



TB640 SS7 user's guide

**Document number
9010-00030-20**

September 2008

Copyright © 2003 - 2008 by TelcoBridges inc.

All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from TelcoBridges inc.

This page in intentionally left blank

1 TABLE OF CONTENT

1	TABLE OF CONTENT.....	3
2	LIST OF FIGURES	8
3	LIST OF TABLES	10
4	MTP2.....	16
4.1	Overview.....	16
4.1.1	Summary.....	16
4.1.2	Features.....	16
4.1.3	Architecture.....	18
4.1.3.1	Priorities.....	20
4.1.3.2	Flow Control.....	20
4.1.4	Specification	20
4.2	MTP2 Configuration.....	21
4.2.1	Configuration of layer.....	21
4.2.1.1	General Configuration	21
4.2.1.2	Link Configuration.....	21
4.2.2	Compatibility	28
4.2.2.1	Variants.....	28
4.3	MTP2 Alarms.....	29
4.3.1	Link Alarms	29
4.4	MTP2 States.....	31
4.4.1	Link States Get.....	31
4.4.2	Link States Set	34
4.5	MTP2 Statistics.....	35
4.5.1	Link Statistics.....	35
4.6	MTP2 Standalone mode.....	37
4.6.1	Send Action.....	37
4.6.2	Send Data.....	38
4.6.3	Received Data.....	40
4.6.4	Retrieved Data	40
5	MTP3.....	42
5.1	Overview.....	42
5.1.1	Summary.....	42
5.1.2	Features.....	42
5.1.3	Architecture.....	44
5.1.3.1	Addressing	52
5.1.3.1.1	Signaling Points	52
5.1.3.2	Routing.....	52
5.1.3.3	Flow Control.....	53
5.1.3.4	Multiple OPC Capability	53
5.1.4	Specification	53
5.2	MTP3 Configuration.....	54

name.

5.2.1	Configuration of layer.....	54
5.2.1.1	General Configuration	55
5.2.1.2	Userpart Configuration	58
5.2.1.3	Linkset Configuration.....	60
5.2.1.4	Link Configuration.....	61
5.2.1.5	Route Configuration.....	67
5.2.2	Compatibility	71
5.2.2.1	Variants.....	71
5.2.2.2	Timers	72
5.3	MTP3 Alarms.....	74
5.3.1	Link Alarms	74
5.3.2	Linkset Alarms.....	76
5.3.3	Route Alarms	76
5.4	MTP3 States.....	77
5.4.1	Link States Get.....	78
5.4.2	Link States Set	79
5.4.3	Route Linkset States Get.....	80
5.4.4	Route Linkset States Set	81
5.4.5	Route States Get.....	82
5.5	MTP3 Statistics.....	83
5.5.1	General Statistics	83
5.5.2	Link Statistics.....	84
5.5.3	Route Linkset Statistics.....	90
5.5.4	Route Statistics.....	91
5.6	MTP3 Standalone mode.....	93
5.6.1	Send Data.....	93
5.6.2	Received Data	95
6	ISUP	97
6.1	Overview.....	97
6.1.1	ISUP summary	97
6.1.2	Architecture within the TB640	101
6.1.3	Configuration sequence	104
6.1.4	Features of the TB640 ISUP layer	106
6.1.5	Specifications.....	107
6.2	ISUP Configuration	108
6.2.1	Layout of the ISUP network	108
6.2.2	Configuration of layer.....	109
6.2.2.1	General.....	110
6.2.2.2	Network.....	112
6.2.2.3	Userpart.....	113
6.2.2.4	Interface	118
6.2.2.5	Circuits.....	121
6.2.3	Compatibility	126
6.2.3.1	Variants.....	126
6.3	ISUP Signaling.....	126

- 6.3.1 Host application requirements 126
 - 6.3.1.1 General guidelines for designing an efficient call control application using ISUP
127
 - 6.3.1.2 General guidelines for designing highly redundant application (blade failure) . 128
 - 6.3.1.3 General guidelines for designing a high-performance system..... 128
 - 6.3.1.4 General guidelines to have highly redundant high-performance system (host
failure) 130
- 6.3.2 Information element usage..... 131
 - 6.3.2.1 IE Usage..... 131
 - 6.3.2.2 IE inclusion in different messages 132
 - 6.3.2.3 Reading and writing IEs..... 134
 - 6.3.2.4 Most common IEs ITU format definitions..... 139
 - 6.3.2.4.1 Called party number IE 139
 - 6.3.2.4.2 Calling party number IE..... 139
 - 6.3.2.4.3 Calling party’s category IE 140
 - 6.3.2.4.4 Forward call indicator IE 140
 - 6.3.2.4.5 Nature of connection indicators IE 140
 - 6.3.2.4.6 Backward call indicators IE 141
 - 6.3.2.4.7 Subsequent number IE 141
 - 6.3.2.4.8 Cause indicators IE 141
 - 6.3.2.4.9 Suspend/resume indicators IE..... 142
 - 6.3.2.4.10 Range and status IE..... 142
- 6.3.3 ISUP functionalities and behaviors..... 144
 - 6.3.3.1 Blocking and unblocking 144
 - 6.3.3.2 Circuit group functions 145
 - 6.3.3.3 Flow control 145
 - 6.3.3.4 Circuit continuity testing..... 146
 - 6.3.3.4.1 Continuity testing within a call 146
 - 6.3.3.4.2 Continuity testing outside a call..... 147
- 6.3.4 Call flow and scenarios..... 147
 - 6.3.4.1 Successful basic call setup (no ACM) 149
 - 6.3.4.2 Successful basic call setup (with ACM) 149
 - 6.3.4.3 Successful call setup (with CPG)..... 150
 - 6.3.4.4 Successful call setup (overlap)..... 150
 - 6.3.4.5 Successful call release..... 150
 - 6.3.4.6 Outgoing call refused by incoming side 151
 - 6.3.4.7 Call collision because racing condition host↔TB640 151
 - 6.3.4.8 Call collision between two SS7 nodes 151
 - 6.3.4.9 Call with successful continuity testing 152
 - 6.3.4.10 Call with failed continuity testing and successful re-check..... 153
 - 6.3.4.11 Call with failed continuity testing and no re-check 154
 - 6.3.4.12 Call with failed continuity testing and failed re-check 154
 - 6.3.4.13 Call with failed continuity testing and failed re-check (timeout) 155
 - 6.3.4.14 Successful continuity testing out-of-call..... 155
 - 6.3.4.15 Failed continuity testing out-of-call with successful re-check 156
 - 6.3.4.16 Failed continuity testing out-of-call with failed re-check..... 156

6.3.4.17	Circuit group reset.....	157
6.3.4.18	Circuit blocking	157
6.3.4.19	Circuit group blocking.....	157
6.3.4.20	Circuit unblocking	158
6.3.4.21	Circuit group unblocking	158
7	SCCP	159
7.1	Overview.....	159
7.1.1	Summary	159
7.1.2	Features	160
7.1.3	Architecture.....	160
7.1.3.1	Connectionless Service	163
7.1.3.2	Management.....	163
7.1.3.3	Addressing	163
7.1.3.3.1	Global Title Translation Association	165
7.1.3.4	Routing.....	166
7.1.3.5	Traffic Limitation and Congestion Control	167
7.1.3.6	Interworking.....	167
7.1.3.7	Messages.....	167
7.1.4	Specification	167
7.2	SCCP Configuration	169
7.2.1	Configuration of layer.....	169
7.2.1.1	General Configuration	170
7.2.1.2	Network Configuration	172
7.2.1.3	Lsap Configuration	174
7.2.1.4	Route Configuration.....	175
7.2.1.5	Userpart Configuration	181
7.2.1.6	GTT Association Configuration	182
7.2.1.7	GTT Address Map Configuration.....	188
7.2.2	Compatibility	191
7.2.2.1	Variants.....	191
7.3	SCCP Alarms	192
7.3.1	Alarm	192
7.3.2	Lsap Alarm.....	193
7.3.3	Userpart Alarm.....	194
7.3.4	Error Performance Report.....	195
7.4	SCCP States	197
7.4.1	General States Get.....	197
7.4.2	General States Set	197
7.4.3	Route States Get.....	198
7.5	SCCP Statistics	200
7.5.1	General Statistics	200
7.5.2	Userpart Statistics	204
7.5.3	Lsap Statistics	205
7.5.4	Route Statistics.....	207
7.6	SCCP Standalone mode	207

7.6.1	Data Request	208
7.6.2	Coordinate Request	212
7.6.3	Coordinate Response	213
7.6.4	Point Code SSN Status Request	213
7.6.5	SSN State Change Request	214
7.6.6	Data Received	215
7.6.7	Data Error Indication	216
7.6.8	Coordinate Indication	217
7.6.9	Coordinate Confirmation	218
7.6.10	Point Code SSN Status Confirmation	219
7.6.11	SSN State Change Indication	220
7.6.12	SSN State Change Confirmation	221
7.6.13	PC State Change Indication	221
8	TCAP	223
8.1	Overview	223
8.1.1	Summary	223
8.1.2	Features	224
8.1.3	Architecture	224
8.1.3.1	Message	227
8.1.3.1.1	Transaction portion	227
8.1.3.1.2	Dialogue portion	227
8.1.3.1.3	Component portion	227
8.1.4	Specification	228
8.2	TCAP Configuration	228
8.2.1	Configuration of layer	228
8.2.1.1	General configuration	229
8.2.1.2	Userpart configuration	230
8.2.2	Compatibility	231
8.2.2.1	Variants	231
8.3	TCAP Transaction	231
8.3.1	Creating new transaction	232
8.3.2	Adding component portion	233
8.3.3	Sending transaction and dialogue portions	233
8.3.4	Receiving component portion	234
8.3.5	Receiving transaction and dialogue portions	234
8.3.6	Receiving transaction error indication	235
8.3.7	Transaction flow and scenarios	236
8.3.7.1	Successful unstructured dialogue (UNIDIRECTIONAL)	236
8.3.7.2	Successful structured dialogue initiation	237
8.3.7.3	Successful structured dialogue acceptance	238
8.3.7.4	Successful structured dialogue continuation	239
8.3.7.5	Successful structured dialogue termination	242
8.3.7.6	Successful structured dialogue abort	243
8.4	TCAP Alarms	245
8.4.1	Alarm	245
8.4.2	Userpart alarm	246

8.5	TCAP States.....	247
8.5.1	Userpart states get.....	247
8.6	TCAP Statistics.....	247
8.6.1	Userpart statistics.....	248
9	REVISION HISTORY	257
9.1	Changed in revision 9010-00030-1U.....	257
9.2	Changed in revision 9010-00030-1V.....	257
9.3	Changed in revision 9010-00030-1W.....	257
9.4	Changed in revision 9010-00030-1X.....	257
9.5	Changed in revision 9010-00030-1Y.....	257
9.6	Changed in revision 9010-00030-1Z.....	257

2 LIST OF FIGURES

Figure 1 - MTP2 OSI model.....	16
Figure 2 - MTP2 layer organization within TB640.....	18
Figure 3 - MTP2 layer hierarchy.....	19
Figure 4 - MTP3 OSI model.....	42
Figure 5 - MTP3 layer organization within TB640.....	45
Figure 6 - MTP3 layer hierarchy.....	46
Figure 7- MTP3 relationship between links, linksets and routes.....	47
Figure 8 - ISUP OSI model.....	98
Figure 9 - Example of an SS7 network.....	100
Figure 10 - ISUP layer organization within TB640.....	102
Figure 11 - ISUP layer hierarchy.....	105
Figure 12 - Circuit setup between multiple SS7 nodes.....	109
Figure 13 - Host application typical architecture.....	127
Figure 14 - Multi-host high-performance system.....	129
Figure 15 - HA introduced in large system.....	130
Figure 16 - IE encoding.....	136
Figure 17 - Circuit states.....	144
Figure 18 - SCCP OSI model.....	159
Figure 19 - SCCP layer organization within TB640.....	161
Figure 20 - SCCP layer hierarchy.....	162
Figure 21 - SCCP Addressing (ITU).....	164
Figure 22 - SCCP Addressing (ANSI).....	164
Figure 23 - TCAP OSI model.....	223
Figure 24 - TCAP layer organization within TB640.....	225
Figure 25 - TCAP layer hierarchy.....	226
Figure 26 - TCAP successful unstructured dialogue (UNIDIRECTIONAL).....	237
Figure 27 - TCAP successful structured dialogue initiation.....	238
Figure 28 - TCAP successful structured dialogue acceptance.....	239
Figure 29 - TCAP successful structured dialogue continuation.....	241
Figure 30 - TCAP successful structured dialogue termination.....	243

