coordinated state change* on page 20 for more information.

SCCPCoordRqst

Requests the subsystem be taken out of service and have all traffic routed to its backup point code.

Prototype

DWORD **SCCPCoordRqst** (CTAHD ***ctahd***, S16 ***spId***, SccpCoordRqst ****coordRqst***)

Argument	Description
ctahd	Natural Access handle returned by **ctaCreateContext**.
spId	SCCP service access point.
coordRqst	Pointer to the caller's coordrqst message structure containing the affected subsystem number.

Return values

Return value	Description
SCCP_SUCCESS	
SCCPERR_RESOURCES	Number of available buffers is exhausted.
CTAERR_BAD_ARGUMENT	One or more arguments are invalid.
CTAERR_DRIVER_SEND_FAILED	Error occurred accessing the CPI driver. Call **SCCPGetStats** to obtain the error code. The error code is in the failTxErr field of the SccpStats structure. The CPI error codes are listed in the *txcpi.h* include file.
CTAERR_INVALID_CTAHD	Natural Access handle is invalid.

Details

This function generates an SCCP subsystem-out-of-service-request (SOR) to the backup signaling point as specified in the SAP configuration.

Refer to *Coordinated state change* on page 20 for more information.

SCCPDAckRqst

Reserved for future implementation.

Prototype

DWORD **SCCPDAckRqst** (CTAHD ***ctahd***, SccpConnId ****connId***, SccpDAckRqst ****dackRqst***)

Argument	Description
ctahd	Natural Access handle returned by **ctaCreateContext**.
connId	Pointer to the connection ID.
dackRqst	Pointer to the caller's data request message structure containing all parameters to be included in the message to the far end point.

Return values

Return value	Description
SCCP_SUCCESS	
SCCPERR_RESOURCES	Number of available buffers is exhausted.
CTAERR_BAD_ARGUMENT	One or more arguments are invalid.
CTAERR_DRIVER_SEND_FAILED	Error occurred accessing the CPI driver. Call **SCCPGetStats** to obtain the error code. The error code is in the failTxErr field of the SccpStats structure. The CPI error codes are listed in the *txcpi.h* include file.
CTAERR_INVALID_CTAHD	Natural Access handle is invalid.

SCCPDataRqst

Sends user data associated with an established connection to the far end point.

Prototype

DWORD **SCCPDataRqst** (CTAHD ***ctahd***, SccpConnId ****connId***, SccpDataRqst ****dataRqst***)

Argument	Description
ctahd	Natural Access handle returned by **ctaCreateContext**.
connId	Pointer to the connection ID.
dataRqst	Pointer to the caller's data request message structure containing all parameters and user data to be included in the message to the far endpoint.

Return values

Return value	Description
SCCP_SUCCESS	
SCCPERR_OVERRUN	Message generated by this request exceeds the maximum allowed size.
SCCPERR_RESOURCES	Number of available buffers is exhausted.
CTAERR_BAD_ARGUMENT	One or more arguments are invalid.
CTAERR_DRIVER_SEND_FAILED	Error occurred accessing the CPI driver. Call **SCCPGetStats** to obtain the error code. The error code is in the failTxErr field of the SccpStats structure. The CPI error codes are listed in the *txcpi.h* include file.
CTAERR_INVALID_CTAHD	Natural Access handle is invalid.

Details

Based on the service class in use on the specified connection, the SCCP layer generates either data form 1 (DT1) or data form 2 (DT2) packets, including any necessary segmentation.

The maximum length of the data field included in the SccpDataRqst structure is MAX_DATA_SZ bytes. The SCCP layer performs any necessary segmentation based on the maximum allowable packet size for the chosen network service access point.

Refer to *Transferring connection-oriented data* on page 38 for more information.

SCCPEDataRqst

Sends an expedited data message associated with an established connection to the far end point.

Prototype

DWORD **SCCPEDataRqst** (CTAHD ***ctahd***, SccpConnId ****connId***, SccpDataRqst ****dataRqst***)

Argument	Description
ctahd	Natural Access handle returned by **ctaCreateContext**.
connId	Pointer to the connection ID.
dataRqst	Pointer to the caller's data request message structure containing all parameters and user data to be included in the message to the far end point.

Return values

Return value	Description
SCCP_SUCCESS	
SCCPERR_OVERRUN	Message generated by this request exceeds the maximum allowed size.
SCCPERR_RESOURCES	Number of available buffers is exhausted.
CTAERR_BAD_ARGUMENT	One or more arguments are invalid.
CTAERR_DRIVER_SEND_FAILED	Error occurred accessing the CPI driver. Call **SCCPGetStats** to obtain the error code. The error code is in the failTxErr field of the SccpStats structure. The CPI error codes are listed in the *txcpi.h* include file.
CTAERR_INVALID_CTAHD	Natural Access handle is invalid.

Details

The maximum length of the data field included in the SccpDataRqst structure is 264 bytes, as per ANSI T1.112 recommendations.

Refer to *Transferring connection-oriented data* on page 38 for more information.

SCCPGetStats

Retrieves congestion level activity statistics from the SCCP service.

Prototype

DWORD **SCCPGetStats** (CTAHD ***ctahd***, SccpStats ****stats***, U8 ***bReset***)

Argument	Description
ctahd	Natural Access handle returned by **ctaCreateContext**.

stats	Pointer to a user-supplied SccpStats structure:
	`typedef struct _SCCP_STAT`
	`{`
	`U32 succTx; /* number of successful transmits */`
	`U32 succRx; /* number of successful receives */`
	`U32 failTx; /* number of transmit failures */`
	`U32 failTxErr; /* last transmit failure error */`
	`U32 failRx; /* number of receive failures */`
	`U32 failRxErr; /* last receive failure error */`
	`U32 outStTx; /* Number of messages outstanding to the board */`
	`U32 maxOutStTx; /* Maximum number of messages outstanding to the board */`
	`U32 quTx; /* Number of messages queued by the SCCP service */`
	`U32 maxQuTx; /* Maximum number of messages queued by the SCCP service */`
	`U8 currCongState; /* Current outbound congestion state */`
	`U8 svcCongState; /* Service congestion state(0-3) */`
	`U8 taskCongState; /* last reported SCCP task congestion state(0-3) */`
	`} SccpStats;`

bReset	If non-zero, statistics are set to zero after retrieval.

Return values

Return value	Description
SCCP_SUCCESS	
SCCPERR_DRIVER	Statistics could not be retrieved from the Natural Access service.
SCCPERR_RESOURCES	Number of available buffers is exhausted.
CTAERR_BAD_ARGUMENT	One or more arguments are invalid.
CTAERR_INVALID_CTAHD	Natural Access handle is invalid.

Details

This function returns SCCP service statistics for the Natural Access handle provided.

Refer to *Handling congestion events* on page 40 for more information.

SCCPInactResp

Called in response to an SCCPINACTIND event received from the SCCP layer to indicate that the specified connection is still active.

Prototype

DWORD **SCCPInactResp** (CTAHD ***ctahd***, SccpConnId ****connId***)

Argument	Description
ctahd	Natural Access handle returned by **ctaCreateContext**.
connId	Pointer to the connection ID.

Return values

Return value	Description
SCCP_SUCCESS	
SCCPERR_RESOURCES	Number of available buffers is exhausted.
CTAERR_DRIVER_SEND_FAILED	Error occurred accessing the CPI driver. Call **SCCPGetStats** to obtain the error code. The error code is in the failTxErr field of the SccpStats structure. The CPI error codes are listed in the *txcpi.h* include file.
CTAERR_INVALID_CTAHD	Natural Access handle is invalid.

Details

The specified connection ID parameter must be identical to the connection ID returned in the SccpRcvInfoBlk with the SCCPINACTIND event.

If the connection ID returned with the SCCPINACTIND event identifies a connection that the application does not believe is active, call **SCCPReleaseRqst** to release the connection towards the far end and free up connection resources in the SCCP layer.

Refer to *Application inactivity control* on page 30 for more information.

SCCPReleaseRqst

Clears an SCCP connection by generating a released message to the far exchange.

Prototype

DWORD **SCCPReleaseRqst** (CTAHD ***ctahd***, SccpConnId ****connId***, SccpReleased ****relRqst***)

Argument	Description
ctahd	Natural Access handle returned by **ctaCreateContext**.
connId	Pointer to the connection ID.
relRqst	Pointer to the caller's released message structure containing all parameters to be included in the message to the far endpoint.

Return values

Return value	Description
SCCP_SUCCESS	
SCCPERR_OVERRUN	Message generated by this request exceeds the maximum allowed size.
SCCPERR_RESOURCES	Number of available buffers is exhausted.
CTAERR_BAD_ARGUMENT	One or more arguments are invalid.
CTAERR_DRIVER_SEND_FAILED	Error occurred accessing the CPI driver. Call **SCCPGetStats** to obtain the error code. The error code is in the failTxErr field of the SccpStats structure. The CPI error codes are listed in the *txcpi.h* include file.
CTAERR_INVALID_CTAHD	Natural Access handle is invalid.

Details

There is no response to this request from the SCCP layer or far end point. The application is not notified of receipt of a release complete message from the far end point.

Refer to *Establishing connections* on page 37 for more information.

SCCPResetResp

Acknowledges that a reset request indication was received from the far end point of an SCCP connection.

Prototype

DWORD **SCCPResetResp** (CTAHD ***ctahd***, SccpConnId ****connId***)

Argument	Description
ctahd	Natural Access handle returned by **ctaCreateContext**.
connId	Pointer to the connection ID.

Return values

Return value	Description
SCCP_SUCCESS	
SCCPERR_RESOURCES	Number of available buffers is exhausted.
CTAERR_BAD_ARGUMENT	One or more arguments are invalid.
CTAERR_DRIVER_SEND_FAILED	Error occurred accessing the CPI driver. Call **SCCPGetStats** to obtain the error code. The error code is in the failTxErr field of the SccpStats structure. The CPI error codes are listed in the *txcpi.h* include file.
CTAERR_INVALID_CTAHD	Natural Access handle is invalid.

Details

If the application does not want to accept a reset of the connection, use **SCCPReleaseRqst** instead of **SCCPResetResp** to clear the connection.

SCCPResetRqst

Sends a reset message to the far end point and starts the reset timer.

Prototype

DWORD **SCCPResetRqst** (CTAHD ***ctahd***, SccpConnId ****connId***, SccpResetRqst ****resetRqst***)

Argument	Description
ctahd	Natural Access handle returned by **ctaCreateContext**.
connId	Pointer to the connection ID identifying the SAP and connection instance to which this request belongs.
resetRqst	Pointer to the caller's reset request message containing all parameters to be included in the message to the far end point.

Return values

Return value	Description
SCCP_SUCCESS	
SCCPERR_RESOURCES	Number of available buffers is exhausted.
CTAERR_BAD_ARGUMENT	One or more arguments are invalid.
CTAERR_DRIVER_SEND_FAILED	Error occurred accessing the CPI driver. Call **SCCPGetStats** to obtain the error code. The error code is in the failTxErr field of the SccpStats structure. The CPI error codes are listed in the *txcpi.h* include file.
CTAERR_INVALID_CTAHD	Natural Access handle is invalid.

Details

If the SCCP layer does not receive a reset confirmation message before the expiration of the timer, it clears the connection in both directions.

SCCPRetrieveMessage

Retrieves the next message from the SCCP layer.

Prototype

DWORD **SCCPRetrieveMessage** (CTAHD ***ctahd***, SccpAllMsgs ****message***, SccpRcvInfoBlk ****infoBlk***, Bool ***wait***)

Argument	Description
ctahd	Natural Access handle returned by **ctaCreateContext**.
message	Pointer to the address of the caller's message buffer where the received message (if any) is returned to the caller.

infoBlk	Pointer to the address of the caller's receive information block where information regarding the received message (if any) is returned to the caller. The structure of the receive information block is as follows:
	`typedef struct rcvInfoBlk`
	`{`
	`  U8  board;  /* sending TX board number */`
	`  U8  indType;  /* indication/confirm type */`
	`  U8  evntType;  /* event type for status indications */`
	`  U8  spare1;  /* spare for alignment */`
	`  S16  suId;  /* service user id - connectionless`
	`  * msgs only */`
	`  U16  spare2;  /* spare for alignment */`
	`  SccpConnId connId;  /* connection ID - connection-oriented`
	`  * messages only */`
	`  U32  opc;  /* originating point code */`
	`} RcvInfoBlk;`
	Refer to the Details section for more information.

wait	Not used.

Return values

Return value	Description
SCCP_SUCCESS	
SCCPERR_NOMSG	No event messages waiting.
CTAERR_BAD_ARGUMENT	One or more arguments are invalid.
CTAERR_DRIVER_RECEIVE_FAILED	Service could not communicate with the device driver.
CTAERR_INVALID_CTAHD	Natural Access handle is invalid.

Details

SCCPRetrieveMessage receives events (messages) from the SCCP layer. Refer to *Establishing connections* on page 37 for more information.

When a message is received, **SCCPRetrieveMessage** copies the event to the caller's event buffer and performs any necessary byte order translations to convert to the host's native byte ordering. Information about the received message is returned to the caller in the ***infoBlk*** parameter.

The indication type (indType) identifies the event received and is coded to one of the following values:

SCCPCONNCFM	0xB8	Connect confirm
SCCPCONNIND	0xB7	Connect indication
SCCPRELIND	0xBE	Release indication
SCCPDATIND	0xB9	Data (DT1 or DT2) indication
SCCPUDATIND	0xB1	Unitdata indication
SCCPEDATIND	0xBA	Expedited data indication
SCCPDACKIND	0xBD	Data acknowledge indication
SCCPRESETCFM	0xBC	Connection reset confirm
SCCPRESETIND	0xBB	Connection reset indication
SCCPSTAIND	0xB2	Unsolicited status indication
SCCPCOORDIND	0xB3	Coordinated out of service (OOS) indication
SCCPCOORDCFM	0xB4	Coordinated out of service (OOS) confirm
SCCPSTATEIND	0xB5	Subsystem state indication
SCCPPCSTIND	0xB6	Point code state indication
SCCPCONNAUDCFM	0xBF	Connection audit confirmation
SCCPINACTIND	0xE0	Connection inactivity indication
SCCPRUNSTATEIND	0xEF	Run state (primary/backup/standalone) indication

The evntType field identifies the actual message received for status and connection status indications.

Note: The application must save the service provider instance id (spInstId) field from the first event received from the SCCP layer for each connection and use it in subsequent requests associated with that connection.

The event structure associated with a received message depends on the type of message received from the SCCP layer. Refer to *Event reference overview* on page 137 for more information.

SCCPStateRqst

Notifies all concerned point codes of a change in its subsystem state. For more information, refer to *Subsystem state changes* on page 21.

Prototype

DWORD **SCCPStateRqst** (CTAHD ***ctahd***, S16 ***spId***, U8 ***aSsn***, U8 ***status***)

Argument	Description
ctahd	Natural Access handle returned by **ctaCreateContext**.
spId	SCCP service access point.
aSsn	Affected subsystem number.
status	New subsystem status: 0x03 = SS_OOS Subsystem out of service 0x04 = SS_IS Subsystem in service

Return values

Return value	Description
SCCP_SUCCESS	
SCCPERR_RESOURCES	Number of available buffers is exhausted.
CTAERR_BAD_ARGUMENT	One or more arguments are invalid.
CTAERR_DRIVER_SEND_FAILED	Error occurred accessing the CPI driver. Call **SCCPGetStats** to obtain the error code. The error code is in the failTxErr field of the SccpStats structure. The CPI error codes are listed in the *txcpi.h* include file.
CTAERR_INVALID_CTAHD	Natural Access handle is invalid.

SCCPUDataRqst

Sends a datagram to the destination specified in the calledPty parameter of the SCCPUDataRqst message.

Prototype

DWORD **SCCPUDataRqst** (CTAHD ***ctahd***, S16 ***spId***, SccpUdataRqst ****dataRqst***)

Argument	Description
ctahd	Natural Access handle returned by **ctaCreateContext**.
spId	SCCP service access point.
dataRqst	Pointer to the caller's data request message structure containing all parameters and user data to be included in the message to the far end point.

Return values

Return value	Description
SCCP_SUCCESS	
SCCPERR_OVERRUN	Message generated by this request exceeds the maximum allowed size.
SCCPERR_RESOURCES	Number of available buffers is exhausted.
CTAERR_BAD_ARGUMENT	One or more arguments are invalid.
CTAERR_DRIVER_SEND_FAILED	Error occurred accessing the CPI driver. Call **SCCPGetStats** to obtain the error code. The error code is in the failTxErr field of the SccpStats structure. The CPI error codes are listed in the *txcpi.h* include file.
CTAERR_INVALID_CTAHD	Natural Access handle is invalid.

Details

Based on the service class requested and the size of the data field, the SCCP layer generates either unitdata (UDT) or extended unitdata (XUDT) packet(s).

The maximum length of the data field included in the SccpUdataRqst structure is MAX_DATA_SZ bytes.

Refer to *Transferring connectionless data* on page 36 for more information.