Figure 31 - TCAP successful structured dialogue abort 245

3 LIST OF TABLES

Table 1 - MTP2 connection mode	23
Table 2 - MTP2 protocol types	23
Table 3 - DPC Length	23
Table 4 - MTP2 timeslot rates	24
Table 5 - MTP2 error correction types	26
Table 6 - Protocols variants compatibility	29
Table 7 - MTP2 link alarms	30
Table 8 - MTP2 supplemental alarm information	31
Table 9 - MTP2 local congestion state	32
Table 10 - MTP2 data link state	32
Table 11 - MTP2 alarm generation state	32
Table 12 - MTP2 internal state	33
Table 13 - MTP2 action	38
Table 14 - MTP3 traffic distribution scenario #1	48
Table 15 - MTP3 traffic distribution scenario #2	49
Table 16 - MTP3 traffic distribution scenario #3	50
Table 17 - MTP3 traffic distribution scenario #4	51
Table 18 - MTP3 signaling point types	56
Table 19 - MTP3 restart procedures	57
Table 20 - MTP3 connection mode	58
Table 21 - SSF values	59
Table 22 - MTP3 link types	59
Table 23 - MTP3 link priorities	63
Table 24 - MTP3 message priorities	63
Table 25 - MTP3 directions	68
Table 26 - MTP3 SLS ranges	69
Table 27 - MTP3 SLS selectors	70
Table 28 - MTP3 timers	72
Table 29 - MTP3 link alarms	74
Table 30 - MTP3 Link alarm cause values	75
Table 31 - MTP3 Linkset alarms	76
Table 32 - MTP3 alarms	77
Table 33 - MTP3 link states	78
Table 34 - MTP3 linkset status	81
Table 35 - MTP3 linkset status for NTT	81
Table 36 - MTP3 SLS ranges value	94
Table 37 - ISUP protocol variants	114
Table 38 - ISUP SSF values	114
Table 39 - ISUP calling address indicator values	115
Table 40 - ISUP numbering plan values	115
Table 41 - ISUP release locations	116
Table 42 - ISUP trunk types	119

Table 43 - ISUP pause actions	119
Table 44 - ISUP multi-rate check values	120
Table 45 - ISUP SLS selector values	120
Table 46 - ISUP SLS range values	120
Table 47 - ISUP circuit control types	123
Table 48 - ISUP slot ID format	123
Table 49 - ISUP circuit options	124
Table 50 - ISUP mandatory/optional IEs (ITU)	133
Table 51 - ISUP mandatory/optional IEs (ANSI)	134
Table 52 - Called party number IE format	139
Table 53 - Calling party number IE format	140
Table 54 - Calling party's category IE format	140
Table 55 - Forward call indicator IE format	140
Table 56 - Nature of connection indicators IE format	141
Table 57 - Backward call indicators IE format	141
Table 58 - Subsequent number IE format	141
Table 59 - Cause indicators IE format	142
Table 60 - Suspend/resume indicators IE format	142
Table 61 - Range and status IE format	142
Table 62 - ISUP flow control events	145
Table 63 - ISUP successful basic call setup (no ACM)	149
Table 64 - ISUP Successful basic call setup (with ACM)	149
Table 65 - ISUP successful call setup (with CPG)	150
Table 66 - ISUP successful call setup (overlap)	150
Table 67 - ISUP successful call release	150
Table 68 - ISUP outgoing call refused by incoming side	151
Table 69 - ISUP call collision (race condition)	151
Table 70 - ISUP call collision between two SS7 nodes	151
Table 71 - ISUP call with successful continuity testing	152
Table 72 - ISUP call with failed continuity testing and successful re-check	153
Table 73 - ISUP call with failed continuity testing and no re-check	154
Table 74 - ISUP call with failed continuity testing and failed re-check	154
Table 75 - ISUP call with failed continuity testing and failed re-check (timeout)	155
Table 76 - ISUP successful continuity testing out-of-call	155
Table 77 - ISUP failed continuity testing out-of-call with successful re-check	156
Table 78 - ISUP failed continuity testing out-of-call with failed re-check	156
Table 79 - ISUP circuit reset	157
Table 80 - ISUP circuit group reset	157
Table 81 - ISUP circuit blocking	157
Table 82 - ISUP circuit group blocking	157
Table 83 - ISUP circuit unblocking	158
Table 84 - ISUP circuit group unblocking	158
Table 85 - SCCP GTT association for ITU	165
Table 86 - SCCP Traffic Limitation	171
Table 87 - SCCP protocol variant	173
Table 88 - SCCP network indicator bit	173

Table 89 - SCCP route protocol variant.....	176
Table 90 - SCCP replicate mode.....	176
Table 91 - SCCP route options	178
Table 92 - SCCP route SLS mask.....	178
Table 93 - SCCP SSN value to identify SCCP user	179
Table 94 - SCCP SSN options	180
Table 95 - SCCP message priority.....	181
Table 96 - SCCP connection mode.....	182
Table 97 - SCCP GT format type	183
Table 98 - SCCP GT action type	184
Table 99 - SCCP GT odd/even type	185
Table 100 - SCCP GT nature of address indicator	186
Table 101 - SCCP GT numbering plan type.....	187
Table 102 - SCCP GT encoding scheme type	187
Table 103 - SCCP routing type.....	191
Table 104 - SCCP alarm category	192
Table 105 - SCCP event and cause alarm under CATEGORY_PROTOCOL.....	192
Table 106 - SCCP event and cause LSAP alarm under CATEGORY_PROTOCOL	193
Table 107 - SCCP event and cause LSAP alarm under CATEGORY_INTERFACE	193
Table 108 - SCCP event and cause USERPART alarm under CATEGORY_PROTOCOL	194
Table 109 - SCCP event and cause USERPART alarm under CATEGORY_INTERFACE.....	194
Table 110 - SCCP cause under error performance report.....	195
Table 111 – SCCP point code status.....	199
Table 112 – SCCP remote status	199
Table 113 – SCCP SMI status	199
Table 114 – SCCP SSN status	200
Table 115 - SCCP protocol class	208
Table 116 - SCCP present flag.....	209
Table 117 - SCCP ISNI mark identification	210
Table 118 - SCCP ISNI routing indicator.....	210
Table 119 - SCCP ISNI type indicator	210
Table 120 - SCCP INS information type indicator.....	211
Table 121 - SCCP INS routing indicator	211
Table 122 - SCCP request status type.....	214
Table 123 - SCCP status type	215
Table 124 - SCCP return cause.....	216
Table 125 - SCCP SMI type	218
Table 126 - SCCP remote status type.....	219
Table 127 - TCAP protocol variant	230
Table 128 - TCAP alarm category	245
Table 129 - TCAP alarm event.....	245
Table 130 - TCAP alarm cause.....	246
Table 131 - TCAP userpart status.....	247

This page in intentionally left blank

The signaling information/code contained in this document/product is based on the best information we have available. Although it has been tested successfully with other piece of signaling equipment, we cannot guarantee that it will conform to the usage of any particular switch in the field.

Parts of this documentation are taken from Continuous Computing (Trillium) service definition documents for MTP2, MTP3 and ISUP according to license agreement with TelcoBridges inc.

Copyright © 1989-2001 by Trillium Digital Systems inc., All rights reserved.

This page in intentionally left blank

4 MTP2

4.1 Overview

This section gives a description of the TB640 Message Transfer Part (MTP) level 2 layer architecture and usage. This layer is referred to as MTP2 in the rest of the section.

4.1.1 Summary

The MTP2 layer ensures the transmission of data from one node to the next. Message sequence validation, error checking and flow control are implemented in MTP2. MTP2 provides the data link layer functionality within the OSI mode.

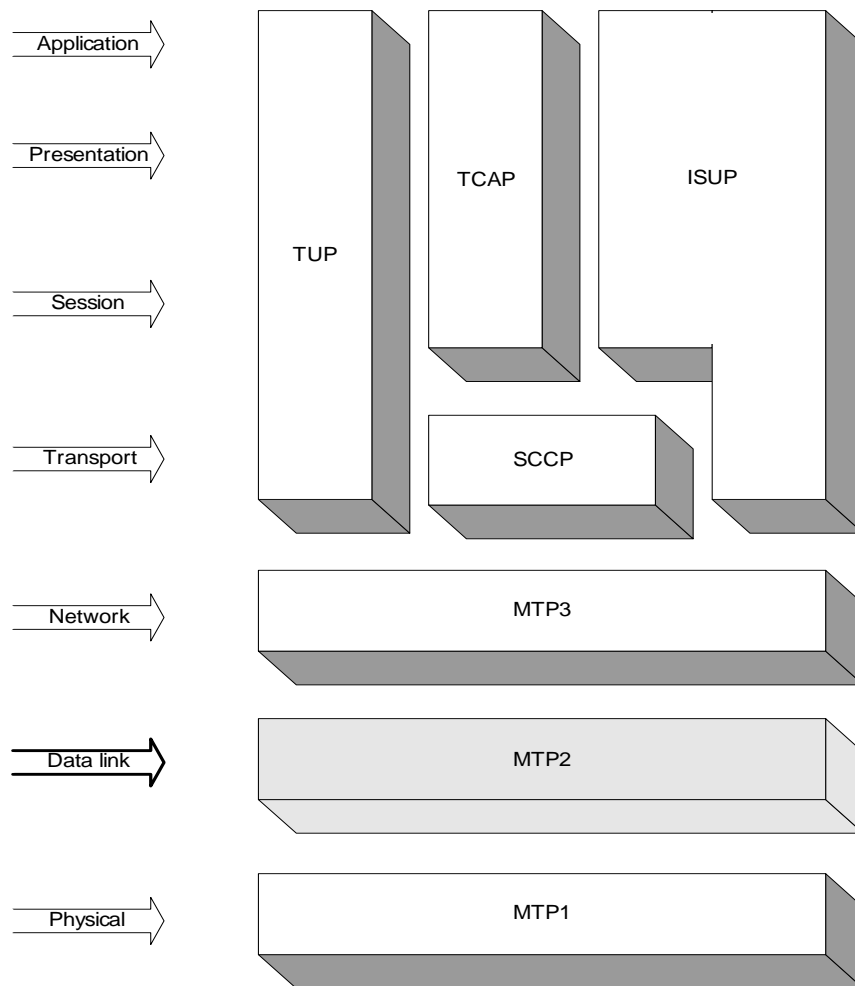


Figure 1 - MTP2 OSI model

4.1.2 Features

MTP Level 2 supports 1988, 1992 CCITT Q.703, 1988, 1992 ANSI recommendation T1.111.3, TTC JT-Q.703, and NTT - Q.703 recommendations.

MTP Level 2 provides the following basic features:

- Establish and release data link connections.
- Transfer data.
- Normal alignment procedure (ITU and ANSI only).
- Emergency alignment procedure.
- Signal unit error rate monitor.
- Alignment error rate monitor.
- Basic error correction method.
- Preventive cyclic retransmission error correction method (ITU and ANSI only).
- Congestion control and congestion abatement.
- Inform MTP Level 3 on detection of error.
- Message retrieval for changeover.
- Processor outage procedure (ITU and ANSI only).
- Realignment on link failure.
- Multiple variants, including ITU 1988 and 1992, ANSI 1988 and 1992, TTC, and NTT.
- Regulates data flow when system's resource utilization reaches configurable thresholds.
- Prioritizing of user data messages (TTC and NTT only).
- Retransmission of LSSUs and FISUs (TTC and NTT only).
- Auto link realignment on link failure (TTC and NTT only).

MTP Level 2 does not directly provide the following features:

- Signal unit delimitation, alignment, and error detection.
- Retransmission of FISU and LSSUs (ITU and ANSI only).

MTP Level 2 assumes these features are provided by layer 1, but also expects errors to be reported by layer 1 so that it can take any appropriate action.

4.1.3 Architecture

MTP Level 2 defines the functions and procedures of the signaling system for providing reliable transfer of signaling messages over a signaling link. The MTP2 layer is a service provider to the MTP3 Link layer. The MTP2 layer is a service user of Trunk Controller (as shown on Figure 1).

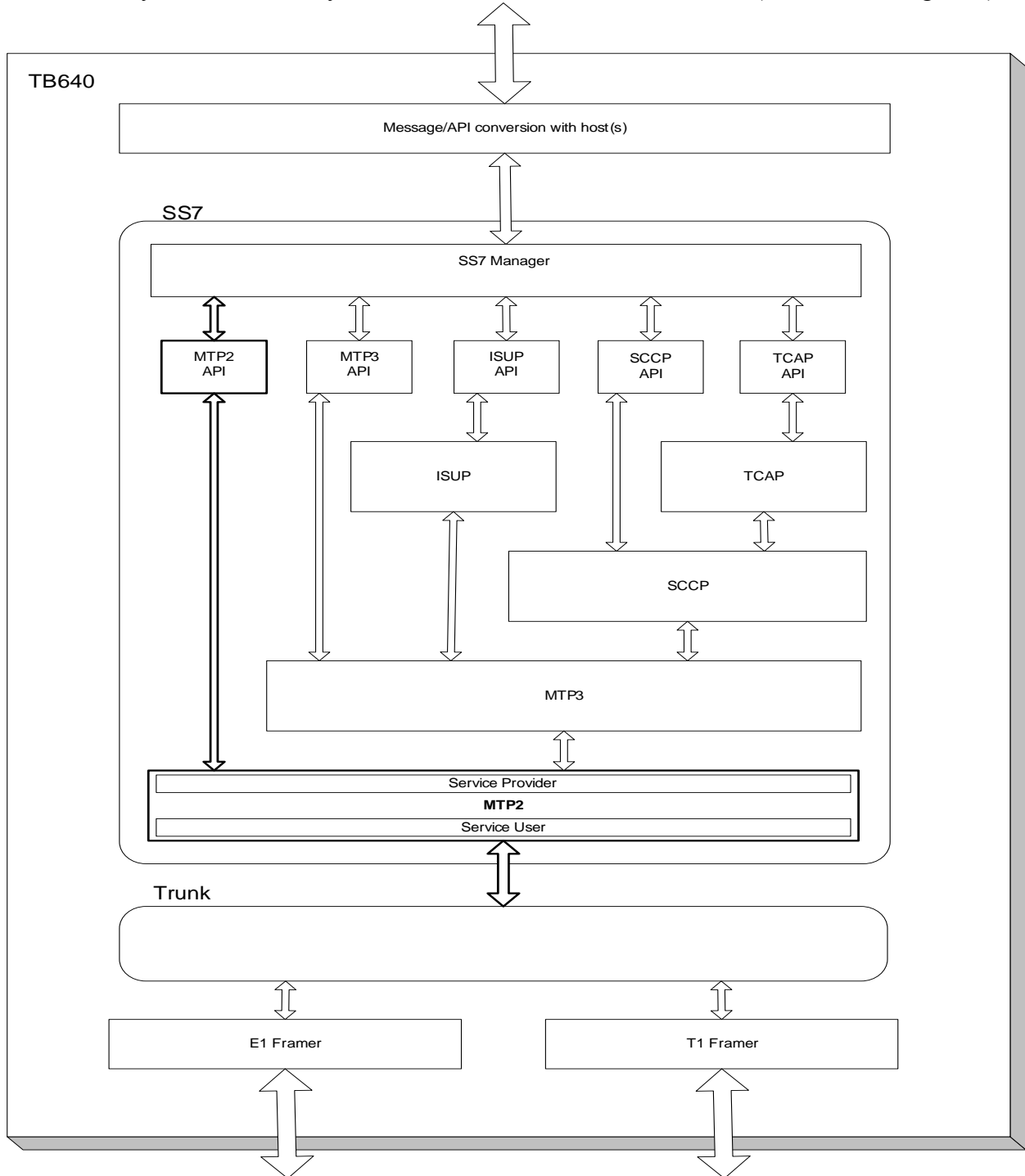


Figure 2 - MTP2 layer organization within TB640

MTP2 can be used in conjunction with an above local MTP3 layer and can be used as a standalone and communicates with remote MTP3 layer (host application). See the connection mode table in the MTP2 Configuration section.

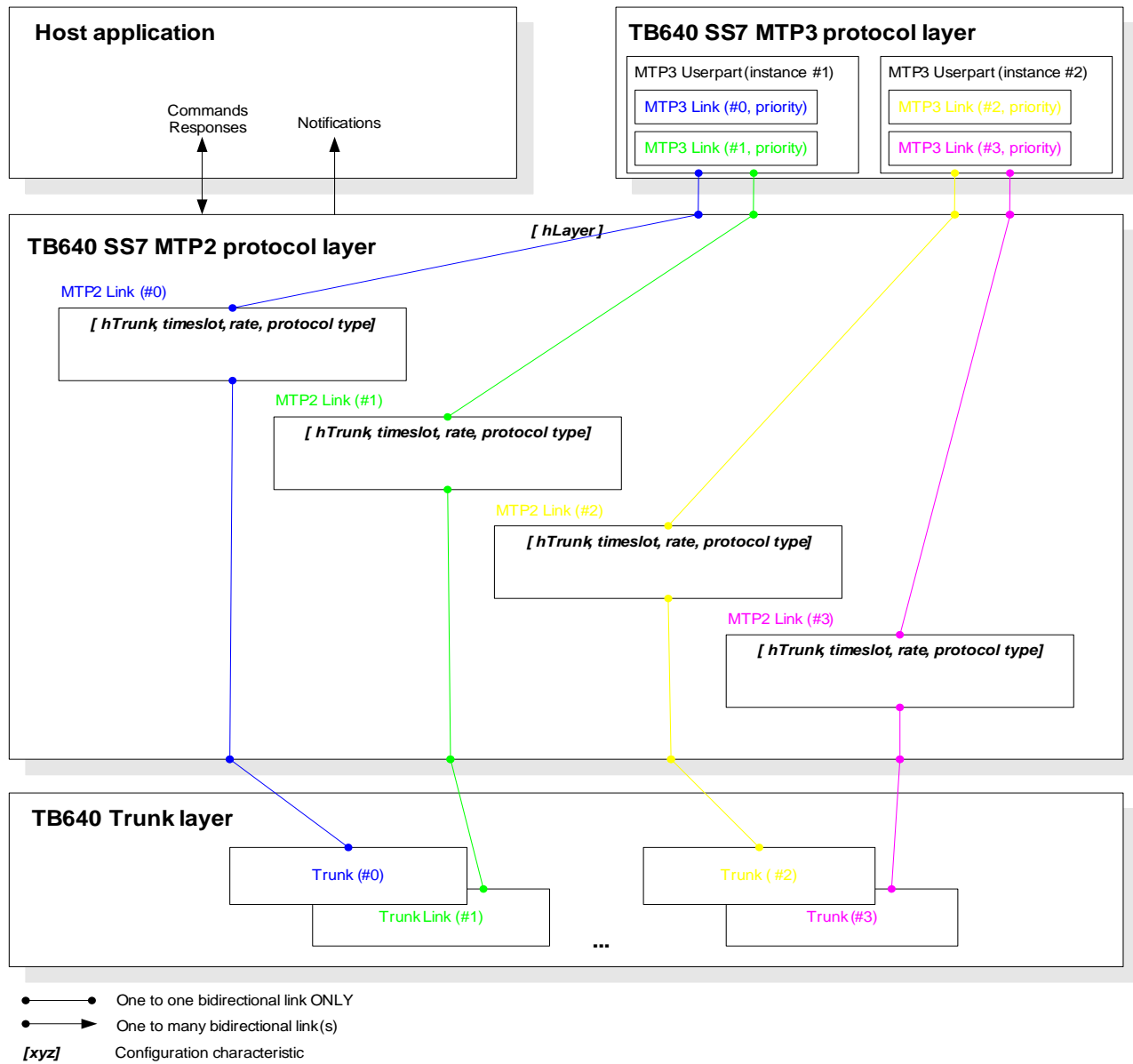


Figure 3 - MTP2 layer hierarchy

A MTP2 Link must be associated with one Trunk instance. It's a 1 to 1 association. MTP Level 2 defines the functions and procedures of the signaling system for providing reliable transfer of signaling messages over a signaling link.



It is important to ensure that the received and transmitted clock of the Trunk was synchronized. This can be useful to remove undesirable FIB and BSN error. See section "Clock Configurations" in the Tb640 user's guide.

4.1.3.1 Priorities

No pre-emption of data occurs in the process of being transmitted or requiring retransmission. Only a single congestion priority is supported for ANSI and ITU. For a TTC/NTT link, transmitted data buffers are prioritized based on the priority indicated by MTP Level 3.

4.1.3.2 Flow Control

The MTP Level 2 software regulates data flow when resource utilization (such as buffer pool size or queue length) reaches specifies thresholds. For each threshold, MTP Level 2 performs one of the following actions:

- Informs the layer manager about flow control.
- Sends a link status signal unit with status "B" (busy).
- Sends a flow control indication to the upper layer.

The MTP Level 2 software prevents deadlocks and overload conditions from occurring.

4.1.4 Specification

The MTP2 software conforms to the following standards:

- TI.111.3 SS7 Signaling Data Link (MTP Level 2), ANSI 1992.
- Q.701 SS7 (MTP), CCITT 1992.
- Q.703 SS7 Signaling Data Link, CCITT 1992.
- Q.781 MTP Level 2 Test Specification, CCITT.
- JT-Q.703 MTP Signaling Link, TTC recommendation.
- NTT - Q.703 MTP Specifications, NTT recommendation.

MTP2 can support different protocol variants:

- ITU (< *ITU BLUE* > used for ITU88)
- ITU92 (< *ITU WHITE* > used for ITU92, ITU97, Q767, SINGAPORE and CHINA)
- ANSI88 (used for ANSI88)
- ANSI92 (used for ANSI92, ANSI96 and TELCORDIA)
- TTC (*not available yet*)
- NTT (*not available yet*)

The following messages may be received or transmitted by MTP 2 across the Trunk controller:


- Message Signal Unit (MSU)
- Link Status Signal Unit (LSSU)
- Fill-In Signal Unit (FISU)


4.2 MTP2 Configuration


4.2.1 Configuration of layer

General guidelines for configuration of MTP2 for a proper operation include the following:

1. The MTP2 general allocation¹ (TB640_MSG_ID_SS7_MTP2_OP_ALLOC) must precede all other messages (other configuration MTP2 alloc, get, states and stats). The response of this message is a MTP2 handle.
2. The MTP2 Link allocation (TB640_MSG_ID_SS7_MTP2_OP_LINK_ALLOC) must be made (with the MTP2 handle from **step 1**) to connect with a Trunk. The response of this message is a MTP2 Link handle.

 For any reconfiguration with a MTP2 (TB640_MSG_ID_SS7_MTP2_OP_SET_PARAMS) message, all reconfigurable parameters must be filled appropriately even if the intention is to modify a single parameter.

 When a reconfiguration with a MTP2 set params is issued, the effect of the set params may become effective with a delay depending on the present protocol activity.

 For the configuration of many MTP2 Link (1 to maximum) you must repeated the **step 2**.

4.2.1.1 General Configuration

The TB640_MSG_ID_SS7_MTP2_OP_ALLOC (request/response) message is used to initialize the general parameters of the MTP2 layer.