6 SCCP management function reference

SCCP management function summary

NMS SCCP consists of the following management functions:

- Configuration functions
- Control functions
- Statistic functions
- Status functions

Configuration functions

Function	Description
SccpDelAddrCfg	Deletes the current configuration parameter values for a specific SCCP global title specified by the ***pGtIn*** parameter.
SccpDelRteCfg	Deletes the current route definition for an SCCP route specified by the ***dpc*** parameter.
SccpGetAddrCfg	Retrieves the current configuration parameter values for an SCCP global title specified by the ***pGtIn*** parameter.
SccpGetGenCfg	Retrieves the current values for the general configuration parameters from the TX board.
SccpGetNSapCfg	Retrieves the current configuration parameter values for a specific SCCP network service access point (NSAP) on the specified TX board.
SccpGetRteCfg	Retrieves the current configuration parameter values for an SCCP route definition specified by the ***dpc*** parameter.
SccpGetUSapCfg	Retrieves the current configuration parameter values for a specific SCCP user service access point on the specified TX board.
SccpInitAddrCfg	Initializes an SCCP global title translation definition to its default values that can be passed to **SccpSetAddrCfg**.
SccpInitGenCfg	Initializes an SCCP general configuration buffer to default values that can be passed to **SccpSetGenCfg**.
SccpInitNSapCfg	Initializes an SCCP network service access point (NSAP) configuration buffer to default values that can be passed to **SccpSetNSapCfg**.
SccpInitRteCfg	Initializes an SCCP route definition that can be passed to **SccpSetRteCfg**.
SccpInitUSapCfg	Initializes an SCCP user service access point configuration buffer to default values that can be passed to **SccpSetUSapCfg**.
SccpSetAddrCfg	Sends an SCCP global title translation parameter block to the specified TX board.
SccpSetGenCfg	Sends the SCCP general configuration parameters to the TX board.
SccpSetNSapCfg	Sends an SCCP network service access point (NSAP) configuration parameter block to the specified TX board to define or update the configuration for a specific SCCP NSAP.
SccpSetRteCfg	Sends an SCCP route definition parameter block to the specified TX board.
SccpSetUSapCfg	Sends an SCCP service access point (SAP) configuration parameter block to the specified TX board to define or update the configuration for a specific SCCP SAP.

Control functions

Function	Description
SccpAlarmControl	Controls the level of alarms generated by the SCCP task on the TX board.
SccpAsciiMaskToBcd	Converts the null-terminated ASCII address mask pointed to by ***pAscii*** to binary coded hexadecimal format.
SccpAsciiNumToBcd	Converts the null-terminated ASCII string of decimal digits pointed to by ***pAscii*** to binary coded decimal format.
SccpBcdMaskToAscii	Converts the binary coded hexadecimal mask pointed to by ***pBcd*** to an ASCII string of digits.
SccpBcdNumToAscii	Converts the binary coded decimal number pointed to by ***pBcd*** to an ASCII string of digits.
SccpMgmtInit	Initializes SCCP management and provides access to the TX device driver.
SccpMgmtTerm	Terminates the connection between the application with the TX device driver, releasing any resources (such as file descriptors) associated with SCCP management.
SccpTraceControl	Sends a request to enable or disable tracing of SCCP protocol messages. Trace control is not currently implemented. No trace messages are generated, regardless of the **SccpTraceControl** setting.

Statistics functions

Function	Description
SccpGetGenStats	Retrieves and optionally resets the general statistics for the SCCP task.
SccpGetNSapStats	Retrieves and optionally resets the SCCP network service access point (NSAP) statistics for a specified network service access point of the SCCP task.
SccpGetUSapStats	Retrieves and optionally resets the user service access point statistics for a particular user service access point of the SCCP task.

Status function

Function	Description
SccpGetUSapStatus	Retrieves the user service access point status for a specified user service access point ID.

Using the SCCP management function reference

This section provides an alphabetical reference to the SCCP management functions. A prototype of each function is shown with the function description and details of all arguments and return values. A typical function includes:

Prototype	The prototype is followed by a listing of the function arguments. NMS 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 shown.
Return values	The return value for a function is either SCCP_SUCCESS or an error code. For asynchronous functions, a return value of SCCP_SUCCESS (zero) indicates the function was initiated; subsequent events indicate the status of the operation.

Unlike the SCCP service functions that send and receive messages asynchronously, each SCCP management function generates a request followed immediately by a response from the TX board. SCCP management functions block the calling application waiting for this response for a maximum of five seconds, but typically a few hundred milliseconds and return an indication as to whether or not an action was completed successfully. For this reason, the SCCP management functions are typically used by one or more management applications, separate from the applications that use the SCCP service. SCCP management is packaged as a separate library with its own interface header files.

SccpAlarmControl

Controls the level of alarms generated by the SCCP task on the TX board.

Prototype

SCCP_STATUS **SccpAlarmControl** (U8 ***board***, U8 ***alarmLvl***)

Argument	Description
board	TX board number.
alarmLvl	Desired new alarm level. SCCP defines the following alarm levels. Refer to the Details section for more information. 0 = SCCP_ALARMLVL_DISABLE 1 = SCCP_ALARMLVL_DEFAULT 2 = SCCP_ALARMLVL_DEBUG 3 = SCCP_ALARMLVL_DETAIL

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BOARD	***board*** is out of range.
SCCP_DRIVER	Error occurred accessing the driver.
SCCP_NOGENCFG	**SccpSetGenCfg** has not been called.
SCCP_RANGE	***alarmLvl*** is out of range.
SCCP_TIMEOUT	Request timed out.
SCCP_UNINIT	Application failed to call **SccpMgmtInit** prior to this call.

Details

All SCCP alarm messages are sent to the *txalarm* utility. The following table defines the SCCP alarm levels and their recommended use:

Alarm level	Description
SCCP_ALARMLVL_DISABLE	All alarms are disabled. This alarm level is not recommended for use.
SCCP_ALARMLVL_DEFAULT	Default alarm level. Service impacting events such as application/SSN unavailable, resource failures, and application interface violations are logged.
SCCP_ALARMLVL_DEBUG	All default level 1 alarms plus detailed message encoding or decoding diagnostics to troubleshoot individual application or transaction problems. Use this alarm level for application development.
SCCP_ALARMLVL_DETAIL	All debug level 2 alarms plus some normal events to help isolate network or application problems. Do not use this alarm level under production load conditions.

SccpAsciiMaskToBcd

Converts the null-terminated ASCII address mask pointed to by ***pAscii*** to binary coded hexadecimal format.

Prototype

SCCP_STATUS **SccpAsciiMaskToBcd** (char ***pAscii**, U8 ***pBcdBuf**, U32 ***nBcdBufLen**, U32 ***pnDigits**)

Argument	Description
pAscii	Pointer to an address of a null-terminated ASCII string of digits.
pBcdBuf	Pointer to an address of a buffer for the converted BCD mask.
nBcdBufLen	Number of bytes in the BCD buffer.
pnDigits	Pointer to an address where number of digits converted is returned.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BADDIGIT	ASCII string contains characters other than 0 and f.
SCCP_BUFLEN	BCD buffer is not large enough to hold the converted mask.
SCCP_NULLPTR	Null pointer was passed as an input parameter.

Details

The results are placed in the buffer pointed to by ***pBcdBuf***. If the length of the ASCII string is less than half the length of the BCD buffer, the BCD buffer is padded with BCD zeros. The number of digits converted is returned at the address ***pnDigits***.

SccpAsciiNumToBcd

Converts the null-terminated ASCII string of decimal digits pointed to by ***pAscii*** to binary coded decimal format.

Prototype

SCCP_STATUS **SccpAsciiNumToBcd** (char ***pAscii**, U8 ***pBcdBuf**, U32 ***nBcdBufLen**, U32 ***pnDigits**)

Argument	Description
pAscii	Pointer to an address of a null-terminated ASCII string of digits.
pBcdBuf	Pointer to an address of a buffer for the converted BCD number.
nBcdBufLen	Number of bytes in the BCD buffer.
pnDigits	Pointer to an address where number of digits converted is returned.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BADDIGIT	ASCII string contains characters other than digits.
SCCP_BUFLEN	BCD buffer is not large enough to hold the converted number.
SCCP_NULLPTR	Null pointer was passed as an input parameter.

Details

The results are placed in the buffer pointed to by ***pBcdBuf***. If the length of the ASCII string is less than half the length of the BCD buffer, the BCD buffer is padded with BCD zeros. The number of digits converted is returned at the address ***pnDigits***.

SccpBcdMaskToAscii

Converts the binary coded hexadecimal mask pointed to by ***pBcd*** to an ASCII string of digits.

Prototype

SCCP_STATUS **SccpBcdMaskToAscii** (U8 ***pBcd***, U32 ***nDigits***, char ***pAsciiBuf***, U32 ***nAsciiBufLen***)

Argument	Description
pBcd	Pointer to an address of a binary coded hexadecimal mask.
nDigits	Number of digits contained in the mask.
pAsciiBuf	Pointer to an address of a buffer for the converted mask.
nAsciiBufLen	Size of the output buffer in bytes.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BADDIGIT	Mask contains values other than zero through nine.
SCCP_BUFLEN	ASCII buffer is not large enough to hold the converted mask.
SCCP_NULLPTR	Null pointer was passed as an input parameter.

Details

The results are placed in the buffer pointed to by ***pAsciiBuf***. If the number of digits is less than the size of the ASCII buffer, the buffer is padded with null characters.

Note: The ASCII string does not end in a null-termination character when the buffer size is exactly the same as the number of digits.

SccpBcdNumToAscii

Converts the binary coded decimal number pointed to by ***pBcd*** to an ASCII string of digits.

Prototype

SCCP_STATUS **SccpBcdNumToAscii** (U8 ****pBcd***, U32 ***nDigits***, char ****pAsciiBuf***, U32 ***nAsciiBufLen***)

Argument	Description
pBcd	Pointer to an address of a binary coded decimal number.
nDigits	Number of digits contained in the number.
pAsciiBuf	Pointer to an address of a buffer for the converted number.
nAsciiBufLen	Size of the output buffer in bytes.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BADDIGIT	BCD number contains values other than zero through nine.
SCCP_BUFLEN	ASCII buffer is not large enough to hold the converted number.
SCCP_NULLPTR	Null pointer was passed as an input parameter.

Details

The results are placed in the buffer pointed to by ***pAsciiBuf***. If the number of digits is less than the size of the ASCII buffer, the buffer is padded with null characters.

Note: The ASCII string does not end in a null-termination character when the buffer size is exactly the same as the number of digits.

SccpDelAddrCfg

Deletes the current configuration parameter values for a specific SCCP global title specified by the ***pGtIn*** parameter.

Prototype

SCCP_STATUS **SccpDelAddrCfg** (U8 ***board***, U8 ****pGtIn***, U8 ***nBytes***)

Argument	Description
board	TX board number.
pGtIn	Pointer to a buffer containing binary coded decimal digits that define the global title to be retrieved.
nBytes	Number of bytes in BCD encoded global title.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BOARD	***board*** is out of range.
SCCP_BUFLEN	***nBytes*** is out of range.
SCCP_DRIVER	Error occurred accessing the driver.
SCCP_NOGENCFG	**SccpSetGenCfg** has not been called.
SCCP_NOTFOUND	Global title specified by ***pGtIn*** was not found.
SCCP_NULLPTR	Null pointer was specified for ***pGtIn***.
SCCP_RESOURCES	TX board resource is exhausted. Check the *txalarm* log.
SCCP_TIMEOUT	Request timed out.
SCCP_UNINIT	Application failed to call **SccpMgmtInit** prior to this call.

SccpDelRteCfg

Deletes the current route definition for an SCCP route specified by the ***dpc*** parameter.

Prototype

SCCP_STATUS **SccpDelRteCfg** (U8 ***board***, U32 ***dpc***)

Argument	Description
board	TX board number.
dpc	Destination point code.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BOARD	***board*** is out of range.
SCCP_DRIVER	Error occurred accessing the driver.
SCCP_NOGENCFG	**SccpSetGenCfg** has not been called.
SCCP_NOTFOUND	Route specified by the ***dpc*** parameter was not found.
SCCP_RESOURCES	TX board resource is exhausted. Check the *txalarm* log.
SCCP_TIMEOUT	Request timed out.
SCCP_UNINIT	Application failed to call **SccpMgmtInit** prior to this call.

SccpGetAddrCfg

Retrieves the current configuration parameter values for an SCCP global title specified by the ***pGtIn*** parameter.

Prototype

SCCP_STATUS **SccpGetAddrCfg** (U8 ***board***, SccpAddrMapCfg ****pCfg***, U8 ****pGtIn***, U8 ***nBytes***)

Argument	Description
board	TX board number.
pCfg	Pointer to the address of a global title translation parameters buffer. The format is specified in **SccpInitAddrCfg**.
pGtIn	Pointer to a buffer containing binary coded decimal digits that define the global title to be retrieved.
nBytes	Number of bytes in BCD encoded global title.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BOARD	***board*** is out of range.
SCCP_BUFLEN	***nBytes*** is out of range.
SCCP_DRIVER	Error occurred accessing the driver.
SCCP_NOGENCFG	**SccpSetGenCfg** has not been called.
SCCP_NOTFOUND	Global title specified by ***pGtIn*** was not found.
SCCP_NULLPTR	Null pointer was specified for either ***pCfg*** or ***pGtIn***.
SCCP_TIMEOUT	Request timed out.
SCCP_UNINIT	Application failed to call **SccpMgmtInit** prior to this call.

SccpGetGenCfg

Retrieves the current values for the general configuration parameters from the TX board.

Prototype

SCCP_STATUS **SccpGetGenCfg** (U8 ***board***, SccpGenCfg ****pCfg***)

Argument	Description
board	TX board number.
pCfg	Pointer to the address of the general configuration parameters buffer. The format is specified in **SccpInitGenCfg**.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BOARD	***board*** is out of range.
SCCP_DRIVER	Error occurred accessing the driver.
SCCP_NOGENCFG	Application failed to call **SccpSetGenCfg** prior to this call.
SCCP_NULLPTR	Null pointer was specified for ***pCfg***.
SCCP_RESOURCES	TX board resource is exhausted. Check the *txalarm* log.
SCCP_TIMEOUT	Request timed out.
SCCP_UNINIT	Application failed to call **SccpMgmtInit** prior to this call.

SccpGetGenStats

Retrieves and optionally resets the general statistics for the SCCP task.

Prototype

SCCP_STATUS **SccpGetGenStats** (U8 ***board***, SccpGenStats ****pStats***, U8 ***bReset***)

Argument	Description
board	TX board number to which this request is directed.
pStats	Pointer to the address of the caller's SCCP general statistics buffer where statistics are returned. Refer to the Details section for more information.
bReset	If non-zero, statistics are reset to zero after retrieval.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BOARD	***board*** is out of range.
SCCP_DRIVER	Error occurred accessing the driver.
SCCP_NOGENCFG	**SccpSetGenCfg** has not been called.
SCCP_NULLPTR	Null pointer was specified for ***pStats***.
SCCP_RESOURCES	TX board resource is exhausted. Check the *txalarm* log.
SCCP_TIMEOUT	Request timed out.
SCCP_UNINIT	Application failed to call **SccpMgmtInit** prior to this call.

Details

The SccpGenStats structure contains the following fields:

```
typedef struct
{
  U8         month;         /* month                          */
  U8         day;           /* day                            */
  U8         year;          /* year - since 1900              */
  U8         hour;          /* hour - 24 hour clock           */
  U8         min;           /* minute                         */
  U8         sec;           /* second                         */
  U8         tenths;        /* tenths of second               */
  U8         fill;          /* alignment                      */
} DateTime;

typedef struct
{
  DateTime dt;           /* Date and Time                    */
  Duration dura;         /* Duration (not used)              */
                         /* Routing failure                  */
  S32      rfNTASN;      /* no translation for address of
                          * such nature                      */
  S32      rfNTSA;       /* no translation for this
                          * specific address                 */
  S32      rfNetFail;    /* network failure (point code
                          * unavailable)                     */
  S32      rfNetCong;    /* network congestion               */
  S32      rfSsnFail;    /* subsystem failure                */
  S32      rfSsnCong;    /* subsystem congestion             */
  S32      rfUnequip;    /* unequipped user                  */
  S32      rfHopViolate; /* Hop counter violation            */
  S32      synError;     /* Syntax Error                     */
  S32      rfUnknown;    /* reason unknown                   */
  S32      uDataTx;      /* unit data sent                   */
  S32      uDataSrvTx;   /* unit data service sent           */
  S32      uDataRx;      /* unit data received               */
  S32      uDataSrvRx;   /* unit data service received       */
  S32      xuDataTx;     /* extended unit data sent          */
  S32      xuDataSrvTx;  /* extended unit data service sent  */
  S32      xuDataRx;     /* extended unit data received      */
  S32      xuDataSrvRx;  /* extended unit data service
                          * received                         */
  S32      msgHand;      /* total msgs handled               */
  S32      msgLoc;       /* total msgs intended for local
                          * subsystems                       */
  S32      gttReq;       /* msgs requiring GT  translation   */
  S32      msgTxC0;      /* total msgs sent, class 0         */
  S32      msgTxC1;      /* total msgs sent, class 1         */
  S32      msgRxC0;      /* total msgs received, class 0     */
  S32      msgRxC1;      /* total msgs received, class 1     */
  U8       spRunState;   /* current redundancy state         */
  U8       spMateState;  /* current mate state
                          * (connected/isolated              */
  U8       memCongLevel; /* task memory congestion level     */
  U8       spare1;       /* spare for alignment              */
  S32      inCongDisc;   /* inbound messages discarded due
                          * to congestion                    */
} SccpGenStats;
```

Counts of type S32 are the number of event occurrences since the statistics were last cleared.

SccpGetNSapCfg

Retrieves the current configuration parameter values for a specific SCCP network service access point (NSAP) on the specified TX board.

Prototype

SCCP_STATUS **SccpGetNSapCfg** (U8 ***board***, SccpNSapCfg ****pCfg***, U16 ***sapId***)

Argument	Description
board	TX board number to which this request is directed.
pCfg	Pointer to the address of a SAP configuration parameters buffer. The format is specified in **SccpInitNSapCfg**.
sapId	SCCP service access point ID.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BOARD	***board*** is out of range.
SCCP_DRIVER	Error occurred accessing the driver.
SCCP_EXCEEDMAXCFG	***sapId*** exceeds the maximum specified through **SccpSetGenCfg**.
SCCP_NOGENCFG	**SccpSetGenCfg** has not been called.
SCCP_NOTFOUND	Application failed to call **SccpSetNSapCfg** prior to this call.
SCCP_NULLPTR	Null pointer was specified for ***pCfg***.
SCCP_TIMEOUT	Request timed out.
SCCP_UNINIT	Application failed to call **SccpMgmtInit** prior to this call.