The **request** part of the message TB640_MSG_ID_SS7_MTP2_OP_ALLOC contains the field:

```
...
TB640_SS7_HANDLE          hLayer; /* Contains the layer handle from system manager
                             module */
...
```

The **response** part of the message TB640_MSG_ID_SS7_MTP2_OP_ALLOC contains the field:

```
...
TB640_SS7_MTP2_HANDLE     hMtp2; /* The handle of the initialized MTP2 layer */
...
```

4.2.1.2 Link Configuration

The TB640_MSG_ID_SS7_MTP2_OP_LINK_ALLOC (request/response) message is used to initialize an instance of an MTP2 link.

¹ All fields of a configuration message alloc must be filled unless explicitly optional or not defined for certain variants.

The **request** part of the message TB640_MSG_ID_SS7_MTP2_OP_LINK_ALLOC contains these fields:

```
...
TB640_SS7_MTP2_HANDLE hMtp2;          /* Handle of the MTP2 layer
                                        (received from TB640_MSG_SS7_MTP2_OP_ALLOC) */
TB640_TRUNK_HANDLE    hTrunk;         /* Handle of the trunk to use as the layer 1 */
TB640_SS7_MTP2_CFG    Cfg;           /* Contains the configuration of the MTP2 link */
TBX_BOOL              afTimeslotIncluded [TB640_TRUNK_MAX_TIMESLOT]; /* Array of Boolean
                                                                    describing which timeslot
                                                                    to included in the link */
...
```

Structure contains the link configuration parameters for MTP2 layer:

```
typedef struct _TB640_SS7_MTP2_CFG
{
    TB640_SS7_UID                UidMtp2Link;

    /* General configuration */
    TB640_SS7_MTP2_CONNECTION_MODE ConnectionMode;
    TB640_SS7_MTP2_PROTOCOL_TYPE   ProtocolType;
    TB640_SS7_DPC_LENGTH           DpcLength;
    TB640_TRUNK_TIMESLOT_RATE      TimeslotRate;

    /* Protocol related configuration (common to all protocols) */
    TBX_UINT32                     un32MaxOutstandingFrames;
    TBX_UINT32                     un32T1Timer;
    TBX_UINT32                     un32T2Timer;
    TBX_UINT32                     un32T3Timer;
    TBX_UINT32                     un32T5Timer;
    TBX_UINT32                     un32T6Timer;
    TBX_UINT32                     un32T7Timer;
    TBX_UINT32                     un32ProvEmrgcy;
    TBX_UINT32                     un32MaxFrameLength;
    TBX_BOOL                       fDiscardFrameInCong;
    TBX_UINT32                     un32SigUnitErrRateThres;
    TBX_UINT32                     un32AligErrRateEmergThres;
    TBX_UINT32                     un32MaxAlignmentAttempt;
    TBX_UINT32                     un32FlowCtrlNbMsgStart;
    TBX_UINT32                     un32FlowCtrlNbMsgEnd;
    TBX_UINT8                      aun8Padding0 [4];


    /* Protocol specific configuration */
    union
    {
        TB640_SS7_MTP2_ITU_CFG      ItuCfg;        /* ITU config. */
        TB640_SS7_MTP2_ANSI_CFG     AnsiCfg;       /* ANSI config. */
        TB640_SS7_MTP2_NTT_CFG      NttCfg;       /* NTT config. */
        TB640_SS7_MTP2_TTC_CFG      TtcCfg;       /* TTC config. */
    };
} TB640_SS7_MTP2_CFG, *PTB640_SS7_MTP2_CFG;
```


General explanation of the parameters of link configuration:

- The unique MTP2 link identifier (*UidMtp2Link*) is used to connect a MTP3 link.
- The connection mode parameter (*ConnectionMode*) specifies the different modes of connection for a MTP2 link. This field is not reconfigurable. Allowable values:

Table 1 - MTP2 connection mode

Connection Mode	Description
TB640_SS7_MTP2_CONNECTION_MODE_STANDALONE	MTP2 layer is used as a standalone and communicates with host application.
TB640_SS7_MTP2_CONNECTION_MODE_MTP3	MPT2 layer is used in conjunction with an above MTP3 layer and cannot be controlled by the host application but the host application will receive ALARM notifications.
TB640_SS7_MTP2_CONNECTION_MODE_HSL_STANDALONE	MTP2 layer high speed link is used as a standalone and communicates with host application (ITU and ANSI).
TB640_SS7_MTP2_CONNECTION_MODE_HSL_MTP3	MPT2 layer high speed link is used in conjunction with an above MTP3 layer and cannot be controlled by the host application but the host application will receive ALARM notifications (ITU and ANSI).

 Do not forget to set correctly the array of Boolean describing which timeslot to included in the link (*afTimeslotIncluded*).

 Mode HSL uses timeslot 1 to 31 for E1 and 1 to 24 for T1/J1.

- The protocol type parameter (*ProtocolType*) specifies the protocol for a MTP2 link. This field is not reconfigurable. Allowable values:

Table 2 - MTP2 protocol types

Protocol Type	Description
TB640_SS7_MTP2_PROTOCOL_TYPE_ITU_88	(<i>ITU BLUE</i>) Used for ITU88.
TB640_SS7_MTP2_PROTOCOL_TYPE_ITU_92	(<i>ITU WHITE</i>) Used for ITU92, ITU97, Q767, SINGAPORE and CHINA.
TB640_SS7_MTP2_PROTOCOL_TYPE_ANSI_88	Used for ANSI88.
TB640_SS7_MTP2_PROTOCOL_TYPE_ANSI_92	Used for ANSI92, ANSI96 and TELCORDIA
TB640_SS7_MTP2_PROTOCOL_TYPE_TTC ²	Used for TTC.
TB640_SS7_MTP2_PROTOCOL_TYPE_NTT ²	Used for NTT.

- The dpc length parameter (*DpcLength*) is only used for tracing purpose. Allowable values:

Table 3 - DPC Length

Dpc Length	Description
TB640_SS7_DPC_LENGTH_14BITS	14bits format=3.8.3 default ITU configuration.
TB640_SS7_DPC_LENGTH_16BITS	16bits format=7.4.5 default TTC and NTT configuration.
TB640_SS7_DPC_LENGTH_24BITS	24bits format=8.8.8 default ANSI, CHINA and some ITU networks configuration.

- The timeslot rate parameter (*TimeslotRate*) indicates the rate of the timeslots included in the link. This field is not reconfigurable. Allowable values:

² Not available yet.

Table 4 - MTP2 timeslot rates

Timeslot Rate	Description
TB640_TRUNK_TIMESLOT_RATE_64KPBS	Timeslot signaling rate to 64 Kbps
TB640_TRUNK_TIMESLOT_RATE_56KPBS	Timeslot signaling rate to 56 Kbps


- The maximum outstanding frames parameter (*un32MaxOutstandingFrames*) indicates the maximum frames to be sent to MTP1 before receiving confirmation. This field is not reconfigurable.
- The common link timers configuration parameters. All timers are reconfigurable.

The timers have the following definitions:

- un32T1Timer*: Alignment ready timer (the same as TTC JT-Q.703 remote verification in progress timer). Typical value is 40000 milliseconds (40 seconds) for ITU and 13000 milliseconds (13 seconds) for ANSI.
- un32T2Timer*: Not aligned timer (the same as TTC JT-Q.703 wait for remote startup in progress timer). Typical value is 5000 milliseconds (5 seconds) for ITU and 11800 milliseconds (11.8 seconds) for ANSI.
- un32T3Timer*: Aligned timer (the same as TTC JT-Q.703 initialization in progress timer). Typical value is 1500 milliseconds (1.5 seconds) for ITU and 11800 milliseconds (11.8 seconds) for ANSI.
- un32T5Timer*: Sending SIB timer. Typical value is 100 milliseconds (0.1 second).
- un32T6Timer*: Remote congestion timer. Typical value is 3000 milliseconds (3 seconds).
- un32T7Timer*: Excessive delay of acknowledgment timer. Typical value is 1000 milliseconds (1 second).
- un32ProvEmrgcy*: Emergency proving period timer value (the same as TTC JT-Q.703 verification timer). Typical value is 500 milliseconds (0.5 seconds) for ITU and 600 milliseconds (0.6 seconds) for ANSI.

 For all timers, the value 0 is not permitted.

- The maximum frame length parameter (*un32MaxFrameLength*) specifies the maximum frame length for a message signal unit (MSU). This field is reconfigurable.

 MUST be set to 272 bytes.

- The discard frame in congestion flag (*fDiscardFrameInCong*) indicates if the discard frame is in congestion situation or not. Allowable values: TBX_TRUE or TBX_FALSE. This field is not reconfigurable.
- The signal unit error rate parameter threshold (*un32SigUnitErrRateThres*) has as its function the estimation of the signal unit error rate in order to decide about the signaling link fault condition. This field is reconfigurable.
- The alignment error rate emergency threshold parameter (*un32AligErrRateEmergThres*) is a linear counter which is operated during normal and emergency proving periods. This field is reconfigurable.

- The maximum alignment attempt parameter (*un32MaxAlignmentAttempt*) indicates the maximum number of attempts at alignment before alignment not possible is reached. This field is reconfigurable.
- The flow control message start number parameter (*un32FlowCtrlNbMsgStart*) is the number of message in queue threshold to start flow control. This field is reconfigurable.
- The flow control message end number parameter (*un32FlowCtrlNbMsgEnd*) is the number of message in queue threshold to stop flow control. This field is reconfigurable.

Structure contains the ITU and ANSI common link specific configuration parameters for MTP2 layer:

```

/* The following structure contains the configuration parameters common to ITU and
ANSI protocol links */
typedef struct _TB640_SS7_MTP2_ITU_ANSI_COMMON_CFG
{
    TBX_UINT32                un32StructVersion;
    TB640_SS7_MTP2_ITU_ANSI_ERROR_CORRECTION ErrorCorrectionType;
    TBX_UINT32                un32ProvNormal;
    TBX_UINT32                un32LssuLenght;
    TBX_UINT32                un32AligErrRateNormalThres;
    TBX_UINT32                un32MaxCyclMsgRetransMsg;
    TBX_UINT32                un32MaxCyclBytesRetransMsg;
    TBX_UINT8                aun8Padding0 [4];
} TB640_SS7_MTP2_ITU_ANSI_COMMON_CFG, *PTB640_SS7_MTP2_ITU_ANSI_COMMON_CFG;

/* The following structure contains the configuration parameters for ITU protocol
links */
typedef struct _TB640_SS7_MTP2_ITU_CFG
{
    TBX_UINT32                un32StructVersion;
    TBX_UINT8                aun8Padding0 [4];
    TB640_SS7_MTP2_ITU_ANSI_COMMON_CFG Common;
} TB640_SS7_MTP2_ITU_CFG, *PTB640_SS7_MTP2_ITU_CFG;

/* The following structure contains the configuration parameters for ANSI protocol
links */
typedef struct _TB640_SS7_MTP2_ANSI_CFG
{
    TBX_UINT32                un32StructVersion;
    TBX_UINT8                aun8Padding0 [4];
    TB640_SS7_MTP2_ITU_ANSI_COMMON_CFG Common;
} TB640_SS7_MTP2_ANSI_CFG, *PTB640_SS7_MTP2_ANSI_CFG;

```


General explanation of the parameters of ITU and ANSI common link specific configuration:

- The error correction type parameter (*ErrorCorrectionType*) specifies either normal or preventative cyclical. This field is not reconfigurable. Allowable values are:


Table 5 - MTP2 error correction types

Error Correction Type	Description
TB640_SS7_MTP2_ITU_ANSI_ERROR_CORRECTION_NORMAL	Normal error correction type.
TB640_SS7_MTP2_ITU_ANSI_ERROR_CORRECTION_CYCLICAL	Preventative cyclical error correction type.

- The proving normal timer parameter (*un32ProvNormal*) specifies the normal proving period timer value. Only used for ITU and ANSI link types. **Not** used for TTC and NTT link type. This field is reconfigurable. Typical value is 8200 milliseconds (8.2 seconds) for ITU and 2300 milliseconds (2.3 seconds) for ANSI.

 The value 0 is not permitted.

- The LSSU length parameter (*un32LssuLenght*) specifies the link status signal unit length. This field is reconfigurable.

 This value is always 1 for TTC and NTT link type.

- The alignment error rate normal threshold parameter (*un32AligErrRateNormalThres*) specifies the alignment error rate monitor threshold for normal condition. This field is reconfigurable.

- The maximum cyclic retransmit message parameter (*un32MaxCyclMsgRetransMsg*) specifies the maximum number of message units available for retransmission under cyclic error type. This field is not reconfigurable.

- The maximum cyclic retransmit bytes parameter (*un32MaxCyclBytesRetransMsg*) specifies the maximum number of bytes available for retransmission under cyclic error type. This field is not reconfigurable.

Structure contains the NTT link specific configuration parameters for MTP2 layer:

```

/* The following structure contains the configuration parameters for NTT protocol
links */
typedef struct _TB640_SS7_MTP2_NTT_CFG
{
    TBX_UINT32                un32StructVersion;
    TBX_UINT8                 aun8Padding0 [4];
    TB640_SS7_MTP2_TTC_NTT_COMMON_CFG Common;
    TBX_BOOL                  fNackIdenticalMsu;
    TBX_BOOL                  fIgnoreInvalidFib;
    TBX_BOOL                  fIgnoreInvalidBsn;
    TBX_BOOL                  fStopCongOnNack;
} TB640_SS7_MTP2_NTT_CFG, *PTB640_SS7_MTP2_NTT_CFG;

```

General explanation of the parameters of NTT link specific configuration:

- The nack identical MSU flag parameter (*fNackIdenticalMsu*). Option for generating negative acknowledgment if the same MSU (same FSN) is received again on the link. If

TBX_FALSE, MSU is ignored. If TBX_TRUE a negative acknowledgment is sent to the remote end. This field is reconfigurable.

- The ignore invalid FIB flag parameter (*fIgnoreInvalidFib*). Option for ignoring FISU/MSU with invalid FIB. If **TBX_TRUE**, the MSU/FISU is ignored. If **TBX_FALSE**, the link goes out of alignment and realignment is initiated. This field is reconfigurable.
- The ignore invalid BSN flag parameter (*fIgnoreInvalidBsn*). Option for ignoring FISU/MSU with invalid BSN. If **TBX_TRUE** the MSU/FISU is ignored. If **TBX_FALSE** link goes out of alignment and realignment is initiated. This field is reconfigurable.
- The stop congestion on nack flag parameter (*fStopCongOnNack*). Option for abating the remote congestion (stop T6 timer) on receipt of negative acknowledgment from the remote end. If **TBX_TRUE**, the congestion is abated on receipt of negative acknowledgment. If **TBX_FALSE**, congestion is not abated. This field is reconfigurable.

Structure contains the TTC link specific configuration parameters for MTP2 layer:

```
/* The following structure contains the configuration parameters for TTC protocol links */
typedef struct _TB640_SS7_MTP2_TTC_CFG
{
    TBX_UINT32                un32StructVersion;
    TBX_UINT8                aun8Padding0 [4];
    TB640_SS7_MTP2_TTC_NTT_COMMON_CFG    Common;
} TB640_SS7_MTP2_TTC_CFG, *PTB640_SS7_MTP2_TTC_CFG;
```

Structure contains the TTC and NTT common link specific configuration parameters for MTP2 layer:

```
/* The following structure contains the configuration parameters for TTC and NTT
protocol links */
typedef struct _TB640_SS7_MTP2_TTC_NTT_COMMON_CFG
{
    TBX_UINT32                un32StructVersion;
    TBX_UINT32                un32MaxOutstandingMsg;
    TBX_UINT32                un32FisuPeriodTimer;
    TBX_UINT32                un32FisuPeriodVerifTimer;
    TBX_UINT32                un32SioPeriodTimer;
    TBX_UINT32                un32SiePeriodTimer;
    TBX_UINT32                un32SiosPeriodTimer;
    TBX_UINT32                un32SiosTxTimer;
    TBX_UINT32                un32ErrorRateMonitoringTimer;
    TBX_UINT8                aun8Padding0 [4];
} TB640_SS7_MTP2_TTC_NTT_COMMON_CFG, *PTB640_SS7_MTP2_TTC_NTT_COMMON_CFG;
```

General explanation of the parameters of TTC and NTT common link specific configuration:

- The maximum outstanding message parameter (*un32MaxOutstandingMsg*) specifies maximum number of outstanding messages. This field is not reconfigurable.

- The common link timers configuration parameters. All timers are reconfigurable.

The timers have the following definitions:

<i>un32FisuPeriodTimer:</i>	Interval between FISU transmission when link is aligned. Typical value is 40 milliseconds.
<i>un32FisuPeriodVerifTimer:</i>	Interval between FISU transmission while remote verification is in progress. Typical value is 160 milliseconds.
<i>un32SioPeriodTimer:</i>	Interval between SIO transmission. Typical value is 40 milliseconds.
<i>un32SiePeriodTimer:</i>	Interval between SIE transmission. Typical value is 40 milliseconds.
<i>un32SiosPeriodTimer:</i>	Interval between SIOS transmission. Typical value is 40 milliseconds.
<i>un32SiosTxTimer:</i>	SIOS transmission time (when out of service). Default value is 0 milliseconds (not used).
<i>un32ErrorRateMonitoringTimer:</i>	SU normalization time. Default value is 0 milliseconds (not used).

The **response** part of the message TB640_MSG_ID_SS7_MTP2_OP_LINK_ALLOC contains the field:

```
...
TB640_SS7_MTP2_LINK_HANDLE    hMtp2Link;    /* The handle of the newly created link */
...
```

4.2.2 Compatibility

4.2.2.1 Variants

MTP2:

- ITU 88
- ITU 92
- ANSI 88
- ANSI 92

MTP3:

- ITU = (88, 92, 97, Q767 and SINGAPORE)
- CHINA
- ANSI = (88 and 92)
- ANSI 96 = (96 and TELCORDIA)

ISUP:

- ITU = (88 and 92)
- ITU 97
- SINGAPORE
- ANSI 88
- ANSI 92
- ANSI 95
- TELCORDIA 97
- CHINA
- UK
- ETSI
- ETSIV3

Protocols variants compatibility:

Table 6 - Protocols variants compatibility

MTP2	MTP3	ISUP
ITU 88	ITU	ITU
ITU 88	ITU	ITU 97
ITU 88	ITU	SINGAPORE
ITU 88	ITU	Q767
ITU 88	ITU	UK
ITU 88	ITU	ETSI
ITU 88	ITU	ETSIV3
ITU 88	CHINA	CHINA
ITU 92	ITU	ITU
ITU 92	ITU	ITU 97
ITU 92	ITU	SINGAPORE
ITU 92	ITU	Q767
ITU 92	ITU	UK
ITU 92	ITU	ETSI
ITU 92	ITU	ETSIV3
ITU 92	CHINA	CHINA
ANSI 88	ANSI	ANSI 88
ANSI 88	ANSI	ANSI 92
ANSI 88	ANSI	ANSI 95
ANSI 88	ANSI	TELCORDIA
ANSI 88	ANSI 96	ANSI 88
ANSI 88	ANSI 96	ANSI 92
ANSI 88	ANSI 96	ANSI 95
ANSI 88	ANSI 96	TELCORDIA
ANSI 92	ANSI	ANSI 88
ANSI 92	ANSI	ANSI 92
ANSI 92	ANSI	ANSI 95
ANSI 92	ANSI	TELCORDIA
ANSI 92	ANSI 96	ANSI 88
ANSI 92	ANSI 96	ANSI 92
ANSI 92	ANSI 96	ANSI 95
ANSI 92	ANSI 96	TELCORDIA

4.3 MTP2 Alarms

Alarms may indicate abnormal changes in status or be advisory in nature. Alarms are sent to the host whenever a condition possibly requiring attention is detected.

4.3.1 Link Alarms

The TB640_MSG_ID_SS7_MTP2_NOTIF_ALARM (event) notification message is received by the host application when a MTP2 link is reporting an error.

Structure contains the **event** notification link alarm for a MTP32link:

```
typedef struct _TB640_EVT_SS7_MTP2_NOTIF_ALARM
{
    TBX_MSG_HEADER                Header ;
```

```

TBX_UINT32                               un32MsgVersion;
TB640_SS7_MTP2_LINK_HANDLE               hMtp2Link;
TB640_SS7_MTP2_PROTOCOL_ALARM           Alarm;
TBX_UINT32                               un32SupplementalInfo;

} TB640_EVT_SS7_MTP2_NOTIF_ALARM, *PTB640_EVT_SS7_MTP2_NOTIF_ALARM;

```

General explanation of the field of event notification link alarm:

- The link handle field (*hMtp2Link*) specifies the handle of the MTP2 link to which the alarm corresponds.
- The alarm field (*Alarm*), indicates an alarm that can be reported at the link level 2. Possible values:

Table 7 - MTP2 link alarms

Link Alarm	Description
TB640_SS7_MTP2_PROTOCOL_ALARM_ENTERING_CONGESTION	Link entered congestion due to: management initiated or unknown reason.
TB640_SS7_MTP2_PROTOCOL_ALARM_EXITING_CONGESTION	Link leaving congestion due to: management initiated or unknown reason.
TB640_SS7_MTP2_PROTOCOL_ALARM_LINK_UP	Link is up at physical level.
TB640_SS7_MTP2_PROTOCOL_ALARM_LINK_DOWN	Link is down at MAC layer: due to management initiated or unknown reason.
TB640_SS7_MTP2_PROTOCOL_ALARM_LINK_ENABLED	Bind and enable link.
TB640_SS7_MTP2_PROTOCOL_ALARM_LINK_DISABLED	Unbind and disable link towards MAC interface.
TB640_SS7_MTP2_PROTOCOL_ALARM_PROTOCOL_ERROR	Protocol error on link due to: abnormal BSN, abnormal FIB, or congestion discard.
TB640_SS7_MTP2_PROTOCOL_ALARM_LINK_ALIGNED	Link aligned so cause is irrelevant.
TB640_SS7_MTP2_PROTOCOL_ALARM_LINK_ALIGNMENT_LOST	Link alignment lost for unknown reason.
TB640_SS7_MTP2_PROTOCOL_ALARM_NACK_RECEIVED	Negative acknowledgment received on the link from remote end.
TB640_SS7_MTP2_PROTOCOL_ALARM_DATA_REQUEST_ACK_LATE	Data request acknowledgment received in more than 0.5 seconds from remote end.
TB640_SS7_MTP2_PROTOCOL_ALARM_REMOTE_ENTERING_CONGESTION	Remote end congestion (SIB received) started for unknown reason.
TB640_SS7_MTP2_PROTOCOL_ALARM_REMOTE_EXITING_CONGESTION	Remote end congestion stopped.
TB640_SS7_MTP2_PROTOCOL_ALARM_REMOTE_PROCESSOR_OUTAGE_ON	Remote processor up (remote processor recovered) (ITU and ANSI).
TB640_SS7_MTP2_PROTOCOL_ALARM_REMOTE_PROCESSOR_OUTAGE_OFF	Remote processor down (remote processor outage) (ITU and ANSI).
TB640_SS7_MTP2_PROTOCOL_ALARM_LOCAL_PROCESSOR_OUTAGE_ON	Local processor up.
TB640_SS7_MTP2_PROTOCOL_ALARM_LOCAL_PROCESSOR_OUTAGE_OFF	Local processor down.
TB640_SS7_MTP2_PROTOCOL_ALARM_FLOW_CONTROL_ON	Start flow control.
TB640_SS7_MTP2_PROTOCOL_ALARM_FLOW_CONTROL_OFF	Stop flow control

- The supplemental information field (*un32SupplementalInfo*). Supplemental code (value) depending on the type of *alarm*.