SccpGetNSapStats

Retrieves and optionally resets the SCCP network service access point (NSAP) statistics for a specified network service access point of the SCCP task.

Prototype

SCCP_STATUS **SccpGetNSapStats** (U8 ***board***, SccpNSapStats ****pStats***, U16 ***sapId***, U8 ***bReset***)

Argument	Description
board	TX board number.

pStats	Pointer to the address of the caller's SCCP network service access point statistics buffer where statistics are returned:
	`typedef struct`
	`{`
	`  U8  month;  /* month */`
	`  U8  day;  /* day */`
	`  U8  year;  /* year - since 1900 */`
	`  U8  hour;  /* hour - 24 hour clock */`
	`  U8  min;  /* minute */`
	`  U8  sec;  /* second */`
	`  U8  tenths;  /* tenths of second */`
	`  U8  fill;  /* alignment */`
	`} DateTime;`
	
	`typedef struct`
	`{`
	`  DateTime dt;  /* Date and Time */`
	`  Duration dura;  /* Duration (not used) */`
	`  S32  ssAllTx;  /* SS allowed transmitted */`
	`  S32  ssOutGTx;  /* SS out of service grant transmitted */`
	`  S32  ssOutRTx;  /* SS out of service request`
	`  * transmitted */`
	`  S32  ssProhTx;  /* SS prohibited transmitted */`
	`  S32  ssStatTx;  /* SS status test transmitted */`
	`  S32  ssAllRx;  /* SS allowed received */`
	`  S32  ssOutGRx;  /* SS out of service grant received */`
	`  S32  ssOutRRx;  /* SS out of service request received */`
	`  S32  ssProhRx;  /* SS prohibited received */`
	`  S32  ssStatRx;  /* SS status test received */`
	`  S32  connActive; /* number of currently active`
	`  * connections */`
	`} SccpNSapStats;`
	
	Counts of type S32 are the number of event occurrences since the statistics were last cleared.

sapId	Network service access point ID.
bReset	If non-zero, statistics are reset to zero after retrieval.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BOARD	***board*** is out of range.
SCCP_DRIVER	Error occurred accessing the driver.
SCCP_EXCEEDMAXCFG	Network service access point ID is out of the configured range.
SCCP_NOGENCFG	**SccpSetGenCfg** has not been called.

Return value	Description
SCCP_NULLPTR	Null pointer was specified for ***pStats***.
SCCP_RESOURCES	TX board resource is exhausted. Check the *txalarm* log.
SCCP_TIMEOUT	Request timed out.
SCCP_UNINIT	Application failed to call **SccpMgmtInit** prior to this call.

SccpGetRteCfg

Retrieves the current configuration parameter values for an SCCP route definition specified by the ***dpc*** parameter.

Prototype

SCCP_STATUS **SccpGetRteCfg** (U8 ***board***, SccpRouteCfg ****pCfg***, U32 ***dpc***)

Argument	Description
board	TX board number to which this request is directed.
pCfg	Pointer to the address of a route definition parameters buffer. The format is specified in **SccpInitRteCfg**.
dpc	Destination point code.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BOARD	***board*** is out of range.
SCCP_DRIVER	Error occurred accessing the driver.
SCCP_NOGENCFG	**SccpSetGenCfg** has not been called.
SCCP_NOTFOUND	Route specified by the ***dpc*** parameter was not found.
SCCP_NULLPTR	Null pointer was specified for ***pCfg***.
SCCP_TIMEOUT	Request timed out.
SCCP_UNINIT	Application failed to call **SccpMgmtInit** prior to this call.

SccpGetUSapCfg

Retrieves the current configuration parameter values for a specific SCCP user service access point on the specified TX board.

Prototype

SCCP_STATUS **SccpGetUSapCfg** (U8 ***board***, SccpUSapCfg ****pCfg***, U16 ***sapId***)

Argument	Description
board	TX board number to which this request is directed.
pCfg	Pointer to the address of the service access point configuration parameters buffer. The format is specified in **SccpInitUSapCfg**.
sapId	Service access point ID being retrieved.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BOARD	***board*** is out of range.
SCCP_DRIVER	Error occurred accessing the driver.
SCCP_EXCEEDMAXCFG	***sapId*** exceeds the maximum specified through **SccpSetGenCfg**.
SCCP_NOGENCFG	**SccpSetGenCfg** has not been called.
SCCP_NOTFOUND	Application failed to call **SccpSetUSapCfg** prior to this call.
SCCP_NULLPTR	Null pointer was specified for ***pCfg***.
SCCP_TIMEOUT	Request timed out.
SCCP_UNINIT	Application failed to call **SccpMgmtInit** prior to this call.

SccpGetUSapStats

Retrieves and optionally resets the user service access point statistics for a particular user service access point of the SCCP task.

Prototype

SCCP_STATUS **SccpGetUSapStats** (U8 ***board***, SccpUSapStats ****pStats***, U16 ***sapId***, U8 ***bReset***)

Argument	Description
board	TX board number.

pStats	Pointer to the address of the caller's SCCP user service access point statistics buffer where statistics are returned:
	`typedef struct`
	`{`
	`  U8  month;  /* month */`
	`  U8  day;  /* day */`
	`  U8  year;  /* year - since 1900 */`
	`  U8  hour;  /* hour - 24 hour clock */`
	`  U8  min;  /* minute */`
	`  U8  sec;  /* second */`
	`  U8  tenths;  /* tenths of second */`
	`  U8  fill;  /* alignment */`
	`} DateTime;`
	
	`typedef struct`
	`{`
	`  DateTime  dt;  /* Date and Time */`
	`  Duration  dura;  /* Duration (not used) */`
	`  S32  ssOOSReqGr;  /* Subsystem out-of-service * request granted */`
	`  S32  ssOOSReqDn;  /* Subsystem out-of-service * request denied */`
	`  S32  msgTxBSS;  /* msgs sent to backup * subsystem */`
	`  S32  inCongDisc;  /* inbound messages discarded * due to congestion */`
	`  S32  inCongRtnd;  /* inbound messages returned * due to congestion */`
	`  S32  outCongDisc;  /* outbound messages discarded * due to congestion */`
	`  S32  outCongRtnd;  /* outbound messages returned * due to congestion */`
	`} SccpUSapStats;`
	Counts of type S32 are the number of event occurrences since the statistics were last cleared.

sapId	User service access point ID.
bReset	If non-zero, statistics are reset to zero after retrieval.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BOARD	***board*** is out of range.
SCCP_DRIVER	Error occurred accessing the driver.
SCCP_EXCEEDMAXCFG	User service access point ID is out of the configured range.

Return value	Description
SCCP_NOGENCFG	**SccpSetGenCfg** has not been called.
SCCP_NULLPTR	Null pointer was specified for ***pStats***.
SCCP_RESOURCES	TX board resource is exhausted. Check the *txalarm* log.
SCCP_TIMEOUT	Request timed out.
SCCP_UNINIT	Application failed to call **SccpMgmtInit** prior to this call.

SccpGetUSapStatus

Retrieves the user service access point status for a specified user service access point ID.

Prototype

SCCP_STATUS **SccpGetUSapStatus** (U8 ***board***, SccpUSapStatus ****pStatus***, U16 ***sapId***)

Argument	Description
board	TX board number to which this request is directed.

pStatus	Pointer to the address of the caller's SCCP user service access point status buffer where the status is returned:
	`typedef struct`
	`{`
	`  U8  month;  /* month */`
	`  U8  day;  /* day */`
	`  U8  year;  /* year - since 1900 */`
	`  U8  hour;  /* hour - 24 hour clock */`
	`  U8  min;  /* minute */`
	`  U8  sec;  /* second */`
	`  U8  tenths;  /* tenths of second */`
	`  U8  fill;  /* alignment */`
	`} DateTime;`
	
	`typedef struct`
	`{`
	`  U32  opc;  /* OPC */`
	`  U16  opcStatus; /* status for this OPC */`
	`} OpcStatusInfo;`
	
	`typedef struct`
	`{`
	`  DateTime  dt;  /* Date and Time */`
	`  S16  sapStatus;  /* status */`
	`  U16  numAltOpc;  /* number of Alternate PCs */`
	`  OpcStatusInfo opcInfo[SCCP_MAXALTOPC];`
	`  U8  inCongLevel; /* inbound queue congestion level */`
	`  U16  spare1;  /* spare for alignment */`
	`} SccpUSapStatus;`
	Refer to the Details section for valid status values.

sapId	User service access point ID being retrieved.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BOARD	***board*** is out of range.
SCCP_DRIVER	Error occurred accessing the driver.
SCCP_EXCEEDMAXCFG	User service access point ID is out of the configured range.
SCCP_NOGENCFG	**SccpSetGenCfg** has not been called.
SCCP_NULLPTR	Null pointer was specified for ***pStatus***.
SCCP_RESOURCES	TX board resource is exhausted. Check the *txalarm* log.
SCCP_TIMEOUT	Request timed out.
SCCP_UNINIT	Application failed to call **SccpMgmtInit** prior to this call.

Details

The following status values are bit-set:

0x01	SCCP_STA_BND	Bind
0x02	SCCP_STA_FLCOFF	Flow control off
0x04	SCCP_STA_GUARD	Guarding
0x08	SCCP_STA_BRT	Backup routed
0x10	SCCP_STA_GRNT	Waiting for grant
0x20	SCCP_STA_IGNR	Ignore
0x40	SCCP_STA_PROH	Prohibited
0x80	SCCP_STA_REST	Restarting

SccpInitAddrCfg

Initializes an SCCP global title translation definition to its default values that can be passed to **SccpSetAddrCfg**.

Prototype

SCCP_STATUS **SccpInitAddrCfg** (SccpAddrMapCfg ***pCfg**, U8 ***pGtIn**, U8 ***nBytes**)

Argument	Description
pCfg	Pointer to an address of the SCCP global title translation buffer.
pGtIn	Pointer to a buffer containing binary coded decimal digits that define the global title (or partial title) to be translated.
nBytes	Number of bytes in the BCD encoded global title.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BUFLEN	***nBytes*** is out of range.
SCCP_NULLPTR	Null pointer was specified for ***pCfg***.

Details

Prior to calling **SccpSetAddrCfg** to send the configuration block to the SCCP layer, the application can change the default values within the specified range for any fields other than those denoted as internal or unused.

```
typedef struct
{
  U8  length;              /* length in bytes            */
  U8  spare1;              /* alignment                  */
  U8  spare2;              /* alignment                  */
  U8  spare3;              /* alignment                  */
  U8  strg[SCCP_LENADDR];  /* address value - bcd digits */
} SccpAddrName;

typedef struct
{
  U8  format;                  /* format type               */
  U8  spare1;                  /* alignment                 */
  U16 spare2;                  /* alignment                 */
  union {
     struct {                  /* Format 1:                 */
       U8  oddEven;            /* odd/even indicator        */
       U8  natAddr;            /* nature of address         */
       U8  spare1;             /* alignment                 */
       U8  spare2;             /* alignment                 */
      } f1;
     struct {                  /* Format 2:                 */
       U8  tType;              /* translation type          */
       U8  spare1;             /* alignment                 */
       U8  spare2;             /* alignment                 */
       U8  spare3;             /* alignment                 */
      } f2;
     struct {                  /* Format 3:                 */
       U8  tType;              /* translation type          */
       U8  numPlan;            /* numbering plan            */
       U8  encSch;             /* encoding scheme           */
       U8  spare1;             /* alignment                 */
```

```
      } f3;
     struct {                   /* Format 4:                   */
       U8 tType;                /* translation type            */
       U8 numPlan;              /* numbering plan              */
       U8 encSch;               /* encoding scheme             */
       U8 natAddr;              /* nature of address           */
      } f4;
  }
 SccpAddrName addr;             /* address digits (BCD hex)    */
} SccpGlbTitleCfg;

typedef struct
{
  U8              pres;   /* address is present (always 1)    */
  U8              spare1; /* alignment                        */
  S16             swType; /* variant switch                   */
  U8              niInd;  /* national/international indicator */
  U8              rtgInd; /* routing indicator                */
  U8              ssnInd; /* subsystem number indicator       */
  U8              pcInd;  /* point code indicator             */
  U8              ssn;    /* subsystem number                 */
  U8              spare2; /* alignment                        */
  U32             pc;     /* point code                       */
  SccpGlbTitleCfg gt;     /* global title                     */
} SccpAddrCfg;

typedef struct
{
  SccpAddrName  gtI;    /* Global Title (incoming)               */
  U8            replGt; /* replace Gt (0=leave glt,1=replace glt)*/
  U8            spare1; /* alignment                             */
  U8            spare2; /* alignment                             */
  SccpAddrCfg   addr;   /* outgoing address                      */
} SccpAddrMapCfg;
```

Default values for the SccpAddrCfg structure that can be overridden by the calling application are listed in the following table.

Note: SccpAddrMapCfg structure members not listed in the following table are either unused or for internal use only. These fields are set to correct values by **SccpInitAddrCfg** and must not be overridden by the application.

Once a global title translation is set by **SccpSetAddrCfg**, it can only be modified by first deleting it with **SccpDelAddrCfg**, and then calling **SccpSetAddrCfg** with the new parameters.

Field		Default value	Description
gtI	N/A	Parameter	Global title or portion of a global title to be matched in outgoing addresses.
replGt	0 or 1	0	Set to 1 if the global title should be replaced on the outgoing message.
addr.swType	SCCP_SW_ANS SCCP_SW_ITU	SCCP_SW_ANS	Format of the address.
addr.rtgInd	ROUTE_PC_SN ROUTE_GLT	ROUTE_PC_SN	Set the routing flag on the outgoing message. Either route by point code and SSN, or route by the global title.
addr.pcInd	0 or 1	0	Set to 1 if a point code is included in an outgoing message.
addr.pc	N/A	None	Translated point code. Entered as a hex number.
addr.ssnInd	0 or 1	0	Set to 1 if a subsystem is included in the outgoing message. If a subsystem is included in the called address of the message, that subsystem is inserted into the message.
addr.ssn	0 - 255	None	Translated subsystem number.
addr.niInd	ADDRIND_INT ADDRIND_NAT	ADDRIND_NAT	National or international indicator in the outgoing message.
addr.gt.format	0 - 4	0	Only used if the global title is to be replaced (replGt is 1). Specifies the format of the global title.
addr.gt.gt.fn.tType	0 - 255	None	Only used if the global title is to be replaced (replGt is 1), and if required by the global title format.
addr.gt.gt.fn.natAddr	0 - 4	None	Only used if the global title is to be replaced (replGt is 1), and if required by the global title format.
addr.gt.gt.fn.numPlan	0 - 15	None	Only used if the global title is to be replaced (replGt is 1), and if required by the global title format.
addr.gt.addr	N/A	None	Only used if the global title is to be replaced (replGt is 1).

You can specify the content of outgoing messages for global title translations using either of the following methods:

- Global title translation completed by this node

  The following example shows the translation settings, where the first three digits of the global title match 847 and ANSI addressing is used:

```
gtI             847
replGt          0
addr.pres       1
addr.swType     SCCP_SW_ANS
addr.rtgInd     ROUTE_PC_SN
addr.pcInd      1
addr.pc         1.1.2
addr.ssnInd     1
addr.ssn        254
addr.niInd      ADDRIND_NAT
```

  The outgoing message is routed to point code 1.1.2, subsystem 254, with the routing flag set to route by point code and SSN. The original global title is included in the outgoing message. Since the replGt field is zero (the global title is not replaced), none of the addr.gt fields need to be filled.

- Global title translation completed by another node

  The following example shows the translation settings, where the first three digits of the global title match 847 and ANSI addressing is used, and the translation node is at point code 1.1.2, subsystem 254:

```
gtI             847
replGt          0
addr.pres       1
addr.swType     SCCP_SW_ANS
addr.rtgInd     ROUTE_GLT
addr.pcInd      1
addr.pc         1.1.2
addr.ssnInd     1
addr.ssn        254
addr.niInd      ADDRIND_NAT
```

  The outgoing message is routed to point code 1.1.2, subsystem 254, with the routing flag set to route by global title. The original global title is included in the outgoing message. The node at 1.1.2 translates the global title. Since the replGt field is zero (the global title is not replaced), none of the addr.gt fields need to be filled.

Refer to *Global title translation* on page 26 for more information.

SccpInitGenCfg

Initializes an SCCP general configuration buffer to default values that can be passed to **SccpSetGenCfg**.

Prototype

SCCP_STATUS **SccpInitGenCfg** (**SccpGenCfg** ***pCfg***)

Argument	Description
pCfg	Pointer to the SCCP general configuration structure to initialize. Refer to the Details section for information.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_NULLPTR	Null pointer was specified for ***pCfg***.

Details

Prior to calling **SccpSetGenCfg** to send the configuration to the SCCP layer, the application can change the default values within the specified range for any fields other than those denoted as internal or unused.