Table 8 - MTP2 supplemental alarm information

Link Alarm	Supplemental Info
TB640_SS7_MTP2_PROTOCOL_ALARM_ENTERING_CONGESTION	0 = Unknown cause 8 = Management initiated
TB640_SS7_MTP2_PROTOCOL_ALARM_EXITING_CONGESTION	0 = Unknown cause 8 = Management initiated
TB640_SS7_MTP2_PROTOCOL_ALARM_LINK_UP	0 = Unknown cause
TB640_SS7_MTP2_PROTOCOL_ALARM_LINK_DOWN	0 = Unknown cause 8 = Management initiated
TB640_SS7_MTP2_PROTOCOL_ALARM_LINK_ENABLED	0 = Unknown cause
TB640_SS7_MTP2_PROTOCOL_ALARM_LINK_DISABLED	0 = Unknown cause
TB640_SS7_MTP2_PROTOCOL_ALARM_PROTOCOL_ERROR	256 = SU with abnormal backward sequence num 257 = SU with abnormal forward indicator bit 258 = congestion discard
TB640_SS7_MTP2_PROTOCOL_ALARM_LINK_ALIGNED	0 = Unknown cause
TB640_SS7_MTP2_PROTOCOL_ALARM_LINK_ALIGNMENT_LOST	0 = Unknown cause
TB640_SS7_MTP2_PROTOCOL_ALARM_NACK_RECEIVED	0 = Unknown cause
TB640_SS7_MTP2_PROTOCOL_ALARM_DATA_REQUEST_ACK_LATE	0 = Unknown cause
TB640_SS7_MTP2_PROTOCOL_ALARM_REMOTE_ENTERING_CONGESTION	0 = Unknown cause
TB640_SS7_MTP2_PROTOCOL_ALARM_REMOTE_EXITING_CONGESTION	0 = Unknown cause
TB640_SS7_MTP2_PROTOCOL_ALARM_REMOTE_PROCESSOR_OUTAGE_ON	0 = Unknown cause
TB640_SS7_MTP2_PROTOCOL_ALARM_REMOTE_PROCESSOR_OUTAGE_OFF	0 = Unknown cause
TB640_SS7_MTP2_PROTOCOL_ALARM_LOCAL_PROCESSOR_OUTAGE_ON	0 = Unknown cause
TB640_SS7_MTP2_PROTOCOL_ALARM_LOCAL_PROCESSOR_OUTAGE_OFF	0 = Unknown cause
TB640_SS7_MTP2_PROTOCOL_ALARM_FLOW_CONTROL_ON	0 = Unknown cause
TB640_SS7_MTP2_PROTOCOL_ALARM_FLOW_CONTROL_OFF	0 = Unknown cause

4.4 MTP2 States

States information indicates the current state of the MTP2 Link. This information can be used to determine the quality of service. Status information can be gathered at any time by the host. Collection of status information does not change any information examined.

4.4.1 Link States Get

The TB640_MSG_ID_SS7_MTP2_STATES_GET (request/response) message is used to obtain states from a MTP2 link.

The **request** part of the message TB640_MSG_ID_SS7_MTP2_STATES_GET contains the field:

```

...
TB640_SS7_MTP2_LINK_HANDLE    hMtp2Link;    /* The handle of the MTP2 link to retrieve
...                               state(s) */
...

```

The **response** part of the message TB640_MSG_ID_SS7_MTP2_STATES_GET contains the field:

```

...
TB640_SS7_MTP2_LOCAL_CONGESTION_STATE    LocalCongestionState;
TB640_SS7_MTP2_DATA_LINK_STATE          DatalinkState;

```

```
TB640_SS7_MTP2_ALARM_GENERATION_STATE AlarmGenerationState;
TB640_SS7_MTP2_LOCAL_PROCESSOR_OUTAGE_STATE LocalProcessorOutageState;
TB640_SS7_MTP2_STATES States; /* Structure containing different
... readable states of the link */
```

Structure contains the states parameters in the message:

```
typedef struct _TB640_SS7_MTP2_STATES
{
    TBX_UINT32    un32StructVersion;
    TBX_UINT8     aun8Padding0 [4];
    TBX_BOOL      fUp;
    TBX_BOOL      fAligned;
    TBX_BOOL      fRemoteProcessorOutage;
    TBX_BOOL      fFlowControl;
    TBX_BOOL      fRemoteCongested;

    /* Internal states */
    TB640_SS7_MTP2_INTERNAL_STATE InternalState;

    TBX_UINT32    un32NbOutstandingFrame;
    TBX_UINT32    un32NbDroppedFrame;
    TBX_UINT32    un32CurrentFwdSeq;
    TBX_UINT32    un32CurrentBwdSeq;
    TBX_UINT32    un32RetransmissionCount;
    TBX_UINT32    un32TxQueueSize;
} TB640_SS7_MTP2_STATES, *PTB640_SS7_MTP2_STATES;
```

General explanation of the states parameters:

- The local congestion state parameter (*LocalCongestionState*) indicates the current flow control state from the host. Possible values:

Table 9 - MTP2 local congestion state

Local Congestion State	Description
TB640_SS7_MTP2_LOCAL_CONGESTION_STATE_NO_CHANGE	Used to leave the setting as it is during 'SET' operation.
TB640_SS7_MTP2_LOCAL_CONGESTION_STATE_ON	Local congestion activated.
TB640_SS7_MTP2_LOCAL_CONGESTION_STATE_OFF	Local congestion deactivated.

- The data link state parameter (*DatalinkState*) indicates the current data link. Possible values:

Table 10 - MTP2 data link state

Data Link State	Description
TB640_SS7_MTP2_DATALINK_STATE_NO_CHANGE	Used to leave the setting as it is during 'SET' operation.
TB640_SS7_MTP2_DATALINK_STATE_ENABLED	Data link is enabled.
TB640_SS7_MTP2_DATALINK_STATE_DISABLED	Data link is disabled.

- The alarm generation state parameter (*AlarmGenerationState*) indicates the current alarm generation state. Possible values:

Table 11 - MTP2 alarm generation state

Alarm Generation State	Description
TB640_SS7_MTP2_ALARM_GENERATION_STATE_NO_CHANGE	Used to leave the setting as it is during 'SET' operation.
TB640_SS7_MTP2_ALARM_GENERATION_STATE_ACTIVATED	Alarms are reported to the host.

TB640_SS7_MTP2_ALARM_GENERATION_STATE_DEACTIVATED	Alarms are not reported to the host.
---	--------------------------------------

- The local processor outage state parameter (*LocalProcessorOutageState*) indicates the current local processor outage condition. Possible values:

Data Link State	Description
TB640_SS7_MTP2_LOCAL_PROCESSOR_OUTAGE_STATE_NO_CHANGE	Used to leave the setting as it is during 'SET' operation.
TB640_SS7_MTP2_LOCAL_PROCESSOR_OUTAGE_STATE_ACTIVATED	Local processor is getting short on CPU.
TB640_SS7_MTP2_LOCAL_PROCESSOR_OUTAGE_STATE_DEACTIVATED	Local processor has recovered from "no more CPU" condition.

- The up flag (*fUp*) indicates if the current state of a MTP2 link is up or down. Possible values are **TBX_TRUE** or **TBX_FALSE**.
- The aligned flag (*fAligned*) indicates if the link is aligned or not. Possible values are **TBX_TRUE** or **TBX_FALSE**.
- The remote processor outage flag (*fRemoteProcessorOutage*) indicates if the remote end processor outage is started. Possible values are **TBX_TRUE** or **TBX_FALSE**.
- The flow control flag (*fFlowControl*) indicates a local congestion from the MTP2 link. Possible values are **TBX_TRUE** or **TBX_FALSE**.
- The remote congested flag (*fRemoteCongested*) indicates a remote congestion. Possible values are **TBX_TRUE** or **TBX_FALSE**.
- The internal state parameter (*InternalState*) indicates the current internal link state. Possible values:

Table 12 - MTP2 internal state

Internal State	Description
TB640_SS7_MTP2_INTERNAL_STATE_OUT_OF_SERVICE	Link out of service.
TB640_SS7_MTP2_INTERNAL_STATE_INITIAL_ALIGNMENT	Link in initial alignment.
TB640_SS7_MTP2_INTERNAL_STATE_ALIGNED_READY	Link aligned and ready.
TB640_SS7_MTP2_INTERNAL_STATE_ALIGNED_NOT_READY	Link aligning and not ready.
TB640_SS7_MTP2_INTERNAL_STATE_PROC_OUT	Processor outage detected.
TB640_SS7_MTP2_INTERNAL_STATE_IN_SERVICE	Link in service - data state.
TB640_SS7_MTP2_INTERNAL_STATE_IDLE	Link idling - data state.
TB640_SS7_MTP2_INTERNAL_STATE_NOT_ALIGNED	Link not yet aligned.
TB640_SS7_MTP2_INTERNAL_STATE_IS_ALIGNED	Link is aligned.
TB640_SS7_MTP2_INTERNAL_STATE_PROVING	Link is in initial proving phase.
TB640_SS7_MTP2_INTERNAL_STATE_LOCAL_PROC_OUT	Local processor outage - level 3.
TB640_SS7_MTP2_INTERNAL_STATE_REMOTE_PROC_OUT	Remote processor outage - level 3.
TB640_SS7_MTP2_INTERNAL_STATE_BOTH_PROC_OUT	Both processors outage - level 3.
TB640_SS7_MTP2_INTERNAL_STATE_MONITORING	Link monitoring.
TB640_SS7_MTP2_INTERNAL_STATE_CONGESTION	Link congested - level 2.
TB640_SS7_MTP2_INTERNAL_STATE_POWER_OFF	Power off state.

- The number outstanding frame parameter (*un32NbOutstandingFrame*) indicates the number of outstanding framing in layer 1.
- The number dropped frame parameter (*un32NbDroppedFrame*) indicates the number of frame dropped by the layer 1.
- The current forward sequence parameter (*un32CurrentFwdSeq*) indicates the current forward sequence number.
- The current backward sequence parameter (*un32CurrentBwdSeq*) indicates the current backward sequence number.
- The retransmission count parameter (*un32RetransmissionCount*) indicates the retransmission count of MSUs.
- The transmit queue size parameter (*un32TxQueueSize*) indicates the current transmit queue size.

4.4.2 Link States Set

The TB640_MSG_ID_SS7_MTP2_STATES_SET (request/response) message is used to request a states change for a MTP2 link.

The **request** part of the message TB640_MSG_ID_SS7_MTP2_STATES_SET contains the field:

```
...  
TB640_SS7_MTP2_LINK_HANDLE          hMtp2Link;      /* The handle of the MTP2 link  
                                     instance */  
TB640_SS7_MTP2_LOCAL_CONGESTION_STATE LocalCongestionState;  
TB640_SS7_MTP2_DATA_LINK_STATE      DatalinkState;  
TB640_SS7_MTP2_ALARM_GENERATION_STATE AlarmGenerationState;  
TB640_SS7_MTP2_LOCAL_PROCESSOR_OUTAGE_STATE LocalProcessorOutageState;  
...
```

General explanation of the states parameters:

See section 4.4.1 for parameters definition.

The **response** part of the message TB640_MSG_ID_SS7_MTP2_STATES_SET contains the field:

```
...  
TBX_RESULT                          Result;  
...
```

General explanation of the states response parameters:

- Result code of the request operation.

4.5 MTP2 Statistics

Statistics are gathered to measure the MTP Level 2 software performance. This information can be used to determine the distribution of traffic loads and the quality of service, and to assist in MTP Level 2 software debugging. Statistical information conforms to CCITT standard Q.752.

Statistics information can be gathered at any time by the host. Collection of statistics information may or may not result in the counters being reset.

4.5.1 Link Statistics

The TB640_MSG_ID_SS7_MTP2_STATS_GET (request/response) message is used to obtain statistics from a MTP2 link.

The **request** part of the message TB640_MSG_ID_SS7_MTP2_STATS_GET contains the field:

```
...
TB640_SS7_MTP2_LINK_HANDLE    hMtp2Link;      /* The handle of the MTP2 link to retrieve
                        statistics */
TBX_BOOL                      fResetStats; /* Reset stats if required */
...
```

The **response** part of the message TB640_MSG_ID_SS7_MTP2_STATS_GET contains the field:

```
...
TB640_SS7_MTP2_STATISTICS     Statistics; /* Structure containing statistics for the link */
...
```

Structure contains the statistics parameters in the message:

```
typedef struct _TB640_SS7_MTP2_STATISTICS
{
    TBX_UINT32          un32StructVersion;
    TBX_UINT32          un32InServiceDuration;
    TBX_UINT32          un32LocalBusyDuration;
    TBX_UINT32          un32NbFailure;
    TBX_UINT32          un32NbAbFailure;
    TBX_UINT32          un32NbAckFailure;
    TBX_UINT32          un32NbErrFailure;
    TBX_UINT32          un32NbCongFailure;
    TBX_UINT32          un32NbAlignFailure;
    TBX_UINT32          un32NbSigUnitErr;
    TBX_UINT32          un32NbNack;
    TBX_UINT32          un32NbSifSioTx;
    TBX_UINT32          un32NbBytesRetransmitted;
    TBX_UINT32          un32NbInfoFrameTx;
    TBX_UINT32          un32NbLSSUTx;
    TBX_UINT32          un32NbFISUTx;
    TBX_UINT32          un32NbSifSioRx;
    TBX_UINT32          un32NbInfoFrameRx;
    TBX_UINT32          un32NbReceiveReadyFrameRx;
    TBX_UINT32          un32NbReceiveNotReadyFrameRx;
    TBX_UINT32          un32RXPeakBandwidth;
    TBX_UINT32          un32RXCurrentBandwidth;
    TBX_UINT32          un32TXPeakBandwidth;
    TBX_UINT32          un32TXCurrentBandwidth;
} TB640_SS7_MTP2_STATISTICS, *PTB640_SS7_MTP2_STATISTICS;
```

General explanation of the statistics parameters:

- The in service duration parameter (*un32InServiceDuration*) indicates the time since the link was aligned last time.

- The local busy duration parameter (*un32LocalBusyDuration*) indicates the total time for which link remained locally congested.
- The number failure counter (*un32NbFailure*) indicates the total number of times link has failed for any reason.
- The number abnormal failure counter (*un32NbAbFailure*) indicates the number of times link has due to 2 out of 3 consecutive abnormal backward sequence numbers or forward indicator bits.
- The number acknowledgement failure counter (*un32NbAckFailure*) indicates the number of times link has failed due to excessive delay in acknowledgment from the remote end (T7 expiry).
- The number error failure counter (*un32NbErrFailure*) indicates the number of times link has failed due to excessive error rate on the link (SUERM threshold reached).
- The number congestion failure counter (*un32NbCongFailure*) indicates the number of times link has failed due to excessive congestion on remote end (T6 expired).
- The number alignment failure counter (*un32NbAlignFailure*) indicates the number of times proving/alignment attempt has failed on the link.
- The number signal unit error counter (*un32NbSigUnitErr*) indicates the number of signaling units received in error on the link.
- The number negative acknowledgement counter (*un32NbNack*) indicates the number of times negative acknowledgment has been received on the link.
- The number SIF/SIO transmitted counter (*un32NbSifSioTx*) indicates the number of SIF/SIO packets transmitted.
- The number bytes retransmitted counter (*un32NbBytesRetransmitted*) indicates the number of octets retransmitted on the link.
- The number information frame transmitted counter (*un32NbInfoFrameTx*) indicates the number of information frame transmitted on the link.
- The number LSSU transmitted counter (*un32NbLSSUTx*) indicates the number of LSSU frame transmitted on the link.
- The number FISU transmitted counter (*un32NbFISUTx*) indicates the number of FISU frame transmitted on the link.
- The number SIF/SIO received counter (*un32NbSifSioRx*) indicates the number of SIF/SIO packets received on the link.

- The number information frame received counter (*un32NbInfoFrameRx*) indicates the number of information frame received.
- The number receive ready frame received counter (*un32NbReceiveReadyFrameRx*) indicates the number of "Receive ready" frame received.
- The number receive not ready frame received counter (*un32NbReceiveNotReadyFrameRx*) indicates the number of "Receive not ready" frame received.
- The received peak bandwidth parameter (*un32RXPeakBandwidth*) indicates the maximum bytes received in one second.
- The received current bandwidth parameter (*un32RXCurrentBandwidth*) indicates the number of bytes received in the last second.
- The transmit peak bandwidth parameter (*un32TXPeakBandwidth*) indicates the maximum bytes transmitted in one second.
- The transmitted current bandwidth parameter (*un32TXCurrentBandwidth*) indicates the number of bytes transmitted in the last second.

4.6 MTP2 Standalone mode

The MTP2 standalone mode is only active when the MTP2 Link connection mode is set to TB640_SS7_MTP2_CONNECTION_MODE_STANDALONE or TB640_SS7_MTP2_CONNECTION_MODE_HSL_STANDALONE.

4.6.1 Send Action

The TB640_MSG_ID_SS7_MTP2_CMD_ACTION (request/response) message is used to execute an action upon a link.

The **request** part of the message TB640_MSG_ID_SS7_MTP2_CMD_ACTION contains the field:

```

...
TB640_SS7_MTP2_LINK_HANDLE    hMtp2Link;    /* The handle of the MTP2 link to execute an
                               action */
TB640_SS7_MTP2_ACTION         Action;         /* Type of action to do on the link */
TBX_UINT32                    un32Value;         /* Supplemental value used depending on the
                               action */
...

```

The **response** part of the message TB640_MSG_ID_SS7_MTP2_CMD_ACTION contains the fields:

```

...
TBX_RESULT                    Result;            /* Result code of the request operation */
TB640_SS7_MTP2_LINK_HANDLE    hMtp2Link;        /* The handle of the MTP2 link */
TB640_SS7_MTP2_ACTION         Action;          /* Type of action did on the link */
TBX_UINT32                    un32Value;        /* Supplemental value used depending on the
                               action */
...

```

General explanation of the action parameters:

- The action parameter (*Action*) indicates the type of action for a link. Possible values:

Table 13 - MTP2 action

Action	Description
TB640_SS7_MTP2_ACTION_EMERGENCY_START	Bring the link up with emergency from now on (ITU and ANSI only).
TB640_SS7_MTP2_ACTION_EMERGENCY_STOP	Bring the link in normal way (no emergency) from now on (ITU and ANSI only).
TB640_SS7_MTP2_ACTION_FLUSH_BUFFERS	Flush buffers (ITU 92 and ANSI 92).
TB640_SS7_MTP2_ACTION_CONTINUE	Continue (ITU 92 and ANSI 92).
TB640_SS7_MTP2_ACTION_RETRIEVE_MESSAGES	Retrieve messages based on the given backward sequence number.
un32Value: <i>backward sequence number</i>	
TB640_SS7_MTP2_ACTION_DROP_TRANSMIT_QUEUE	Drop transmit queue messages.
TB640_SS7_MTP2_ACTION_RETRIEVE_BSN	Retrieve backwards sequence number.
un32Value: <i>backward sequence number</i>	
TB640_SS7_MTP2_ACTION_CONNECT	Connect levels 1 and 2.
un32Value: TB640_SS7_MTP2_CONNECTION_STATUS_NORMAL	
TB640_SS7_MTP2_CONNECTION_STATUS_EMERGENCY	
TB640_SS7_MTP2_ACTION_DISCONNECT	Disconnects link at level 2.

4.6.2 Send Data

The TB640_MSG_ID_SS7_MTP2_CMD_SEND_DATA (request/response) message is used by the host application to transmit a buffer. The message can contain 16 different frames. Each frames of the array will be sent with the same buffer order. If one frame generates an error while being enqueued, all subsequent frames in the same message will NOT be sent to prevent frames to be sent in the wrong order.

The **request** part of the message TB640_MSG_ID_SS7_MTP2_CMD_SEND_DATA contains the fields:

...

```
TB640_SS7_MTP2_LINK_HANDLE    hMtp2Link;
TBX_UINT32                    un32NbFrameInMsg;
TBX_UINT64                    aun64UserContext [TB640_SS7_MTP2_MAX_FRAME_IN_MESSAGE];
TBX_UINT32                    aun32FrameOfset [TB640_SS7_MTP2_MAX_FRAME_IN_MESSAGE];
```

```

TBX_UINT32          aun32PayloadSize [TB640_SS7_MTP2_MAX_FRAME_IN_MESSAGE];
TB640_SS7_MTP2_PRIORITY  aPriority [TB640_SS7_MTP2_MAX_FRAME_IN_MESSAGE];
TBX_UINT8          aun8Payload [1];
...

```

General explanation of the parameters of send data configuration:

- The Mtp2 link handle parameter (*hMtp2Link*) specifies the handle of the MTP2 link to transmit frames.
- The number frame in message parameter (*un32NbFrameInMsg*) indicates the number of frames included in this request.
- The array user context parameter (*aun64UserContext*), it's an array of user contexts (user value) that are going to be returned in the response for a frame of the message.
- The array of frame offset parameter (*aun32FrameOffset*) specifies the offset from the start of the *aun8Payload* field to which each frame starts.
- The array of payload size parameter (*aun32PayloadSize*) indicates the byte count for every frame in the *aun8Payload*.
- The array of priority parameter (*aPriority*) indicates the priority of each frame (**not used**).
- The array of payload parameter (*aun8Payload*) indicates the byte count for every frame in the *aun8Payload*.
- The payload parameter (*aun8Payload*) contains all sent frames.

The **response** part of the message TB640_MSG_ID_SS7_MTP2_CMD_SEND_DATA contains the fields:

```

...
TBX_RESULT          Result;          /* Result code of the request operation */
TBX_UINT32          un32NbFrameSentOk;
TBX_UINT32          un32NbFrameSentFail;
TBX_UINT64          aun64UserContext [TB640_SS7_MTP2_MAX_FRAME_IN_MESSAGE];
...

```

General explanation of some parameters of send data response:

- The number of frame sent ok parameter (*un32NbFrameSentOk*) indicates the number of frames sent successfully.
- The number of frame sent fail parameter (*un32NbFrameSentFail*) indicates the number of frames not sent.
- The array user context parameter (*aun64UserContext*), it's an array of user contexts (user value) copied from request. This array contains user context in the same order than TB640_REQ_SS7_MTP2_CMD_SEND_DATA request.

4.6.3 Received Data

The TB640_MSG_ID_SS7_MTP2_NOTIF_RECEIVED_DATA (event) notification message is received by the host application when a buffer is received on an MTP2 link. The notification message can contain 16 different frames. Each frames of the array will be received with the same MTP2 received order.

Structure contains the **event** notification received data for a MTP2 received data:

```
typedef struct _TB640_EVT_SS7_MTP2_NOTIF_RECEIVED_DATA
{
    TBX_MSG_HEADER                Header;
    TBX_UINT32                    un32MsgVersion;
    TB640_SS7_MTP2_LINK_HANDLE    hMtp2Link;
    TBX_UINT32                    un32NbFrameInMsg;
    TBX_UINT8                    aun8Padding0 [4];
    TBX_UINT32                    aun32FrameOffset [TB640_SS7_MTP2_MAX_FRAME_IN_MESSAGE];
    TBX_UINT32                    aun32PayloadSize [TB640_SS7_MTP2_MAX_FRAME_IN_MESSAGE];
    TB640_SS7_MTP2_PRIORITY      aPriority [TB640_SS7_MTP2_MAX_FRAME_IN_MESSAGE];
    TBX_UINT8                    aun8Payload [1];
} TB640_EVT_SS7_MTP2_NOTIF_RECEIVED_DATA, *PTB640_EVT_SS7_MTP2_NOTIF_RECEIVED_DATA;
```

General explanation of the field of event notification received data:

- The Mtp2 link handle parameter (*hMtp2Link*) specifies the handle of the MTP2 link.
- The number of frame parameter (*un32NbFrameInMsg*) indicates the number of frames included in the *aun8Payload*.
- The array of frame offset parameter (*aun32FrameOffset*) specifies the offset from the start of the *aun8Payload* field to which each frame starts.
- The array of payload size parameter (*aun32PayloadSize*) indicates the byte count for every frame in the *aun8Payload*.
- The array of priority parameter (*aPriority*) indicates the priority of each frame (**not used**).
- The payload parameter (*aun8Payload*) contains all received frames.

4.6.4 Retrieved Data

The TB640_MSG_ID_SS7_MTP2_NOTIF_RETRIEVED_DATA (event) notification message is received by the host application for every non-transmitted (according to the backward sequence number) buffer after a request using the primitive TB640_MSG_ID_SS7_MTP2_CMD_ACTION was sent to the link. Frames are returned in the same order as within the array.