The SCCPGenCfg structure contains the following fields:

```
typedef struct
{
  U8       enable;         /* 0 = disabled, 1 = enabled              */
  U8       spare1;         /* alignment                              */
  U16      value;          /* timeout in tenths of seconds           */
}

typedef struct
{
  U8       maxSaps;      /* Max Number of SCCP User Saps               */
  U8       maxNSaps;     /* Max Number of Network Saps(MTP3)           */
  U16      maxScli;      /* Max Number of Sequence
                          * Connectionless Instances                  */
  U16      maxAddrs;     /* Max Number of Addresses                    */
  U16      maxRtes;      /* Max Number of Routes                       */
  U8       DefRouting;   /* Flag for Default Routing                   */
  U8       ConnDrop;     /* Flag prevents Connection Drop on
                          * link loss                                  */
  U16      maxAdjDpc;    /* Max Number of Adjacent Point Codes         */
  U16      maxMsgDrn;    /* Max message to drain  (max: 65535)         */
  U16      maxXUDTs;     /* Max external. unit data control
                          * blocks (ITU92)                            */
  U16      maxXUDTref;   /* Obsolete, not used                         */
  U8       status;       /* Local Status -- SCCP_ONLINE | SCCP_OFFLINE */
  U8       pcDispFmt     /* point code display format                  */
  S16      sogThresh;    /* Threshold for granting state change        */
  S16      scliTimeRes;  /* Time Resolution                            */
  S16      rteTimeRes;   /* Time Resolution                            */
  S16      sapTimeRes;   /* Time Resolution                            */
  S16      nsapTimeRes;  /* Time resolution                            */
  S16      xrefTimeRes;  /* Time Resolution                            */
  S16      asmbTimeRes;  /* Time Resolution                            */
  U16      spare2;       /* alignment                                  */
  TimerCfg scliTimer;    /* default SCLI timer                         */
  TimerCfg sstTimer;     /* default SST timer                          */
  TimerCfg nsapTimer;    /* default network sap timer                  */
  TimerCfg srtTimer;     /* Default Timer for SRT                      */
```

```
  TimerCfg ignoreTimer;  /* default ignore SST timer                    */
  TimerCfg coordTimer;   /* default Cord State Change SST timer         */
  TimerCfg xrefFrzTimer; /* default timer for freezing external
                          * unit data reference (ITU-92)                */
  TimerCfg defAsmbTmr;   /* default timer for external unit
                          * data assembly (CCITT92)                     */
  PDesc    stkmgr;       /* Stack Manager - not used                    */
  U16      maxConn;      /* max number of connections                   */
  S16      connThresh;   /* Connection Threshold                        */
  S16      queueThresh;  /* Queue Threshold                             */
  S16      conTimeRes;   /* connection Timer resolution                 */
  S16      frzTimeRes;   /* freeze Timer resolution                     */
  U16      spare3;       /* alignment                                   */
  TimerCfg freezeTimer;  /* default lcl reference freeze timer          */
  TimerCfg connTimer;    /* default connection timer                    */
  TimerCfg txInactTimer; /* time to allow connection to exist
                          * with no transmit packets                    */
  TimerCfg rxInactTimer; /* time to allow connection to exist
                          * with no receive packets                     */
  TimerCfg relTimer;     /* default release timer                       */
  TimerCfg repRelTimer;  /* default repeat release timer (ITU-92 only)  */
  TimerCfg intvalTimer;  /* default report abnormal release timer       */
  TimerCfg guardTimer;   /* default restart timer                       */
  TimerCfg resetTimer;   /* default reset timer                         */
  TimerCfg aicTimer;     /* default app. inactivity control timer       */
  TimerCfg aicRespTimer; /* default app. inactivity response timer      */
  TimerCfg conRespTimer; /* default connect. indication response timer  */
  U8       alarmLevel;   /* alarm level                                 */
  U8       traceFlags;   /* trace flags                                 */
  U8       spare4;       /* alignment                                   */
  U8       spare5;       /* alignment                                   */
  U32      memThresh1;   /* memory congestion threshold 1               */
  U32      memThresh2;   /* memory congestion threshold 2               */
  U32      memThresh3;   /* memory congestion threshold 3               */
} SccpGenCfg;
```

Default values for the SccpGenCfg structure that can be overridden by the calling application are listed in the following table.

Note: SccpGenCfg structure members not listed in the following table are either unused or for internal use only. These fields are set to correct values by **SccpInitGenCfg** and must not be overridden by the application.

All values listed in the following table can be modified on the first call to **SccpSetGenCfg**. On subsequent calls to **SccpSetGenCfg**, only parameters listed in **bold** can be modified. Those fields not in **bold** are ignored on subsequent calls.

Field	Range	Default value	Description
maxSaps	1 - 255	2	Maximum number of SCCP user SAPs (subsystem number and protocol variant) that can be defined.
maxNSaps	1 - 255	1	Maximum number of MTP 3 network SAPs that SCCP uses.
maxScli	0 - 65535	20	Maximum number of simultaneous sequenced connectionless data transfers.
maxAddrs	0 - 65535	7	Maximum number of global title translation entries.
maxRtes	1 - 65535	4	Maximum number of route definition entries.

Field	Range	Default value	Description
DefRouting	0 or 1	0	Set to 1 to enable the default routing feature.
ConnDrop	0 or 1	0	Set to 1 to retain connections over a link loss.
maxAdjDpc	0 - 65535	4	Maximum number of route definitions that can be defined as adjacent (SCCP_ADJACENT). If a route is specified as adjacent, it is notified of some point code changes in this or other nodes. It is also notified of some subsystem status changes in this or other nodes.
maxMsgDrn	0 - 65535	5	Maximum number of messages queued to MTP 3 to send in one time interval when exiting flow control.
maxXUDTs	0 - 65535	0	Maximum number of control blocks to allocate for reassembling segmented extended unit data messages.
status	SCCP_OFFLINE SCCP_ONLINE	SCCP_ONLINE	Initial status of SCCP layer.
pcDispFmt	DEFAULT[DFLT] INTERNATIONAL[INTL] JNTT	DEFAULT	For DEFAULT, point codes are interpreted as 24 bit 8.8.8 values. For INTERNATIONAL, point codes are interpreted as 14 bit 3.8.3 values. For JNTT, point codes are interpreted as 16 bit 7.4.5 values.
sogThresh	0 - 9	1	Minimum percentage of board memory that must be available before granting a subsystem out-of-service request (SOR) from a backup signaling point.
scliTimer	0 - 65535	2 seconds	Maximum time that a sequenced connectionless transmission can take before control block is deallocated.
sstTimer	0 - 65535	30 seconds	Time to wait between subsystem test (SST) messages.
nsapTimer	0 - 65535	1 seconds	Time to wait between draining blocks of queued messages to the MTP 3 layer after exiting flow control.

Field	Range	Default value	Description
srtTimer	0 - 65535	30 seconds	Time to wait between subsystem routing test (SRT) messages.
ignoreTimer	0 - 65535	30 seconds	Time period after local subsystem goes out of service to ignore subsystem test messages.
coordTimer	0 - 65535	30 seconds	Time to wait for response to coordinated state change request.
xrefFrzTimer	0 - 65535	1 second	Time to freeze an XUDT local reference before reusing it.
defAsmbTmr	0 - 65535	20 seconds	Maximum time for reassembling all segments of an XUDT message.
maxConn	0 - 65535	512	Maximum number of simultaneous connections.
connThresh	0 - 9	3	Maximum percentage of board memory that must be available before accepting a new connection in either direction (expressed in units of 10%).
queueThresh	0 - 32766	3	Maximum number of data messages that can be queued for a connection waiting for a connection window to open.
freezeTimer	0 - 65535	1 second	Time to freeze a connection local reference before reusing it.
connTimer	0 - 65535	180 seconds	Time to wait for a response to a connection request.
txInactTimer	0 - 65535	600 seconds	Time to wait with no outgoing packets on a connection before sending an inactivity test (IT) message.
rxInactTimer	0 - 65535	900 seconds	Time to wait with no incoming packets on a connection before clearing the connection. The amount of time should be greater than txInactTimer.
relTimer	0 - 65535	4 seconds	Time to wait for a response to a release request.
repRelTimer	0 - 65535	4 seconds	Time to wait for a response to a second release request.
intvalTimer	0 - 65535	8 seconds	Time to wait before reporting abnormal release timer.

Field	Range	Default value	Description
guardTimer	0 - 65535	1 second	Time to wait after a MTP 3 restart before allowing traffic.
resetTimer	0 - 65535	6 seconds	Time to wait for a response to a reset request.
aicTimer	0 - 65535	480 seconds	Time with no application activity for an active connection before NMS SCCP generates a connection inactivity indication event. Used only if application inactivity control is enabled for that user SAP.
aicRespTimer	0 - 65535	10 seconds	Time that the application has to respond to a connection inactivity indication event. Used only if application inactivity control is enabled for that user SAP.
conRespTimer	0 - 65535	10 seconds	Time that the application has to respond to an incoming connection indication with either a connection response or a release request before the SCCP layer refuses the connection. If the value is zero, no timing for the application response is performed.
alarmLevel	SCCP_ALARMLVL_DISABLE SCCP_ALARMLVL_DEFAULT SCCP_ALARMLVL_DEBUG SCCP_ALARMLVL_DETAIL	SCCP_ALARMLVL_DEFAULT	Level of alarms generated by the SCCP layer. Refer to **SccpAlarmControl** for a definition of alarm levels.
traceFlags	0 - 255	0	Data tracing. Refer to **SccpTraceControl** for a definition of trace flags.
memThresh1	0 - 99	10	Percentage of board memory available at which memory congestion level 1 starts.
memThresh2	0 - 99	8	Percentage of board memory available at which memory congestion level 2 starts.
memThresh3	0 - 99	5	Percentage of board memory available at which memory congestion level 3 starts.

SccpInitNSapCfg

Initializes an SCCP network service access point (NSAP) configuration buffer to default values that can be passed to **SccpSetNSapCfg**.

Prototype

SCCP_STATUS **SccpInitNSapCfg** (SccpNSapCfg ***pCfg***, U16 ***sapId***, U32 ***dpc***, S16 ***swType***)

Argument	Description

pCfg	Pointer to the address of the SCCP NSAP configuration parameters buffer:
	`typedef struct`
	`{`
	`  U8  length;  /* length in bytes */`
	`  U8  strg[SCCP_LENADDR];  /* address value - hex bcd digits */`
	`  U8  spare1;  /* alignment */`
	`} SccpAddrMask;`
	
	`typedef struct`
	`{`
	`  S16  swType;  /* Protocol Switch type/version */`
	`  U8  dpcLen;  /* dpc length(PCLEN_ANSI |`
	`  * PCLEN_ITU) */`
	`  U8  useMsk;  /* use address mask? (0=no, 1=yes) */`
	`  U32  dpc;  /* point code for this nsap`
	`  * (switch) */`
	`  S16  maxMsgLen;  /* max message length for`
	`  * MTP3 on this SAP */`
	`  S16  txQThr;  /* max msgs queued to MTP3`
	`  * before discarding */`
	`  SccpAddrMask addrMaskList[SCCP_MAXMASKS];`
	`  /* address mask for incoming pkts`
	`  * on this SAP */`
	`  MemoryId  mem;  /* memory region & pool id */`
	`  U16  dstProcId;  /* destination processor id */`
	`  S16  spId;  /* service provider id */`
	`  U8  selector;  /* selector */`
	`  U8  priority;  /* priority */`
	`  U8  route;  /* route */`
	`  U8  dstEnt;  /* destination entity */`
	`  U8  dstInst;  /* detination instance */`
	`  U8  hopCnt;  /* default Hop Count`
	`  * (between 1 and 15) */`
	`  U8  ssf;  /* subsytem part of SIO to`
	`  * be used on NSAP */`
	`U8  niInd;  /* network indicator (0..1) for`
	`  * SCCP mgmt msgs */`
	`  U16  spare1;  /* alignment */`
	`  U16  spare2;  /* alignment */`
	`} SccpNSapCfg;`

sapId	Network service access point ID.
dpc	Point code of this NSAP.
swType	Switch type that the SCCP NSAP uses: SCCP_SW_ITU SCCP_SW_ANS

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_NULLPTR	Null pointer was specified for ***pCfg***.
SCCP_SWTYPE	***swType*** contains an invalid switch type.

Details

Prior to calling **SccpSetNSapCfg** to send the configuration block to the SCCP layer, the application can change the default values within the specified range for any fields other than those denoted as internal or unused.

Default values for the SccpNSapCfg structure that can be overridden by the calling application are listed in the following table.

Note: SccpNSapCfg structure members not listed in the following table are either unused or for internal use only. These fields are set to correct values by **SccpInitNSapCfg** and must not be overridden by the application.

All values listed in the following table can be modified on the first call to **SccpSetNSapCfg**. On subsequent calls to **SccpSetNSapCfg**, only parameters listed in **bold** can be modified. Those fields not in **bold** are ignored on subsequent calls.

Field	Range	Default value	Description
swType	SCCP_SW_ANS SCCP_SW_ITU	parameter	Protocol variant for this NSAP.
dpcLen	PCLEN_ANSI PCLEN_ITU	ANSI: PCLEN_ANSI ITU: PCLEN_ITU	Point code length of this network.
dpc	Hex number	parameter	Point code of this node. Enter 1.1.1 as 0x00010101.
maxMsgLen	32 - 1500	256	Maximum length of a message passed to MTP 3 on this NSAP.
txQThr	0 - 32766	20	Maximum number of messages to queue to MTP 3 before discarding.
useMsk	0 or 1	0	Set to 1 if an addrMsk is defined.
AddrMaskList[i].length	0 to SCCP_LENADR	none	Length of the address mask in bytes. Set to 0 if a mask is not used.
AddrMaskList[i].strg	0 to SCCP_LENADR characters	none	Address mask describing which digits to match when performing a global title translation.
spId	0 to (SccpGenCfg maxNSaps-1)	parameter	NSAP ID.
hopCnt	1 - 15	10	Hop count value used on outgoing SCCP messages from this service access point.
ssf	SCCP_SSF_INTL SCCP_SSF_SPARE SCCP_SSF_NAT SCCP_SSF_RES	ANSI: SCCP_SSF_NAT ITU: SCCP_SSF_INTL	Value used in the subservice field for this network.
niInd	0 - 1	ANSI: 1 ITU: 0	Network indicator for SCCP management messages.

The addrMsk can be composed of only 0 and f. Each byte contains two hexadecimal digits. Use **SccpAsciiMaskToBcd** to convert an ASCII string into the correct format.

An addrMsk of a single 0 (zero) matches all global titles. Any global title masked with an addrMsk of zero matches an ADDRESS section of 0 (zero).

SccpInitRteCfg

Initializes an SCCP route definition that can be passed to **SccpSetRteCfg**.

Prototype

SCCP_STATUS **SccpInitRteCfg** (SccpRouteCfg ***pCfg**, U32 ***dpc**)

Argument	Description

pCfg	Pointer to the address of the SCCP route definition buffer:
	`typedef struct`
	`{`
	`  S16  swType;  /* not used */`
	`  U8  ssn;  /* subsystem number */`
	`  U8  status;  /* status */`
	`  U8  bkupPcInd;  /* backup point code indicator * (0=none) */`
	`  U8  upOnResume;  /* alignment */`
	`  U16  numConPc;  /* number of concerned point codes */`
	`  U32  bkupPc;  /* backup point code */`
	`  U32  conPcList[SCCP_MAXCONPC];  /* concerned point codes */`
	`} SccpSsnCfg;`
	
	`typedef struct`
	`{`
	`  U32  rteOpc;  /* OPC for this route */`
	`  U8  status;  /* status of the route for this OPC */`
	`  U8  align1;  /* */`
	`  U16  align2;  /* */`
	`} RouteOpcInfo;`
	
	`typedef struct`
	`{`
	`  S16  swType;  /* Protocol Switch * type/version */`
	`  U8  status;  /* adjacent/translator * flags */`
	`  U8  bkupPcInd;  /* backup point code * indicator (0=none) */`
	`  U32  dpc;  /* destination point code */`
	`  U32  bkupPc;  /* backup point code */`
	`  U8  numSsns;  /* number of subsystems */`
	`  U8  spare1;  /* alignment */`
	`  U8  spare2;  /* alignment */`
	`  U16  nmbAltOpc;  /* number of alt. point codes */`
	`  RouteOpcInfo rteAltOpc[SCCP_MAXALTOPC]; /* Alternate OPCs */`
	`  SccpSsnCfg ssnList[SCCP_MAXSSNS]; /*subsystems for this dpc */`
	`} SccpRouteCfg;`

dpc	Destination point code of the route to be added, specified as a hexadecimal number. Point code 1.1.1 is stored as 0x00010101.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_NULLPTR	Null pointer was specified for ***pCfg***.
SCCP_POINTCODE	***dpc*** is out of range.

Details

Prior to calling **SccpSetRteCfg** to send the configuration block to the SCCP layer, the application can change the default values within the specified range for any fields other than those denoted as internal or unused.

Default values for the SccpRouteCfg structure that can be overridden by the calling application are listed in the following table.

Note: SccpRouteCfg structure members not listed in the following table are either unused or for internal use only. These fields are set to correct values by **SccpInitRteCfg** and must not be overridden by the application.

Once a route definition is set by **SccpSetRteCfg**, it can only be modified by first deleting it with **SccpDelRteCfg**, and then calling **SccpSetRteCfg** with the new parameters.

Field	Range	Default value	Description
swType	SCCP_SW_ANS SCCP_SW_ITU	SCCP_SW_ANS	Protocol variant for this point code.
dpc	Hex number	None	Input parameter. Enter 1.1.1 as 0x00010101.
status	SCCP_ONLINE SCCP_TRANS SCCP_ADJACENT	SCCP_ONLINE SCCP_TRANS SCCP_ADJACENT	Flags to set initial status of point code. Flags are OR'd together: SCCP_ONLINE SCCP_TRANS SCCP_ADJACENT
bkupPcInd	0 or 1	0	Set to 1 if backup point code is specified.
bkupPc	Hex number	None	Point code of backup for this destination. Enter 1.1.1 as 0x00010101.
numSsns	0 to SCCP_MAXSSNS	0	Number of subsystem definitions contained in the ssnList array.
ssnList	N/A	none	Array of subsystem definitions for this point code.
ssnList[n].ssn	0 - 255	none	Subsystem number.
ssnList[n].status	SCCP_SNR SCCP_ACC	SCCP_SNR SCCP_ACC	Flags to set initial status of subsystem. Flags are OR'd together: SCCP_SNR SCCP_ACC
ssnList[n].bkupPcInd	0 or 1	0	Set to 1 if backup point code is specified.
ssnList[n].bkupPc	Hex number	none	Point code of backup for this subsystem. Enter 1.1.1 as 0x00010101.
ssnList[n].numConPc	0 to SCCP_MAXCONPC	0	Number of concerned point codes contained in the conPcList array.
ssnList[n].conPcList	see dpc	none	List of concerned point codes for this subsystem.
ssnList[n].upOnResume	0 - 1	1	Subsystem is immediately put back in service when a point code resume message is received from MTP. The subsystem test procedure is not started.

Field	Range	Default value	Description
numAltOpc	0-8	0	Number of alternate originating point codes for this route.
rteAltOpc	N/A	none	Array of alternate originating point codes.
rteAltOpc.rteOpc	Hex number	none	Originating point code for this route.
rteAltOpc.status	none	none	Status of this route for this originating point code.

SccpInitUSapCfg

Initializes an SCCP user service access point configuration buffer to default values that can be passed to **SccpSetUSapCfg**.

Prototype

SCCP_STATUS **SccpInitUSapCfg** (SccpUSapCfg ***pCfg**, S16 ***swType***)

Argument	Description

pCfg	Pointer to the address of the SCCP SAP configuration parameters buffer:
	`typedef struct`
	`{`
	`  U16  numConPc;  /* number of concerned point codes */`
	`  U16  dummy;  /*  */`
	`  U32  usapOpc;  /* OPC for this USAP  */`
	`  U32  conPcList[SCCP_MAXCONPC];  /* concerned point code list  */`
	`} UsapOpcInfo;`
	
	`typedef struct`
	`{`
	`  S16  swType;  /* Protocol Switch * type/version  */`
	`  U8  selector;  /* selector  */`
	`  U8  spare1;  /* alignment  */`
	`  MemoryId  mem;  /* memory region & pool id  */`
	`  U8  bkupPcInd;  /* backup point code indicator * (0==none)  */`
	`  U8  aicEnabled;  /* app. inactivity timing * enabled (1 = yes)  */`
	`  U16  numAltOpc;  /* number of alternate PCs  */`
	`  UsapOpcInfo  usapAltOpc[SCCP_MAXALTOPC];  /* multiple OPCs  */`
	`  U32  bkupPc;  /* backup point code  */`
	`  SccpAddrMask  addrMaskList[SCCP_MAXMASKS];`
	`  /* address mask for outgoing * pkts on this SAP  */`
	`  U8  useMsk;  /* use address mask? * (0=no, 1=yes)  */`
	`  U8  priority;  /* priority  */`
	`  U8  route;  /* route  */`
	`  U8  hopCnt;  /* default Hop Count * (between 1 and 15)  */`
	`  U32  qThresh1;  /* inbound queue congestion * threshold 1  */`
	`  U32  qThresh2;  /* inbound queue congestion * threshold 2  */`
	`  U32  qThresh3;  /* inbound queue congestion * threshold 3  */`
	`} SccpUSapCfg;`

swType	Switch type used by the SCCP service access point: SCCP_SW_ITU88 SCCP_SW_ITU92 SCCP_SW_ITU96 SCCP_SW_ANS88 SCCP_SW_ANS92 SCCP_SW_ANS96

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_NULLPTR	Null pointer was specified for ***pCfg***.
SCCP_SWTYPE	Invalid switch type was specified for ***swType***.

Details

Prior to calling **SccpSetUSapCfg** to send the configuration block to the SCCP layer, the application can change the default values within the specified range for any fields other than those denoted as internal or unused.

Default values for the SccpUSapCfg structure that can be overridden by the calling application are listed in the following table.

Note: SccpUSapCfg structure members not listed in the following table are either unused or for internal use only. These fields are set to correct values by **SccpInitUSapCfg** and must not be overridden by the application.

All values listed in the following table can be modified on the first call to **SccpSetUSapCfg**. On subsequent calls to **SccpSetUSapCfg**, only parameters listed in **bold** can be modified. Those fields not in **bold** are ignored on subsequent calls.

Field	Range	Default value	Description
swType	SCCP_SW_ANS88 SCCP_SW_ANS92 SCCP_SW_ANS96 SCCP_SW_ITU88 SCCP_SW_ITU92 SCCP_SW_ITU96	SCCP_SW_ANS92	Protocol variant used for this SAP.
bkupPcInd	0 or 1	0	Set to 1 if a backup point code is to be configured.
bkupPc	Hex number	none	Point code where this subsystem is backed up. Enter 1.1.1 as 0x00010101.
aicEnabled	0 or 1	0	Set to 1 to enable application inactivity timer.
numAltOpc	0 - 8	none	Number of alternate point codes.
usapAltOpc	N/A	none	Array of alternate point codes.
usapAltOpc[i].numConPc	0 to SCCP_MAXALTPC	0	Number of concerned point codes in the conPcList.
usapAltOpc[i].usapOpc	Hex number	none	Originating point code for this USAP.
usapAltOpc[i].conPcList	Hex number	none	Up to SCCP_MAXCONPC point codes, each entered as a hex value. Enter 1.1.1 as 0x00010101.
AddrMaskList[i].length	0 to SCCP_LENADR	none	Length of the address mask in bytes. Set to 0 if a mask is not used.
AddrMaskList[i].strg	0 to SCCP_LENADR characters	none	An address mask describing which digits to match when performing a global title translation.
useMsk	0 or 1	0	Set to 1 if an addrMsk is defined.
hopCnt	1 - 15	10	Hop count value used on outgoing SCCP messages from this SAP.
qThresh1	1 - 2000	600	Number of messages outstanding to a higher level task (TCAP) or an application at which inbound congestion level 1 starts.
qThresh2	1 - 2000	900	Number of messages outstanding to a higher level task (TCAP) or an application at which inbound congestion level 2 starts.
qThresh3	1 - 2000	1200	Number of messages outstanding to a higher level task (TCAP) or an application at which inbound congestion level 3 starts.

The addrMsk can be composed of only 0 and f. Each byte contains two hexadecimal digits. **SccpAsciiMaskToBcd** can be used to convert an ASCII string into the correct format. As an example, FFF masks off everything but the first three digits of a global

title. The result is then compared to the various global titles configured. The mask FFFFFFFFFF does not mask off any digits.

An addrMsk of a single 0 (zero) matches all global titles. Any global title masked with an addrMsk of 0 (zero) matches an ADDRESS section of 0 (zero).

SccpMgmtInit

Initializes SCCP management and provides access to the TX device driver.

Prototype

SCCP_STATUS **SccpMgmtInit** (U8 ***board***, U8 ***srcEnt***, U8 ***srcInst***)

Argument	Description
board	TX board number.
srcEnt	Calling application entity ID.
srcInst	Calling application instance ID.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BOARD	***board*** is out of range.
SCCP_DRIVER	Error occurred accessing the driver.
SCCP_INSTANCE	Instance ID is out of range.
SCCP_TOOMANY	Too many applications have initialized the board concurrently.

Details

This function must be called before any other SCCP management functions are called.

Note: If the same application uses both the SCCP service functions and the SCCP management functions, separate entity IDs must be specified when the two APIs are initialized.

SccpMgmtTerm

Terminates the connection between the application with the TX device driver, releasing any resources (such as file descriptors) associated with SCCP management.

Prototype

SCCP_STATUS **SccpMgmtTerm** (U8 ***board***)

Argument	Description
board	TX board number.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BOARD	***board*** is out of range.
SCCP_UNINIT	Application failed to call **SccpMgmtInit** prior to this call.

Details

Call this function once for each board the application initialized separately.

SccpSetAddrCfg

Sends an SCCP global title translation parameter block to the specified TX board.

Prototype

SCCP_STATUS **SccpSetAddrCfg** (U8 ***board***, SccpAddrMapCfg ****pCfg***)

Argument	Description
board	TX board number.
pCfg	Pointer to the address of the global title translation parameters buffer. The format is specified in **SccpInitAddrCfg**.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BADDIGIT	SccpAddrName contains invalid BCD digits.
SCCP_BOARD	***board*** is out of range.
SCCP_BUFLEN	Length specified for SccpAddrName is invalid.
SCCP_DRIVER	Error occurred accessing the driver.
SCCP_EXCEEDMAXCFG	Too many global title definitions.
SCCP_NOGENCFG	**SccpSetGenCfg** has not been called.
SCCP_NULLPTR	Null pointer was specified for ***pCfg***.
SCCP_PARAM	Invalid parameter was detected. Check the *txalarm* log.
SCCP_POINTCODE	Translated point code is out of range.
SCCP_RANGE	One or more of the following fields is out of range: addr.niInd, addr.rtgInd, addr.gt.format.
SCCP_RESOURCES	TX board resource is exhausted. Check the *txalarm* log.
SCCP_SWTYPE	Invalid switch type was specified.
SCCP_TIMEOUT	Request timed out.
SCCP_UNINIT	Application failed to call **SccpMgmtInit** prior to this call.

Details

This function can be called any time after the general configuration is downloaded to the TX board, but before any application attempts to send data for transaction processing.

Once a global title translation is set by **SccpSetAddrCfg**, it can only be modified by first deleting it with **SccpDelAddrCfg**, and then calling **SccpSetAddrCfg** with the new parameters.

pCfg must have been previously initialized with **SccpInitAddrCfg**.

SccpSetGenCfg

Sends the SCCP general configuration parameters to the TX board.

Prototype

SCCP_STATUS **SccpSetGenCfg** (U8 ***board***, SccpGenCfg ****pCfg***)

Argument	Description
board	TX board number.
pCfg	Pointer to the address of the general configuration parameters buffer. The format is specified in **SccpInitGenCfg**.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BOARD	***board*** is out of range.
SCCP_DRIVER	Error occurred accessing the driver.
SCCP_MAXROUTES	Value specified for ***pCfg***->maxRtes is out of range.
SCCP_NULLPTR	Null pointer was specified for ***pCfg***.
SCCP_PARAM	Invalid parameter was detected. Check the *txalarm* log.
SCCP_RESOURCES	TX board resource is exhausted. Check the *txalarm* log.
SCCP_TIMEOUT	Request timed out.
SCCP_TIMERVALUE	Enabled timer has a value of zero.
SCCP_UNINIT	Application failed to call **SccpMgmtInit** prior to this call.

Details

The first time **SccpSetGenCfg** is called after a download of the TX board, all the parameters listed in **SccpInitGenCfg** can be modified. On any subsequent call to **SccpSetGenCfg** (without downloading the board), only some parameters can be modified. These are listed in **bold** in **SccpInitGenCfg**. Those fields not in **bold** are ignored on subsequent calls.

SccpSetGenCfg must be called before any SCCP user SAPs, NSAPs, global titles, or routes are configured.

pCfg must have been previously initialized with **SccpInitGenCfg**.

SccpSetNSapCfg

Sends an SCCP network service access point (NSAP) configuration parameter block to the specified TX board to define or update the configuration for a specific SCCP NSAP.

Prototype

SCCP_STATUS **SccpSetNSapCfg** (U8 ***board***, SccpNSapCfg ****pCfg***, U16 ***sapId***)

Argument	Description
board	TX board number.
pCfg	Pointer to the address of the service access point configuration parameters buffer. The format is specified in **SccpInitNSapCfg**.
sapId	Network service access point ID.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BADDIGIT	addrMsk contains invalid digits.
SCCP_BOARD	***board*** is out of range.
SCCP_BUFLEN	addrMsk.length is out of range.
SCCP_DRIVER	Error occurred accessing the driver.
SCCP_EXCEEDMAXCFG	***sapId*** exceeds the maximum specified through **SccpSetGenCfg**.
SCCP_HOPCOUNT	hopCnt is out of range.
SCCP_NOGENCFG	**SccpSetGenCfg** has not been called.
SCCP_NULLPTR	Null pointer was specified for ***pCfg***.
SCCP_PARAM	Invalid parameter was detected. Check the *txalarm* log.
SCCP_POINTCODE	Either dpcLen or dpc is out of range.
SCCP_RANGE	One of the following is out of range: maxMsgLen, txQThr, spId, or ssf.
SCCP_RESOURCES	TX board resource is exhausted. Check the *txalarm* log.
SCCP_SWTYPE	Invalid switch type was specified.
SCCP_TIMEOUT	Request timed out.
SCCP_UNINIT	Application failed to call **SccpMgmtInit** prior to this call.

Details

This function can be called any time after the general configuration is downloaded to the TX board, but before any application attempts to send data for transaction processing.

The first time **SccpSetNSapCfg** is called after a download of the TX board, all the parameters listed in **SccpInitNSapCfg** can be modified. On any subsequent call to **SccpSetNSapCfg** (without downloading the board), only some parameters can be modified. These are listed in **bold** in **SccpInitNSapCfg**. Those fields not in **bold** are ignored on subsequent calls.

SccpSetNSapCfg always sends a value for dstInst equal to that specified for ***board***, regardless of the value specified in the buffer pointed to by ***pCfg***.

pCfg must have been previously initialized with **SccpInitNSapCfg**.

SccpSetRteCfg

Sends an SCCP route definition parameter block to the specified TX board.

Prototype

SCCP_STATUS **SccpSetRteCfg** (U8 ***board***, SccpRouteCfg ****pCfg***)

Argument	Description
board	TX board number.
pCfg	Pointer to the address of the route definition parameters buffer. The format is specified in **SccpInitRteCfg**.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BOARD	***board*** is out of range.
SCCP_BUFLEN	Either numSsns or ssnList[n].numConPc is out of range.
SCCP_DRIVER	Error occurred accessing the driver.
SCCP_EXCEEDMAXCFG	Too many route definitions.
SCCP_NOGENCFG	**SccpSetGenCfg** has not been called.
SCCP_NULLPTR	Null pointer was specified for ***pCfg***.
SCCP_PARAM	Invalid parameter was detected. Check the *txalarm* log.
SCCP_POINTCODE	Point code is out of range.
SCCP_RANGE	Either the SccpRouteCfg or SccpSsnCfg status is out of range.
SCCP_RESOURCES	TX board resource is exhausted. Check the *txalarm* log.
SCCP_SWTYPE	Invalid switch type was specified.
SCCP_TIMEOUT	Request timed out.
SCCP_UNINIT	Application failed to call **SccpMgmtInit** prior to this call.

Details

This function can be called any time after the general configuration is downloaded to the TX board, but before any application attempts to send data for transaction processing.

Once a global title translation is set by **SccpSetRteCfg**, it can only be modified by first deleting it with **SccpDelRteCfg**, and then calling **SccpSetRteCfg** with the new parameters.

pCfg must have been previously initialized with **SccpInitRteCfg**.

SccpSetUSapCfg

Sends an SCCP service access point (SAP) configuration parameter block to the specified TX board to define or update the configuration for a specific SCCP SAP.

Prototype

SCCP_STATUS **SccpSetUSapCfg** (U8 ***board***, SccpUSapCfg ****pCfg***, U16 ***sapId***)

Argument	Description
board	TX board number.
pCfg	Pointer to the address of the service access point configuration parameters buffer. The format is specified in **SccpInitUSapCfg**.
sapId	Service access point ID being defined.

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BADDIGIT	addrMsk contains invalid digits.
SCCP_BOARD	***board*** is out of range.
SCCP_BUFLEN	Either addrMsk.length or numConPc is out of range.
SCCP_DRIVER	Error occurred accessing the driver.
SCCP_EXCEEDMAXCFG	***sapId*** exceeds the maximum specified through **SccpSetGenCfg**.
SCCP_HOPCOUNT	hopCnt is out of range.
SCCP_NOGENCFG	**SccpSetGenCfg** has not been called.
SCCP_NULLPTR	Null pointer was specified for ***pCfg***.
SCCP_PARAM	Invalid parameter was detected. Check the *txalarm* log.
SCCP_POINTCODE	Backup or concerned point code exceeds 24 bits.
SCCP_RESOURCES	TX board resource is exhausted. Check the *txalarm* log.
SCCP_SWTYPE	swType contains an invalid switch type.
SCCP_TIMEOUT	Request timed out.
SCCP_UNINIT	Application failed to call **SccpMgmtInit** prior to this call.
SCCP_ZEROLEN	addrMsk.length is zero while useMsk is set to 1.

Details

This function can be called any time after the general configuration is downloaded to the TX board, but before any application attempts to bind to this SAP for transaction processing.

pCfg must have been previously initialized with **SccpInitUSapCfg**.

The first time **SccpSetUSapCfg** is called after a download of the TX board, all the parameters listed in **SccpInitUSapCfg** can be modified. On any subsequent call to **SccpSetUSapCfg** (without downloading the board), only some parameters can be modified. These are listed in **bold** in **SccpInitUSapCfg**. Those fields not in **bold** are ignored on subsequent calls.

SccpTraceControl

Sends a request to enable or disable tracing of SCCP protocol messages. Trace control is not currently implemented. No trace messages are generated, regardless of the **SccpTraceControl** setting.

Prototype

SCCP_STATUS **SccpTraceControl** (U8 ***board***, U8 ***bTraceOn***, U8 ***flags***)

Argument	Description
board	TX board number.
bTraceOn	Zero turns tracing off. Non-zero turns tracing on.
flags	Specifies which level of tracing is affected. Currently, NMS SCCP supports only protocol buffer tracing (dumps of all SCCP protocol messages sent/received): `#define SCCP_TRACE_DATA   0x01`

Return values

Return value	Description
SCCP_SUCCESS	
SCCP_BOARD	***board*** is out of range.
SCCP_DRIVER	Error occurred accessing the driver.
SCCP_NOGENCFG	**SccpSetGenCfg** has not been called.
SCCP_RANGE	***flags*** is out of range.
SCCP_TIMEOUT	Request timed out.
SCCP_UNINIT	Application failed to call **SccpMgmtInit** prior to this call.

7 Demonstration programs and utilities

Summary of the demonstration programs and utilities

NMS SCCP provides the following demonstration programs and utilities:

Program	Description
sccldemo	Demonstrates how the SCCP service sends and receives connectionless messages.
sccodemo	Demonstrates how the SCCP service sends and receives connection-oriented messages.
sccpcfg	Downloads the SCCP configuration to the TX board at boot time.
sccpmgr	Monitors and manages the status of the SCCP layer.

Connectionless messages: sccldemo

Demonstrates how the SCCP service sends and receives connectionless messages. It can be run as the client that sends the message, or as the server that receives the message.

Usage

```
sccldemo [options] called address [calling address]
```

Requirements

- A computer with a TX board installed
- Windows or UNIX
- Natural Access
- NMS SS7

Procedure

Follow this procedure to run *sccldemo*:

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>From the command line prompt, navigate to the \nms\tx\samples\sccp\ directory under Windows or /usr/bin directory under UNIX.</td></tr>
<tr><td>2</td><td>Connect two TX boards back-to-back. In this example, both boards are installed on the same PC, with board 1 configured with point code 1.1.1 (ANSI), and board 2 configured with point code 1.1.2 (ANSI).</td></tr>
<tr><td>3</td><td>Enter the following command:
<code>sccldemo [options] called address [calling address]</code>
where options include:
<table>
<tr><th>Options</th><th>Description</th></tr>
<tr><td>-b board</td><td>Board number (default = 1, maximum = 8).</td></tr>
<tr><td>-p sapId</td><td>SAP ID to be used (default = 0, maximum = 255).</td></tr>
<tr><td>-n subsystem</td><td>Subsystem number (default = 254, maximum = 255).</td></tr>
<tr><td>-i messages</td><td>Total number of messages to be sent (default = 1, maximum = 1,000,000).</td></tr>
<tr><td>-j delay</td><td>Delay between messages in milliseconds (default = 5000, maximum = 1,000,000).</td></tr>
<tr><td>-s</td><td>Application acts as the server.</td></tr>
</table>
The called address and calling address are specified as:
<table>
<tr><th>Option</th><th>Specifies...</th></tr>
<tr><td>ssn</td><td>Only a subsystem number. Default routing must be enabled.</td></tr>
<tr><td>x.y.z:ssn</td><td>Point code and subsystem, with the point code in dotted format.</td></tr>
<tr><td>n:ssn</td><td>Point code and subsystem, with the point code in hexadecimal format.</td></tr>
<tr><td>GT n/digits</td><td>Global title, where n is the global title format (from 1 - 4) and digits is the global title (each digit is in the range 0 - F).</td></tr>
<tr><td>GT n/digits:ssn</td><td>Global title, where n is the global title format (from 1 - 4) and digits is the global title (each digit is in the range 0 - F). Also specifies a specific subsystem number, ssn.</td></tr>
<tr><td>NONE</td><td>No address.</td></tr>
</table></td></tr>
<tr><td>4</td><td>Start the server side application first with the following syntax:
<code>sccldemo -b 1 -p 0 -n 254 -s</code>
This application uses TX board 1 (-b 1), SAP ID 0 (-p 0), and listens on subsystem 254 (-n 254). The -s option specifies that it is acting as a server and not sending any messages.</td></tr>
</table>

Step	Action

5	Start the client side application with the following syntax:
	`sccldemo -b 2 -p 0 -n 254 1.1.1:254`
	This application uses TX board 2 (-b 2), SAP ID 0 (-p 0), and listens on subsystem 254 (-n 254). The called address specifies that a message is sent to point code 1.1.1, subsystem 254. Since no calling address is specified in the command line, the default address is used, and the SCCP message contains a calling address with the point code and subsystem (1.1.2, subsystem 254).
	The client side application shows the following output:
	`Sending Connectionless Data`
	The server side application shows the following output:
	`Unitdata Indication Received, len = 100`
	If the server side application does not receive the message, refer to the **Details** section for suggestions.

Details

Each SCCP connectionless message contains a destination address (the called address) and a source address (the calling address). The client application must specify a destination for the message, which is the called address. The most common address consists of a point code and a subsystem.

If no calling address is specified, the message is sent with a default calling address of the client application point code and subsystem.

sccldemo uses SCCP management to determine the address type (ANSI or ITU) of the SAP ID in use. It adjusts the called and calling addresses appropriately.

Troubleshoot as follows:

- Use the *txalarm* utility to confirm that the two TX boards have active links between them.
- Check the SCCP configuration file to confirm that the correct destination point code and subsystem are configured.
- If using a global title for a called address, try a simple point code/subsystem combination first, and then try a global title.

Connection-oriented messages: sccodemo

Demonstrates how the SCCP service sends and receives connection-oriented messages. It can be run as the client that initiates an SCCP connection and transmits data over the connection, or as the server that receives the connection and any data transmitted.

Usage

```
sccodemo [options] called address [calling address]
```

Requirements

- A computer with a TX board installed
- Windows or UNIX
- Natural Access
- NMS SS7

Procedure

Use this procedure to run *sccodemo*:

Step	Action
1	From the command line prompt, navigate to the *\nms\tx\samples\sccp* directory under Windows or */usr/bin* directory under UNIX.
2	Connect two TX boards back-to-back. In this example, both boards are installed on the same PC, with board 1 configured with point code 1.1.1 (ANSI), and board 2 configured with point code 1.1.2 (ANSI).
3	Enter the following command: `sccodemo [options] called address [calling address]` where ***options*** include: **Options** / **Description** -b ***board*** — Board number (default = 1, maximum = 8). -p ***sapId*** — SAP ID to be used (default = 0, maximum = 255). -n ***subsystem*** — Subsystem number (default = 254, maximum = 255). -s — Application acts as the server. The ***called address*** and ***calling address*** are specified as: **Option** / **Specifies...** ***ssn*** — Only a subsystem number. Default routing must be enabled. ***x.y.z:ssn*** — Point code and subsystem, with the point code in dotted format. ***n:ssn*** — Point code and subsystem, with the point code in hexadecimal format. GT ***n/digits*** — Global title, where ***n*** is the global title format (from 1 - 4) and ***digits*** is the global title (each digit is in the range 0 - F). GT ***n/digits:ssn*** — Global title, where ***n*** is the global title format (from 1 - 4) and ***digits*** is the global title (each digit is in the range 0 - F). Also specifies a specific subsystem number, ***ssn***. NONE — No address.

Step	Action
4	Start the server side application first with the following syntax: `sccldemo -b 1 -p 0 -n 254 -s` This application uses TX board 1 (-b 1), SAP ID 0 (-p 0), and listens on subsystem 254 (-n 254). The -s option specifies that it is acting as a server and not sending any messages.

5	Start the client side application with the following syntax:
	`sccldemo -b 2 -p 0 -n 254 1.1.1:254`
	This application uses TX board 2 (-b 2), SAP ID 0 (-p 0), and listens on subsystem 254 (-n 254). The called address specifies that a message is sent to point code 1.1.1, subsystem 254. Since no calling address is specified in the command line, the default address is used, and the SCCP message contains a calling address with the point code and subsystem (1.1.2, subsystem 254).
	The client side application shows the following output:
	`Sending Connection Request`
	`Connection Confirmation received for connection 1`
	`<5 second delay>`
	`Sending Data over Connection`
	`<5 second delay>`
	`Sending Release Request`
	The server side application shows the following output:
	`Sending Indication Received`
	`Sending Connection Response`
	`<5 second delay>`
	`Data Indication received on connection 1`
	`<5 second delay>`
	`Release Indication received on connection 1`
	If the server side application does not receive the message, refer to the **Details** section for suggestions.

Details

Each SCCP connectionless message contains a destination address (the called address) and a source address (the calling address). The client application must specify a destination for the message, which is the called address. The most common address consists of a point code and a subsystem.

If no calling address is specified, then the message is sent with a default calling address of the client application point code and subsystem.

sccodemo uses SCCP management to determine the address type (ANSI or ITU) of the SAP ID in use. It adjusts the called and calling addresses appropriately.

Troubleshoot as follows:

- Use the *txalarm* utility to confirm that the two TX boards have active links between them.
- Check the SCCP configuration file to confirm that the correct destination point code and subsystem are configured.
- If using a global title for a called address, try a simple point code/subsystem combination first, and then try a global title.

SCCP configuration utility: sccpcfg

Scans the SCCP configuration text file and downloads the SCCP configuration to the SCCP task on the TX board at boot time.

Usage

```
sccpcfg options
```

Requirements

- A computer with a TX board installed
- Windows or UNIX
- Natural Access
- NMS SS7

Procedure

Use this procedure to run *sccpcfg*:

Step	Action
1	From the command line prompt, navigate to the *\nms\tx\samples\sccp\sccpcfg* directory under Windows or the */usr/bin* directory under UNIX.
2	Enter the following command: `sccpcfg options` where ***options*** include: **Options** / **Description** -b ***board*** — Board number to which the SCCP configuration is downloaded. Default = 1. -f ***filename*** — Name and location of the SCCP configuration file to be downloaded. The SCCP configuration program scans the information in the ASCII file (specified with the -f option) and downloads this information to the task on the TX board.

Details

The SCCP configuration utility is available in both source code and executable formats. Use *sccpcfg* if you want your application to load the SCCP configuration to the TX board.

SCCP layer status: sccpmgr

Monitors the status of the SCCP layer after the SCCP configuration is downloaded to the TX board with *sccpcfg*. The SCCP manager (*sccpmgr*) provides a command line interface that enables an application to set alarm levels, trace buffers, and view and reset SCCP statistics.

Usage

```
sccpmgr -b board
```

Requirements

- A computer with a TX board installed
- Windows or UNIX
- Natural Access
- NMS SS7

Procedure

Use this procedure to run *sccpmgr*:

Step	Action
1	From the command line prompt, navigate to the *\nms\tx\samples\sccp\sccpmgr* directory under Windows or the */usr/bin* directory under UNIX.
2	Enter the following command: `sccpmgr -b board` where ***board*** is the TX board number on which the SCCP layer is loaded.

The *sccpmgr* program supports the following commands:

Command	Description
ALARMLVL ***options***	Sets the current alarm output level. Valid range for ***options*** is 0 - 3.
GENERAL CONFIG	Displays general configuration.
GENERAL STATS ***[reset]***	Displays general statistics and optionally resets them to zero (***reset***) after fetch.
NSAP ***sapno*** STATS ***[reset]***	Displays statistics on a network SAP (***sapno***) and optionally resets them to zero (***reset***) after fetch.
ROUTE ***pointcode*** STATUS	Displays current status of remote signaling point and its associated subsystems.
ROUTE ***pointcode*** CONFIG	Displays current configuration for remote signaling point.
USAP ***sapno*** STATS ***[reset]***	Displays statistics on an application SAP (***sapno***) and optionally resets them to zero (***reset***) after fetch.
USAP ***sapno*** STATUS	Displays status on an application SAP (***sapno***).
TRACE ***on|off***	Turns buffer tracing ON or OFF.
BOARD ***board***	Switches to a new target board (***board***).
? [***command***]	Lists available commands or parameters of a specific command (***command***).
QUIT	Quits the application.

Details

The SCCP manager program is available in both source code and executable formats. The source code demonstrates the use of SCCP management for developers who want to integrate management of the SS7 SCCP layer into their own configuration management systems.

8 Common parameters

Parameters overview

This section specifies the layout of the common parameters contained in the messages passed between the application and the SS7 SCCP layer implementation.

Data types

The SCCP service uses the following conventions for data types:

SCCP type	C implementation	Description
U8	unsigned char	unsigned 8-bit quantity
U16	unsigned short	unsigned 16-bit quantity
S16	short	signed 16-bit quantity
U32	unsigned long	unsigned 32-bit quantity
S32	long	signed 32-bit quantity

Address parameter

Represents a called party address or calling party address.

```
typedef struct sccpAddr    /* Called/Calling Party Address   */
{
  U8  presind;             /* presence indicator             */
  U8  spare1;              /* spare for alignment            */
  U8  swType;              /* switch type (ANSI/ITU-T)       */
  U8  subsystemInd;        /* subsystem indicator            */
  U8  pointCodeInd;        /* point code indicator           */
  U8  glTitleInd;          /* global title indicator         */
  U8  routingInd;          /* routing indicator              */
  U32 pointCode;           /* point code                     */
  U8  natIntInd;           /* national/international ind.    */
  U8  subsystem;           /* subsystem number               */
  U8  glTransType;         /* global title translation type  */
  U8  encoding;            /* address encoding scheme        */
  U8  numPlan;             /* numbering plan                 */
  U8  natAddrInd;          /* nature of address indicator    */
  U8  spare2;              /* spare for alignment            */
  U8  glTitleLen;          /* length of global title         */
  U8  glTitle[MAX_GLT_SZ]; /* Global Title                   */
} SccpAddr;
```

The fields in the SccpAddr structure are encoded as follows:

Field	Value
presind	0 = NOT_PRESENT Field is not present in incoming message or not included in outgoing message 1 = PRESENT Field is present in incoming message or to be included in outgoing message
swtype	1 = SW_ITU ITU operation 2 = SW_ANSI ANSI operation
subsystemInd	0x00 = SUBSYS_NONE No subsystem number in address 0x01 = SUBSYS_INC Address contains subsystem number
pointCodeInd	0x00 = PTCODE_NONE No point code in address 0x01 = PTCODE_INC Address contains point code
glTitleInd	0x00 = GLT_NONE No global title in address 0x01 = GLT_TT_NP_E ANSI only - Global title includes translation type, numbering plan, and encoding 0x02 = GLT_TT ANSI only - Global title includes translation type only 0x01 = GLT_ITU_FMT1 ITU only - Global title includes encoding and nature of address only 0x02 = GLT_ITU_FMT2 ITU only - Global title includes translation type only 0x03 = GLT_ITU_FMT3 ITU only - Global title includes translation type, numbering plan, and encoding 0x04 = GLT_ITU_FMT4 ITU only - Global title includes translation type, numbering plan, encoding, and nature of address
routingInd	0x00 = ROUTE_GLT Route uses global title only 0x01 = ROUTE_PC_SN Route uses point code + SSN
pointCode	A 32-bit quantity of which the least significant 24 bits (ANSI or ITU-T) or the least significant 14 bits (ITU-T) are used. For example, an ANSI point code represented by the decimal string 1.4.7 is encoded as hex 0x00010407. If a point code is not included in the called address, then the default point code is used.
natIntInd	0x00 = ADDRIND_INT International address indicator 0x01 = ADDRIND_NAT National address indicator

Field	Value
subsystem	0x00 = SUBSYS_NONE Subsystem unknown or not used 0x01 = SUBSYS_SCCPMGMT SCCP management subsystem 0x03 = SUBSYS_ISUP ISUP subsystem 0x04 = SUBSYS_OMAP Operations, maintenance, and administration 0x05 = SUBSYS_MAP Mobile application part 0x06 = SUBSYS_HLR Home location register 0x07 = SUBSYS_VLR Visitor location register 0x08 = SUBSYS_MSC Mobile switching center 0x09 = SUBSYS_EIR Equipment identification register 0x0A = SUBSYS_AUTH Authentication center Other values in range 0x0B - 0xFF are also allowed.
glTransType	Translation type when the global title indicator field (glTitleInd) specifies that the global title includes translation type. Any 8-bit value [0x00 - 0xFF] is allowed.
encoding	Specifies whether the number of digits in the addrSig field is even or odd. If the number of digits is even, the last octet contains 2 digits. If the number of digits is odd, the last octet contains only one digit and the most significant 4 bits are not used. 0x00 = ENC_UNKNOWN Encoding unknown 0x01 = ENC_BCD_ODD BCD, odd number of digits 0x02 = ENC_BCD_EVEN BCD, even number of digits
numPlan	0x00 = NP_UNK Unknown 0x01 = NP_ISDN ISDN/telephony - E.164/E.163 0x02 = NP_TEL Telephony numbering - E.163 0x03 = NP_DATA Data numbering - X.121 0x04 = NP_TELEX Telex numbering - recommendation F.69 0x05 = NP_MARITIME Maritime mobile numbering 0x06 = NP_LANDMOB Land mobile numbering 0x07 = NP_ISDNMOB ISDN/mobile numbering 0x08 = NP_NATIONAL National standard numbering 0x09 = NP_PRIVATE Private numbering 0x0f = NP_EXT Reserved for extension
natAddrInd	0x01 = NATIND_SUBS Subscriber number 0x03 = NATIND_NATL National number 0x04 = NATIND_INTNATL International number
glTitleLen	Encoded as the byte length of the BCD-encoded global title. For example, a ten-digit global title is BCD-encoded as a value with a five-byte length. glTitleLen is set to five.
glTitle	The BCD-encoded global title.

Global title is encoded as follows:

Octet 1	2nd address digit	1st (most significant) address digit
...	...	...
Octet n	m + 1th address digit or filler	mth address digit

Each digit is encoded with the following bit pattern:

Bit pattern	Digit/signal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	spare
1011	code 11
1100	code 12
1101	spare
1110	spare
1111	ST

Cause value parameter

Indicates the reason for an error, connection rejection, clearing, or reset.

```
typedef U8 SccpCauseVal;
```

The cause value has different meanings depending on the message in which it is included. Pre-defined values for various messages are shown in the following examples. Application-specific values are also allowed.

Error cause values (error message)

ECDESTLRN	0	Unassigned destination local reference number
ECSRCLRN	1	Inconsistent source local reference number
ECPOINTCODE	2	Point code mismatch
ECSERVCLASS	3	Service class mismatch
ECUNQUALIFIED	4	Unqualified

Refusal cause values (connection refused message)

REFCENDUSER	0	End user originated
REFCUSERCONG	1	End user congested
REFCUSERFAIL	2	End user failed
REFCSCCPUSER	3	SCCP user originated
REFCDESTUNK	4	Destination unknown
REFCDESTINACC	5	Destination inaccessible
REFCQOSPERM	6	QOS not available (permanent)
REFCQOSTRANS	7	QOS not available (transient)
REFCACCFAIL	8	Access failure
REFCACCCONG	9	Access congestion
REFCSUBSFAIL	10	Subsystem failure
REFCSUBSCONG	11	Subsystem congestion
REFCCONNTIMER	12	Connection timer expired
REFCBADDATA	13	Inconsistent user data
REFCNOTOBTAIN	14	Not obtainable
REFCUNQUALIFIED	15	Unqualified
REFCHOPCNT	16	Hop counter failure
REFCSCCPFAIL	17	SCCP failure
REFCNOTRANS	18	No translation for address
REFCUNEQUSER	19	Unequipped user

Release cause values (release message)

RELCENDUSER	0	End user originated
RELCUSERBUSY	1	End user busy
RELCUSERFAIL	2	End user failed
RELCSCCPUSER	3	SCCP user originated
RELCREMPROC	4	Remote procedure error
RELCCONNDATA	5	Inconsistent connection data
RELCACCFAIL	6	Access failure
RELCACCCONG	7	Access congestion
RELCSUBSFAIL	8	Subsystem failure
RELCSUBSCONG	9	Subsystem congestion
RELCNETFAIL	10	Network failure
RELCNETCONG	11	Network congestion
RELCRESETTIMER	12	Reset timer expired
RELCINACT	13	Inactivity timer expired
RELCNOTOBTAIN	14	Not obtainable
RELCUNQUALIFIED	15	Unqualified
RELCSCCPFAIL	0x10	SCCP failure

Reset cause values (reset request message)

RESCENDUSER	0	End user originated
RESCSCCPUSER	1	SCCP user originated
RESCPSERR	2	Sequence error - bad P(s)
RESCPRERR	3	Sequence error - bad P(r)
RESCREMWIN	4	Message out of window
RESCREMPSERR	5	Bad P(s) after reinit
RESCREMGEN	6	General remote proc error
RESCREMUSER	7	Remote end user operational
RESCNETWORK	8	Network operational
RESCACCESS	9	Access operational
RESCNETCONG	10	Network congestion
RESCNOTOBTAIN	11	Not obtainable
RESCUNQUALIFIED	12	Unqualified

Return cause values (class 0 and 1 returned messages)

RETCGENTRANS	0	No translation for address of this type
RETCNOTRANS	1	No translation for this address
RETCSUBSCONG	2	Subsystem congestion
RETCSUBSFAIL	3	Subsystem failure
RETCUNQUIP	4	Unequipped user
RETCNETFAIL	5	Network failure
RETCNETCONG	6	Network congestion
RETCUNQUALIFIED	7	Unqualified
RETCHOPCNT_ANS92	8	Hop counter violation (ANS92)
RETCMSGXPORT	8	Error in message transport
RETCLOCALPROC	9	Error in local processing
RETCREASSEMBLY	10	Destination cannot do re-assembly
RETCSCCPFAIL	11	SCCP failure
RETCHOPCNT	12	Hop counter violation (ITU and ANS96)
RETCSEGMNOTSUPP	13	Error in message transport
RETCSEGMFAIL	14	Error in local processing
RETCBADISNI	0xF9	Invalid ISNI routing request
RETCAUTH	0xFA	Unauthorized message
RETCINCOMPAT	0xFB	Message incompatibility
RETCNOISNI	0xFC	Cannot do ISNI constrained routing
RETCREDISNI	0xFD	Redundant ISNI constrained routing information
RETCISNIID	0xFE	Cannot do ISNI identification

Credit parameter

Contains the window size (for example, the number of messages that can be sent without receiving an acknowledgment) on a particular SCCP connection. This field is found in the SCCP connection request (CR) and connection confirm (CC) messages.

```
typedef U8 SccpCredit;
```

The credit field is encoded as an unsigned number in the range of 1 - 255.

Data parameter

Passes transparently user data to the far end point.

```
typedef struct sccpData
{
  U8   presind;              /* presence indicator       */
  U8   spare1;               /* spare for alignment      */
  S16  dataLen;              /* length of data           */
  U8   data[MAX_DATA_SZ];    /* user data                */
} SccpData;
```

The data field is binary data passed transparently to the far end. The byte length of the data is specified in the dataLen field.

Note: The maximum message length may not be supported for all message types. Refer to the appropriate ANSI or ITU-T standard for maximum data lengths for messages.

The presind field must be set to indicate whether data is present or not in the message.

End of sequence parameter

Indicates to the SCCP layer that this is the end of a connectionless sequence and the SCCP layer can release resources allocated to this sequence. This parameter is used only with the class 1 service unitdata request and has no meaning on incoming unitdata indications.

```
typedef U8 SccpEOS;
```

This field is encoded as follows:

0	EOS_NO	Not end of sequence
1	EOS_YES	End of sequence

Expedited data selection parameter

Selects expedited data service on a connection.

```
typedef U8 SccpExpDatSel;
```

This field is encoded as follows:

0	EDS_NONE	No expedited data
1	EDS_REQ	Expedited data requested

Importance parameter

Selects the importance level of the message that enables SCCP to restrict messages (such as during congestion) based on their importance (ITU-96 and later).

This field is encoded as an unsigned integer with a range of 0 - 7, where 0 indicates the least important message and 7 indicates the most important message.

Protocol class parameter

Specifies a protocol class in connection request or confirm messages or in connectionless unitdata/extended unitdata messages.

```
typedef struct sccpProtoClass
{
  U8   classInd;      /* class indicator                    */
  U8   msgHandling;   /* message handling indicator         */
  U8   spare1;        /* spare for alignment                */
} SccpProtoClass;
```

The fields in the SccpProtoClass structure are encoded as follows:

Field	Value
classInd	0x00 = SCCP_CLASS0 0-basic connectionless 0x01 = SCCP_CLASS1 1-sequenced connectionless 0x02 = SCCP_CLASS2 2-basic connection-oriented 0x03 = SCCP_CLASS3 3-connection-oriented w/flow
msgHandling	0x00 = MSG_DISCARD Discard message on error 0x08 = MSG_RETURN Return message on error

Receipt confirmation selection parameter

Requests the receipt confirmation service on a connection.

```
typedef U8 SccpRecConfSel;
```

This field is encoded as follows:

0	RCS_NONE	No receipt confirmation
1	RCS_REQ	Receipt confirmation requested

Receive sequence number parameter

Specifies the next expected sequence number in a data acknowledgment message.

```
typedef U8 SccpNextRS;
```

The SccpNextRs field is coded to a value in the range of 0 - 127.

Subsystem multiplicity indicator parameter

Indicates whether or not a subsystem is replicated on another node. This parameter is used in subsystem coordination indications or confirmations.

```
typedef U8 SccpSmi;
```

This field is encoded as follows:

0x00	SMI_UNKNOWN	Unknown
0x01	SMI_SOLO	Subsystem not replicated
0x02	SMI_DUP	Subsystem is replicated
0x10	SMI_DENIED	Denial of coordinate state change

Subsystem number parameter

Indicates whether or not a subsystem is replicated on another node. This parameter is used in subsystem coordination indications or confirmations.

```
typedef U8 SccpSsn;
```

This field is encoded as a standard SCCP subsystem number.

Subsystem status indicator parameter

Identifies the new state in subsystem or point code state change indications.

```
typedef U8 SccpStatus;
```

This field is encoded as follows:

0x00	SP_ACC	Signaling point accessible
0x01	SP_INACC	Signaling point inaccessible
0x06	SP_INACC_NODROP	Signaling point inaccessible, connections not dropped
0x03	SS_OOS	Subsystem out of service
0x04	SS_IS	Subsystem in service

9 Messages

Messages overview

This section specifies the layout of all messages passed between the application and the SCCP layer implementation.

Parameters tagged with the (M) notation in the comment field are mandatory parameters. Parameters tagged with the (O) notation are optional.

Data types

The SCCP service uses the following conventions for data types:

SCCP type	C implementation	Description
U8	unsigned char	unsigned 8-bit quantity
U16	unsigned short	unsigned 16-bit quantity
S16	short	signed 16-bit quantity
U32	unsigned long	unsigned 32-bit quantity
S32	long	signed 32-bit quantity

Connection information message

The connection information (SccpConnInfo) message is returned to the application by the SCCP layer with a SCCPCONNAUDCFM event, in response to an application call to **SCCPConnAuditRqst**. The connection information in this message includes the protocol class (class 2 or class 3), the calling and called party addresses, the connection type, and current connection state.

```
typedef struct sccpConnInfo             /* Connection Info (audit confirm)  */
{
    SccpProtoClass  protoClass;         /* protocol class                   */
    SccpAddr        calledPty;          /* called party address             */
    SccpAddr        callingPty;         /* calling party address            */
    U8              reason;             /* reserved for future use          */
    U8              connState;          /* connection state                 */
#define SCCP_CONN_RDY_ST 0x00           /* ready (idle) state               */
#define SCCP_CONN_CON_ST 0x01           /* connecting state                 */
#define SCCP_CONN_DTX_ST 0x02           /* data Transfer state              */
#define SCCP_CONN_RCG_ST 0x03           /* reset (calling side) state       */
#define SCCP_CONN_RCD_ST 0x04           /* reset (called side) state        */
#define SCCP_CONN_RBT_ST 0x05           /* reset (both sides) state         */
#define SCCP_CONN_RLS_ST 0x06           /* releasing state                  */
    U8              connType;           /* connection type                  */
#define SCCP_CONN_ORIG   0x01           /* originating node                 */
#define SCCP_CONN_DEST   0x02           /* destination node                 */
#define SCCP_CONN_INTR   0x04           /* intermediate node                */
    SccpCredit      credit;             /* proposed window size             */
} SccpConnInfo;
```

Connect request message

The connect request message is used by the application to request establishment of a connection (**SCCPConnectRqst**) or acceptance of an incoming connection (**SCCPConnectResp**) when employing the SCCP connection-oriented services (class 2 or 3). This message is also received by the application to indicate an incoming connect request from a far end point (SCCPCONNIND) or to indicate an incoming connect confirmation from a far end point (SCCPCONNCFM).

```
typedef struct sccpConnRqst
{
  SccpProtoClass  protoClass;   /* protocol class           (M)    */
  SccpAddr        calledPty;    /* called party address     (M)    */
  SccpAddr        callingPty;   /* calling party addr       (O)    */
  SccpCredit      credit;       /* proposed window size     (M)    */
  SccpRecConfSel  rcs;          /* recpt confirm selected   (M)    */
  SccpExpDatSel   eds;          /* expedited data selector  (M)    */
  U8              spare1;       /* spare for alignment             */
  SccpImportance  importance;   /* reserved for ITU-96 importance  */
  SccpData        data;         /* data                     (O)    */
} SccpConnRqst;
```

Coordination request message

The coordination request message is used by the application to request a coordinate state change to a subsystem (**SCCPCoordRqst**) or to accept a coordinated state change (**SCCPCoordResp**) request from a backup signaling point. This message is also received by the application to indicate an incoming coordinated state change request from a backup signaling point (SCCPCOORDIND) or to indicate an incoming coordination confirmation from a backup signaling point (SCCPCOORDCFM).

This message is also used as an indication to the application of an unsolicited subsystem state change or signaling point state change.

```
typedef struct sccpCoordRqst
{
  SccpSsn    aSsn;   /* affected subsystem (M)                          */
  SccpSmi    smi;    /* subsystem multiplicity ind (M - Indication/Confirm
                      * only)                                           */
  SccpStatus status; /* New status (M - subsystem & SP state change
                      * indications only)                              */
  U8         spare1; /* spare for alignment                             */
} SccpCoordRqst;
```

Data acknowledge message

The data acknowledge message is used by the application to acknowledge one or more regular or a single expedited data packet from the far end (**SCCPDAckRqst**) when employing the SCCP connection-oriented services (class 2 or 3). This message is also received by the application to indicate an incoming regular or expedited data ACK packet from a far end point (SCCPDACKIND).

```
typedef struct sccpDAckRqst
{
  SccpNextRS  rsn;     /* rcv seq. # (M-normal data only- not used for
                        * expedited                                     */
  SccpCredit  credit;  /* window size (M-normal data only- not used for
                        * expedited)                                    */
  U16         spare1;  /* spare for alignment                          */
} SccpDAckRqst;
```

Data request message

The data request message is used by the application to send a regular or expedited data packet to the far end (**SCCPDataRqst** and **SCCPEDataRqst**) when employing the SCCP connection-oriented services (class 2 or 3). This message is also received by the application to indicate an incoming regular or expedited data packet from a far end point (SCCPDATIND and SCCPEDATIND).

```
typedef struct sccpDataRqst
{
  SccpData   data;    /* data (M)        */
} SccpDataRqst;
```

Released message

The released message is used by the application to request clearing of an existing connection (**SCCPReleaseRqst**) or is received by the application to indicate the far end cleared an existing connection (SCCPRELIND).

This message is also used to refuse an incoming connection request.

```
typedef struct sccpReleased
{
  U8              relOrig;    /* originator of release                  */
  SccpCauseVal    relCause;   /* release cause (M)                      */
  U16             spare1;     /* spare for alignment                    */
  SccpAddr        rspPty;     /* Release indication only-
                               * addr of responding party when outgoing
                               * connection refused (O)                 */
  SccpImportance  importance; /* reserved for ITU-96 importance (O)     */
  SccpData        data;       /* data (O)                               */
} SccpReleased;
```

Reset request message

The reset message is used by the application to request resetting of an existing connection (**SCCPResetRqst**) or is received by the application to indicate the far end requested a reset of an existing connection (SCCPRESETIND). This message is not used in **SCCPResetResp** and the SCCPRESETCFM event.

```
typedef struct sccpResetRqst
{
  SccpCauseVal  resCause;  /* release cause (M-Rqst & Indication only)   */
  U8            resOrig;   /* origin of reset                            */
  U16           spare1;    /* spare for alignment                        */
} SccpResetRqst;
```

The resOrig field is coded as follows:

```
#define  ORIGNETWORK      0x00     /* network originated                 */
#define  ORIGENDUSER      0x01     /* user originated                    */
```

Unitdata request message

The unitdata request message is used by the application to send a packet to the far end (**SCCPUDataRqst**) when employing the SCCP connectionless services (class 0 or 1) or is received by the application to indicate an incoming data packet from a far end point (SCCPUDATIND).

```
typedef struct sccpUdataRqst
{
  SccpProtoClass  protoClass; /* protocol class       (M)                */
  SccpAddr        calledPty;  /* called party address (M)                */
  SccpAddr        callingPty; /* calling party address(M)                */
  SccpEOS         eos;        /* end-of-sequence flag (M - class 1 only) */
  U8              spare1;     /* spare for alignment                     */
  U8              spare2;     /* spare for alignment                     */
  U8              spare3;     /* spare for alignment                     */
  SccpImportance  importance; /* reserved for ITU-96 importance (O)      */
  SccpIsni        isni;       /* reserved for Ansi-96 ISNI (O)           */
  SccpData        data;       /* data                 (M)                */
} SccpUdataRqst;
```

10 SCCPRetrieveMessage event reference

Event reference overview

SCCPRetrieveMessage is unique in that the event structure associated with a received message depends on the type of message received from the SCCP layer. This section provides a description of each possible event, as well as the structures that are returned with each event.

SCCPCONNAUDCFM

Returned in response to a call to **SCCPConnAuditRqst**. It contains information about an active connection or an indication that there are no more connections to audit.

Usage

SccpRcvInfoBlk

Field	Description
board	Not used. The CTAHD handle that delivered the event identifies the board that originated the event.
indType	SCCPCONNAUDCFM
evntType	SCCP_CONNAUDCFM Valid connection data is returned in the SccpConnInfo structure. SCCP_CONNAUDEOF No more connections to audit. No data is returned in the SccpConnInfo structure.
suId	Not used.
connId	Connection for which data is returned in the SccpConnInfo structure.
opc	Not used.

SccpConnInfo

Field	Description
protoClass	Protocol class (class 2 or class 3) used on this connection.
calledPty	Called party address from the connection establishment message.
callingPty	Calling party address from the connection establishment message.
reason	Reserved for future use.
connState	Current connection state. Must be one of the following: SCCP_CONN_CON_ST Connecting state. SCCP_CONN_DTX_ST Data transfer state. SCCP_CONN_RCG_ST Calling side resetting state. SCCP_CONN_RCD_ST Called side resetting state. SCCP_CONN_RBT_ST Both sides resetting state. SCCP_CONN_REL_ST Releasing state.

Field	Description
importance	Connection type. Must be one of the following: SCCP_CONN_ORIG Caller is on originating side. SCCP_CONN_DEST Caller is on destination side. SCCP_CONN_INTR Intermediate node (both calling and called are remote).
credit	For class 3 connections, contains the window size agreed to by the far responding party for this connection.

SCCPCONNCFM

Indicates that a connection request was accepted by the called party. The SCCPCONNCFM event returns a data structure of type SccpConnRqst.

Usage

SccpRcvInfoBlk

Field	Description
board	Not used. The CTAHD handle that delivered the event identifies the board that originated the event.
indType	SCCPCONNCFM
evntType	Not used.
suId	Not used.
connId	Connection with which this event is associated. Refer to *Connection IDs* on page 29 for more information. The content of the spConnId subfield must be saved by the application and included in subsequent requests associated with this connection.
opc	Not used.

SccpConnRqst

Field	Description
protoClass	Protocol class (classInd subfield) set by the sender in the connection confirm (CC) protocol message. Even if the caller originally requested a class 3 connection, the sender can negotiate this down to a class 2 connection. The msgHandling subfield is not used for connection-oriented data transfer. Mandatory field.
calledPty	
callingPty	Content of the called party address field from the incoming connection confirm (CC) protocol message, if present. This is typically the responder's address, which can be different from the called party address in the caller's original connect request. Optional field.
credit	For class 3 connections, contains the window size agreed to by the far responding party for this connection. Optional field.
rcs	Not used.
eds	Not used.
importance	Reserved for future use.
data	Contains any user data included by the responding party in the connection confirm (CC) protocol message. Optional field.

SCCPCONNIND

Indicates that a new connection request was received from a remote node. The SCCPCONNIND event returns a data structure of type SccpConnRqst.

Usage

SccpRcvInfoBlk

Field	Description
board	Not used. The CTAHD handle that delivered the event identifies the board that originated the event.
indType	SCCPCONNIND
evntType	Not used.
suId	Not used.
connId	Connection with which this event is associated. Refer to *Connection IDs* on page 29 for more information. The content of the spConnId subfield must be saved by the application and included in subsequent requests associated with this connection.
opc	Point code from the routing label of the incoming connection request (CR) message.

SccpConnRqst

Field	Description
protoClass	Protocol class (classInd subfield) from the incoming CR protocol message. The msgHandling subfield is not used for connection-oriented data transfer. Mandatory field.
calledPty	Content of the called party address field from the incoming CR message. Mandatory field.
callingPty	If present, contains the content of the calling party address field from the incoming CR protocol message. Optional field.
credit	For class 3 connections, contains the window size proposed by the calling party for this connection. Optional field.
rcs	Not used.
eds	Not used.
importance	Reserved for future use.
data	User data included by the calling party in the CR protocol message. Optional field.

SCCPCOORDCFM

Indicates that a subsystem-out-of-service grant message was received from the application's mate subsystem indicating that the application may go out of service. The SCCPCOORDCFM event returns a data structure of type SccpCoordRqst.

Usage

SccpRcvInfoBlk

Field	Description
board	Not used. The CTAHD handle that delivered the event identifies the board that originated the event.
indType	SCCPCOORDCFM
evntType	Not used.
suId	Application service user ID that identifies with which SCCP user SAP this message is associated.
connId	Not used.
opc	Not used.

SccpCoordRqst

Field	Description
aSsn	Affected subsystem number from the received subsystem-out-of-service grant (SOG) message. Mandatory field.
smi	Not used.
status	Not used.

SCCPCOORDIND

Indicates that a coordinated state change request was received from the application's mate subsystem indicating that the mate is requesting to go out of service. The SCCPCOORDIND event returns a data structure of type SccpCoordRqst.

Usage

SccpRcvInfoBlk

Field	Description
board	Not used. The CTAHD handle that delivered the event identifies the board that originated the event.
indType	SCCPCOORDIND
evntType	Not used.
suId	Application service user ID that identifies with which SCCP user SAP this message is associated.
connId	Not used.
opc	Not used.

SccpCoordRqst

Field	Description
aSsn	Affected subsystem number from the received subsystem-out-of-service request (SOR) message. Mandatory field.
smi	Not used.
status	Not used.

SCCPDACKIND

Indicates that a data acknowledgement message was received from a far end point.

Usage

SccpRcvInfoBlk

Field	Description
board	Not used.
indType	SCCPDACKIND
evntType	Not used.
suId	Not used.
connId	Connection with which this event is associated. Refer to *Connection IDs* on page 29 for information.
opc	Not used.

SCCPDATIND and SCCPEDATIND

Indicates a received data or expedited data packet for a connection. The SCCPDATIND and SCCPEDATIND events return a data structure of type SccpDataRqst.

Usage

SccpRcvInfoBlk

Field	Description
board	Not used. The CTAHD handle that delivered the event identifies the board that originated the event.
indType	SCCPDATIND or SCCPEDATIND
evntType	Not used.
suId	Not used.
connId	Connection with which this event is associated. Refer to *Connection IDs* on page 29 for more information.
opc	Not used.

SccpDataRqst

Field	Description
data	User data received. Mandatory field.

SCCPINACTIND

Indicates no application data traffic on this connection for the AIC_TIMER period. The application must respond by calling **SCCPInactResp** to keep the connection alive. If the application wants to release the specified connection, it must call **SCCPReleaseRqst** instead.

This event is received only if the receiver's SAP is configured with application inactivity control enabled (INACT_CONTROL_TRUE).

Usage

SccpRcvInfoBlk

Field	Description
board	Not used. The CTAHD handle that delivered the event identifies the board that originated the event.
indType	SCCPINACTIND
evntType	Not used.
suId	Not used.
connId	Connection for which there has been no activity. This same connection ID must be passed to **SCCPInactResp** to preserve the connection.
opc	Not used.

SCCPPCSTIND

Indicates that the state of a remote concerned signaling point has changed to accessible or inaccessible. The SCCPPCSTIND event returns a data structure of type SccpCoordRqst.

Usage

SccpRcvInfoBlk

Field	Description
board	Not used. The CTAHD handle that delivered the event identifies the board that originated the event.
indType	SCCPPCSTEIND
evntType	Not used.
suId	Application service user ID that identifies with which SCCP user SAP this message is associated.
connId	Not used.
opc	Point code of the concerned signaling point whose state changed.
mopc	Originating point code for the point code state indication (multiple OPC support).

SccpCoordRqst

Field	Description
aSsn	Not used.
smi	Not used.
status	Mandatory field. Indicates one of the new signaling point states: SP_ACC SP now accessible. SP_INACC SP now inaccessible, all connections to that destination have been dropped. SP_INACC_NODROP SP now inaccessible, but connections to that destination have been retained. • SP_CONG1 Signaling point code has entered level 1 congestion for outbound messages. • SP_CONG2 Signaling point code has entered level 2 congestion for outbound messages. • SP_CONG3 Signaling point code has entered level 3 congestion for outbound messages. • SP_CONG_OFF Signaling point code is no longer in congestion for outbound messages.

SCCPRELIND

Indicates that an existing connection was released by the remote party or that a new connection request was refused either by the local stack or by the called party. The SCCPRELIND event returns a data structure of type SccpReleased.

Usage

SccpRcvInfoBlk

Field	Description
board	Not used. The CTAHD handle that delivered the event identifies the board that originated the event.
indType	SCCPRELIND
evntType	Not used.
suId	Not used.
connId	Connection that is released or refused.
opc	Not used.

SccpReleased

Field	Description
relOrig	Originator of the release or connection refusal, either the network (for example, the local or remote stack, as in a protocol or routing error) or the remote user application. Mandatory field.
relCause	Reason for the release or connection refusal, either from the local stack (in the case where the connection was released or refused locally) or from the received RLSD or CREF message. Mandatory field.
rspPty	Content of the responding party address field, if present. Optional field.
importance	Reserved for future use.
data	User data included by the far party in the RLSD or CREF protocol messages. Optional field.

SCCPRESETCFM

Indicates that a reset request issued for a connection was accepted by the far party. The SCCPRESETCFM event returns a data structure of type SccpResetRqst.

Usage

SccpRcvInfoBlk

Field	Description
board	Not used. The CTAHD handle that delivered the event identifies the board that originated the event.
indType	SCCPRESETCFM
evntType	Not used.
suId	Not used.
connId	Connection with which this event is associated. Refer to *Connection IDs* on page 29 for more information.
opc	Not used.

SccpResetRqst

Field	Description
relOrig	Not used.
relCause	Echoes the reset cause from the original reset request. Mandatory field.

SCCPRESETIND

Indicates that a reset request for a connection was received from the far party. The SCCPRESETIND event returns a data structure of type SccpResetRqst.

Usage

SccpRcvInfoBlk

Field	Description
board	Not used. The CTAHD handle that delivered the event identifies the board that originated the event.
indType	SCCPRESETIND
evntType	Not used.
suId	Not used.
connId	Connection with which this event is associated. Refer to *Connection IDs* on page 29 for more information.
opc	Not used.

SccpResetRqst

Field	Description
relOrig	Not used.
relCause	Reason that the connection is reset. Mandatory field.

SCCPRUNSTATEIND

Indicates the redundancy state of the board to the local application. This event is received immediately after binding (if the board state has already been determined), or any time the redundancy state changes between PRIMARY and BACKUP states. The SCCPRUNSTATEIND event does not return any data other than the content of the SccpRcvInfoBlk. For more information, refer to *Handling redundancy events* on page 40.

Usage

SccpRcvInfoBlk

Field	Description
board	Not used. The CTAHD handle that delivered the event identifies the board that originated the event.
indType	SCCPRUNSTATEIND
evntType	New redundancy state of the board: SPRS_STANDALONE SPRS_PRIMARY SPRS_BACKUP
suId	Application service user ID that identifies with which SCCP user SAP this message is associated.
connId	Not used.
opc	Point code of the concerned signaling point whose state changed.

SCCPSTAIND

Indicates that a unitdata request issued by the application could not be delivered, either by the local SCCP layer or by the remote layer (a UDTS or XUDTS message was returned). This status indication is returned only if the message handling options in the original unitdata request (protoClass.msgHandling) specified the message return (MSG_RETURN) option. The SCCPSTAIND event returns a data structure of type SccpUdataRqst.

Usage

SccpRcvInfoBlk

Field	Description
board	Not used. The CTAHD handle that delivered the event identifies the board that originated the event.
indType	SCCPSTAIND
evntType	Reason why the message could not be delivered.
suId	Application service user ID that identifies with which SCCP user SAP this message is associated.
connId	Not used.
opc	Originating point code from the routing label of the received unitdata (UDT) or extended unitdata (XUDT) message.

SccpUdataRqst

Field	Description
protoClass	Protocol class (classInd subfield) and message handling options (msgHandling subfield) from the original unitdata request. Mandatory field.
calledPty	Content of the called party address from the original unitdata request. Mandatory field.
callingPty	Content of the calling party address field from the original unitdata request, if present. Optional field.
eos	Content of the end-of-sequence field from the original unitdata request. Optional field.
importance	Reserved for future use.
isni	Reserved for future use.
data	User data from the original unitdata request. Mandatory field.

Return causes

The SccpRcvInfoBlk.evntType field contains one of the following return causes:

RETCGENTRANS	0x00	No translation for address of this type
RETCNOTRANS	0x01	No translation for this address
RETCSUBSCONG	0x02	Subsystem congestion
RETCSUBSFAIL	0x03	Subsystem failure
RETCUNQUIP	0x04	Unequipped user
RETCNETFAIL	0x05	Network failure
RETCNETCONG	0x06	Network congestion
RETCUNQUALIFIED	0x07	Unqualified
RETCHOPCNT_ANS92	0x08	Hop counter violation (ANS92)
RETCMSGXPORT	0x08	Error in message transport
RETCLOCALPROC	0x09	Error in local processing
RETCREASSEMBLY	0x0A	Destination cannot do re-assembly
RETCSCCPFAIL	0x0B	SCCP failure
RETCHOPCNT	0x0C	Hop counter violation (ITU and ANS96)
RETCSEGMNOTSUPP	0x0D	Error in message transport
RETCSEGMFAIL	0x0E	Error in local processing
RETCBADISNI	0xF9	Invalid ISNI routing request
RETCAUTH	0xFA	Unauthorized message
RETCINCOMPAT	0xFB	Message incompatibility
RETCNOISNI	0xFC	Cannot do ISNI constrained routing
RETCREDISNI	0xFD	Redundant ISNI constrained routing information
RETCISNIID	0xFE	Cannot do ISNI identification

SCCPSTATEIND

Indicates that the state of a remote concerned subsystem has changed to in-service or out-of-service. The SCCPSTATEIND event returns a data structure of type SccpCoordRqst.

Usage

SccpRcvInfoBlk

Field	Description
board	Not used. The CTAHD handle that delivered the event identifies the board that originated the event.
indType	SCCPSTATEIND
evntType	Not used.
suId	Application service user ID that identifies with which SCCP user SAP this message is associated.
connId	Not used.
opc	Point code of the concerned signaling point whose subsystem state changed.
mopc	Originating point code for the subsystem state indication (multiple OPC support).

SccpCoordRqst

Field	Description
aSsn	Subsystem number whose state is changed.
smi	Indicates whether or not the affected subsystem is replicated at another node.
status	New subsystem state, either SS_OOS (out-of-service) or SS_IS (in-service).

All fields in SccpCoordRqst are mandatory.

SCCPUDATIND

Indicates that a connectionless unitdata or extended unitdata message was received for the application. The SCCPUDATIND event returns a data structure of type SccpUdataRqst.

Usage

SccpRcvInfoBlk

Field	Description
board	Not used. The CTAHD handle that delivered the event identifies the board that originated the event.
indType	SCCPUDATIND
evntType	Not used.
suId	Application service user ID that identifies the SCCP user service access point with which this message is associated.
connId	Not used.
opc	Originating point code from the routing label of the received unitdata (UDT) or extended unitdata (XUDT) message.

SccpUdataRqstt

Field	Description
protoClass	Protocol class (classInd subfield) and message handling options (msgHandling subfield) from the received UDT or XUDT message. Mandatory field.
calledPty	Content of the called party address from the received UDT or XUDT message. Mandatory field.
callingPty	Content of the calling party address field from the received UDT or XUDT message, if present. Optional field.
eos	Not used.
importance	Reserved for future use.
isni	Reserved for future use.
data	Received user data. Optional field.

Index

A

Address parameter 124

addressing and routing 24

application inactivity control 30

application programming models 15

auditing connections 31

C

Cause value parameter 127

clearing connections 39

configuration 23, 120, 121

congestion 40

Connect request message 134

ConnectID 29

Connection information message 133

connectionless 36, 115

connections 29, 31, 36, 37, 38, 39, 118

contexts and queues 15

Coordination request message 134

Credit parameter 130

D

Data acknowledge message 134

Data parameter 130

Data request message 135

data transfer 36, 38

demonstration programs 115

E

End of sequence parameter 130

entity 12

events 137, 138, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 154, 155

Expedited data selection parameter 130

F

functions 13, 44, 66

G

global title translation 24, 26, 87, 124

I

Importance parameter 131

inbound congestion 41

instance ID 12

M

management functions 13, 63, 66

messages 133, 133, 134, 134, 134, 135, 135, 135, 136

N

Natural Access 15, 33

O

outbound congestion 40

out-of-service 20

P

parameters 123, 124, 127, 130, 130, 130, 130, 131, 131, 131, 131, 132, 132, 132

programming model 11

Protocol class parameter 131

Q

queues and contexts 15

R

RcvInfoBlk 58

Receipt confirmation selection parameter 131

Receive sequence number parameter 131

redundancy 40

Released message 135

Reset request message 135

routing 24

S

SAP 11
sccldemo 115
sccodemo 118
SCCP task 10
SccpAddrMapCfg 87
SccpAlarmControl 67
SccpAsciiMaskToBcd 68
SccpAsciiNumToBcd 69
SccpBcdMaskToAscii 70
SccpBcdNumToAscii 71
sccpcfg 120
SCCPCONNAUDCFM 138
SCCPConnAuditRqst 45
SCCPCONNCFM 140
SCCPConnectResp 46
SCCPConnectRqst 47
SCCPCONNIND 141
SCCPCOORDCFM 142
SCCPCOORDIND 143
SCCPCoordResp 48
SCCPCoordRqst 49
SCCPDACKIND 144
SCCPDAckRqst 50
SCCPDataRqst 51
SCCPDATIND and SCCPEDATIND 145
SccpDelAddrCfg 72
SccpDelRteCfg 73
SCCPEDataRqst 52
SccpGenCfg 91
SccpGenStats 76
SccpGetAddrCfg 74
SccpGetGenCfg 75
SccpGetGenStats 76
SccpGetNSapCfg 78
SccpGetNSapStats 79
SccpGetRteCfg 81
SCCPGetStats 53
SccpGetUSapCfg 82
SccpGetUSapStats 83
SccpGetUSapStatus 85
SCCPINACTIND 146
SCCPInactResp 54
SccpInitAddrCfg 87
SccpInitGenCfg 91
SccpInitNSapCfg 96
SccpInitRteCfg 99
SccpInitUSapCfg 102
SccpMgmtInit 106
SccpMgmtTerm 107
sccpmgr 121
SccpNSapCfg 96
SccpNSapStats 79
SCCPPCSTIND 147
SCCPReleaseRqst 55
SCCPRELIND 148
SCCPRESETCFM 149
SCCPRESETIND 150
SCCPResetResp 56
SCCPResetRqst 57
SCCPRetrieveMessage 58, 137
SccpRouteCfg 99
SCCPRUNSTATEIND 151
SccpSetAddrCfg 108
SccpSetGenCfg 109
SccpSetNSapCfg 110
SccpSetRteCfg 112
SccpSetUSapCfg 113
SCCPSTAIND 152
SCCPSTATEIND 154
SCCPStateRqst 60
SccpStats 53
SccpTraceControl 114
SCCPUDataRqst 61
SCCPUDATIND 155
SccpUSapCfg 102

SccpUSapStats 83
SccpUSapStatus 85
service access points (SAP) 11
service functions 13, 43, 44
service users 11
signaling parameters 19
signaling point 20
SS7 architecture 9
state changes 21
Subsystem multiplicity indicator parameter 132
Subsystem number parameter 132
subsystem status 20
Subsystem status indicator parameter 132

T

TCAP requests 26
tracing 42

U

unconfirmed operations 13
Unitdata request message 136
utilities 115