Structure contains the **event** notification received data for a MTP2 retrieved data:

```
typedef struct _TB640_EVT_SS7_MTP2_NOTIF_RETRIEVED_DATA
{
    TBX_MSG_HEADER                Header;
```



```
TBX_UINT32          un32MsgVersion;
TB640_SS7_MTP2_LINK_HANDLE  hMtp2Link;
TBX_UINT32          un32NbFrameInMsg;
TBX_BOOL            fMoreFrameToCome;
TBX_UINT32          aun32FrameOffset [TB640_SS7_MTP2_MAX_FRAME_IN_MESSAGE];
TBX_UINT32          aun32PayloadSize [TB640_SS7_MTP2_MAX_FRAME_IN_MESSAGE];
TBX_UINT8           aun8Payload [1];

} TB640_EVT_SS7_MTP2_NOTIF_RETRIEVED_DATA, *PTB640_EVT_SS7_MTP2_NOTIF_RETRIEVED_DATA;
```

General explanation of the field of event notification retrieved data:

- The Mtp2 link handle parameter (*hMtp2Link*) specifies the handle of the MTP2 link.
- The number of frame parameter (*un32NbFrameInMsg*) indicates the number of frames included in this event.
- The more frame to come flag (*fMoreFrameToCome*) is set to `TBX_TRUE` for all retrieved data except the last data buffer. For the last data buffer the flag is set to `TBX_FALSE`.
- The array of frame offset parameter (*aun32FrameOffset*) specifies the offset from the start of the *aun8Payload* field to which each frame starts.
- The array of payload size parameter (*aun32PayloadSize*) indicates the byte count for every frame in the *aun8Payload*.
- The payload parameter (*aun8Payload*) contains all retrieved frames.

5 MTP3

5.1 Overview

This section gives a description of the TB640 Message Transfer Part (MTP) level 3 layer architecture and usage. This layer is referred to as MTP3 in the rest of the section.

5.1.1 Summary

The MTP3 layer provides messages routing between signaling points in a SS7 network. This layer reroutes traffic away from failed links and signaling points and controls traffic when congestion occurs. MTP3 provides the network layer functionality within the OSI model.

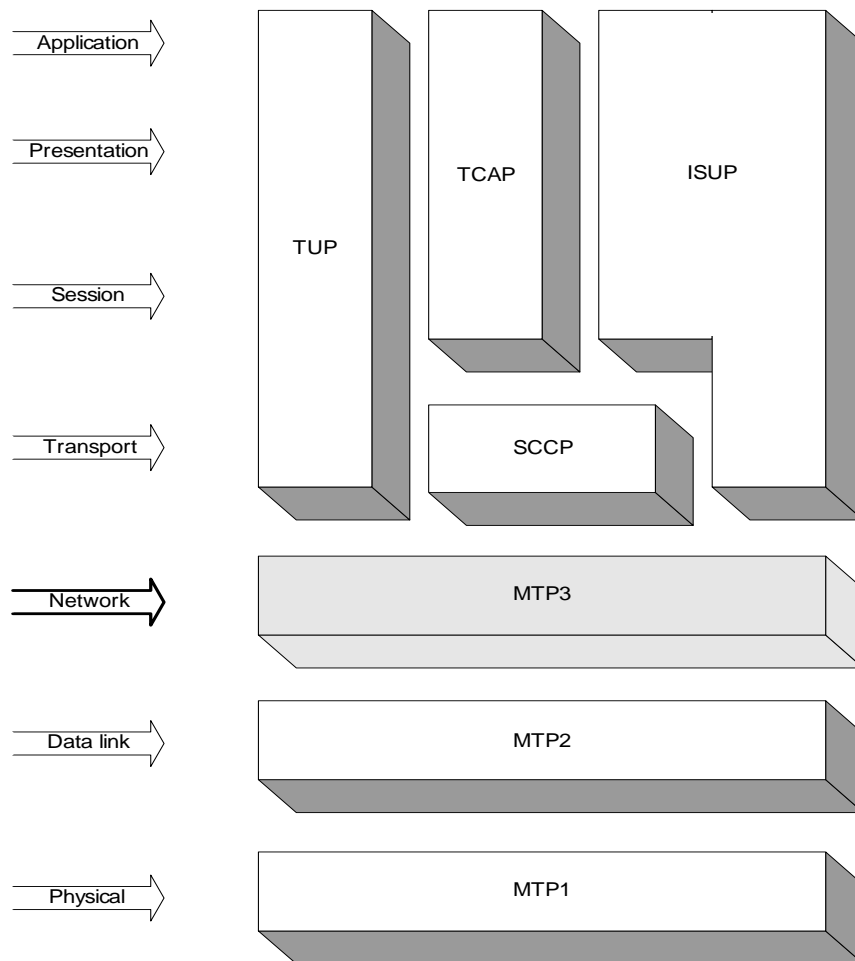



Figure 4 - MTP3 OSI model

5.1.2 Features


MTP3 supports the following features:

- Acts as a signaling point or signaling transfer point.

- Handles either national or international signaling traffic.
- Supports national option with and without multiple congestion priorities.
- Supports message discrimination function. If the signaling point is the destination point, the message is delivered to the message distribution function. If the signaling point is not the destination point, and the transfer capability is enabled, the message is delivered to the message routing function.
- Supports message distribution function. Delivers a message to the appropriate user part or MTP3 function based on the service indicator.
- Supports message routing function. Selects the appropriate signaling link for the transmission of messages. Performs load sharing between links or linksets.
- Uses signaling traffic management procedures to divert signaling traffic from signaling links or signaling routes. Also used to temporarily reduce the quantity of traffic in case of congestion. The following signaling traffic management procedures are supported:
 - Changeover
 - Changeback
 - Forced rerouting
 - Controlled rerouting
 - Signaling point restart
 - Management inhibiting
 - Processor outage

 The signaling point restart is not required for certain country variants, and can be disabled through configuration.
- Uses signaling link management procedures to control the local signaling links. The following basic signaling link management procedures are supported:
 - Signaling link activation
 - Signaling link restoration
 - Signaling link deactivation
 - Linkset activation (normal and emergency restart)
 - Signaling link congestion
 - Signaling link test
 - Signal routing test (only for TTC/NTT, Japan)
- Uses signaling route management procedures to ensure a reliable exchange of information between signaling points concerning the availability of signaling routes. The following signaling route management procedures are supported:
 - Transfer prohibited
 - Transfer cluster prohibited (ANSI/Bellcore)
 - Transfer allowed
 - Transfer cluster allowed (ANSI/Bellcore)

- Transfer restricted
- Transfer cluster restricted (ANSI/Bellcore)
- Transfer controlled (international and national with congestion priorities)
- Signaling route set test
- Signaling cluster route set test (ANSI/Bellcore)
- Signaling route set congestion test
- Broadcasting of route management messages (such as transfer prohibit and transfer allowed) with multiple point codes (only for TTC/NTT, Japan)

 Transfer restricted, MTP restart, and route-set-congestion test procedures are national options and can be enabled or disabled through configuration. Supporting multiple congestion priorities is also a national option and can be controlled through configuration. The incoming cluster route management messages (**TCP/TCR/TCA/RCP**) are handled by MTP3. MTP3 software does not generate cluster management messages and cluster routing.

5.1.3 Architecture

MTP3 defines the functions and procedures of the signaling system for *signaling message handling* and *signaling network management*.

- Signaling message handling consists of the actual transfer of a signaling message to the proper signaling link or user part.
- Signaling network management consists of controlling message routing and configuring the signaling network facilities based on predetermined information and the status of the signaling network facilities.

The MTP3 layer is a service provider to the user part such as ISUP, SCCP, and so on. The MTP3 layer is a service user of MTP2 (as shown on Figure 5).

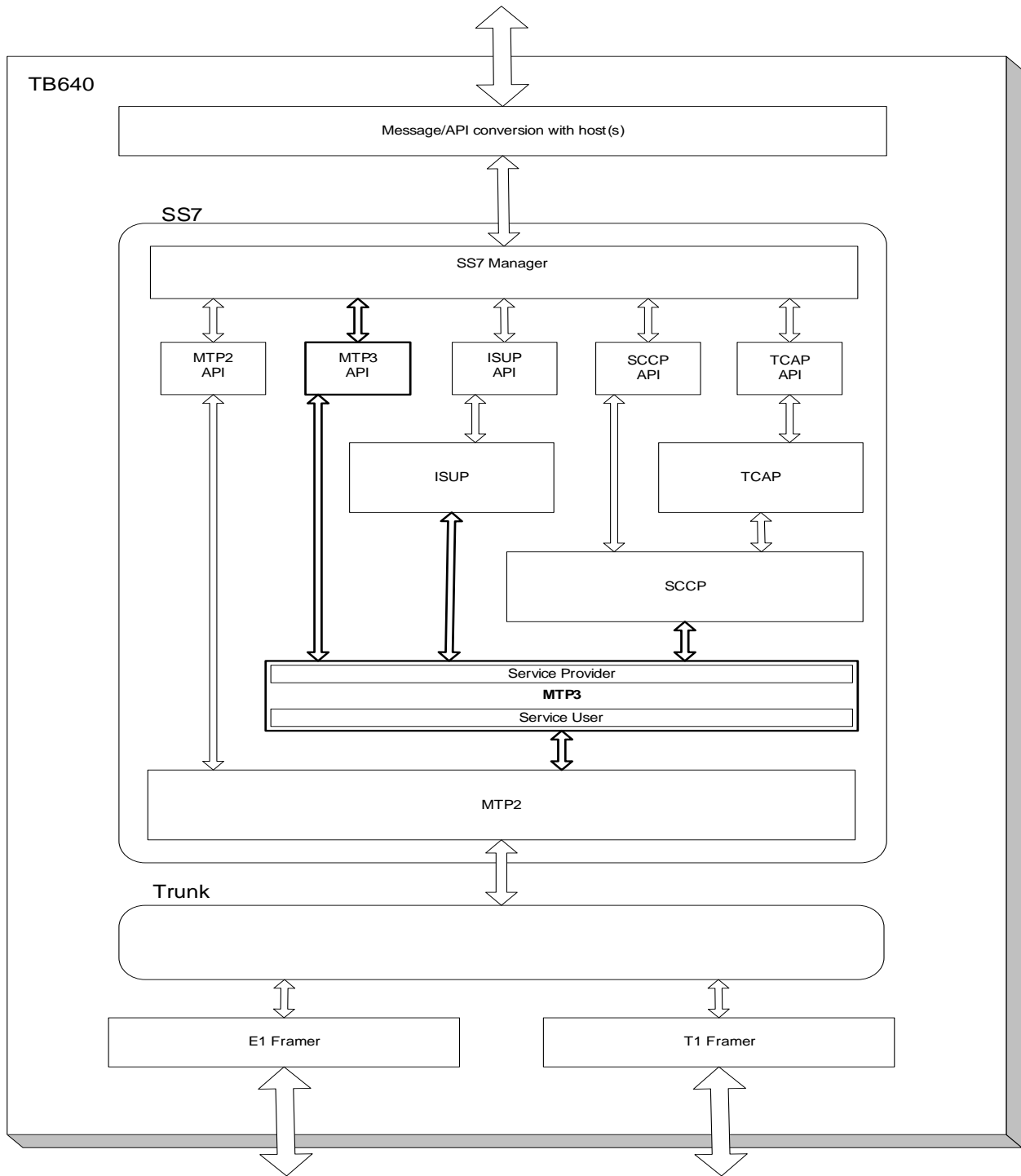


Figure 5 - MTP3 layer organization within TB640

MTP3 Userpart can be used in conjunction with an above local ISUP or SCCP layer and can be used as a standalone and communicates with remote ISUP or SCCP layer (host application). See the connection mode table in the Userpart Configuration section.

A MTP3 layer has an Userpart section which represent a protocol variant. For a Userpart you must define:

- Linkset(s) (association of Links)
- Routes (for a specific or a range of dpc)

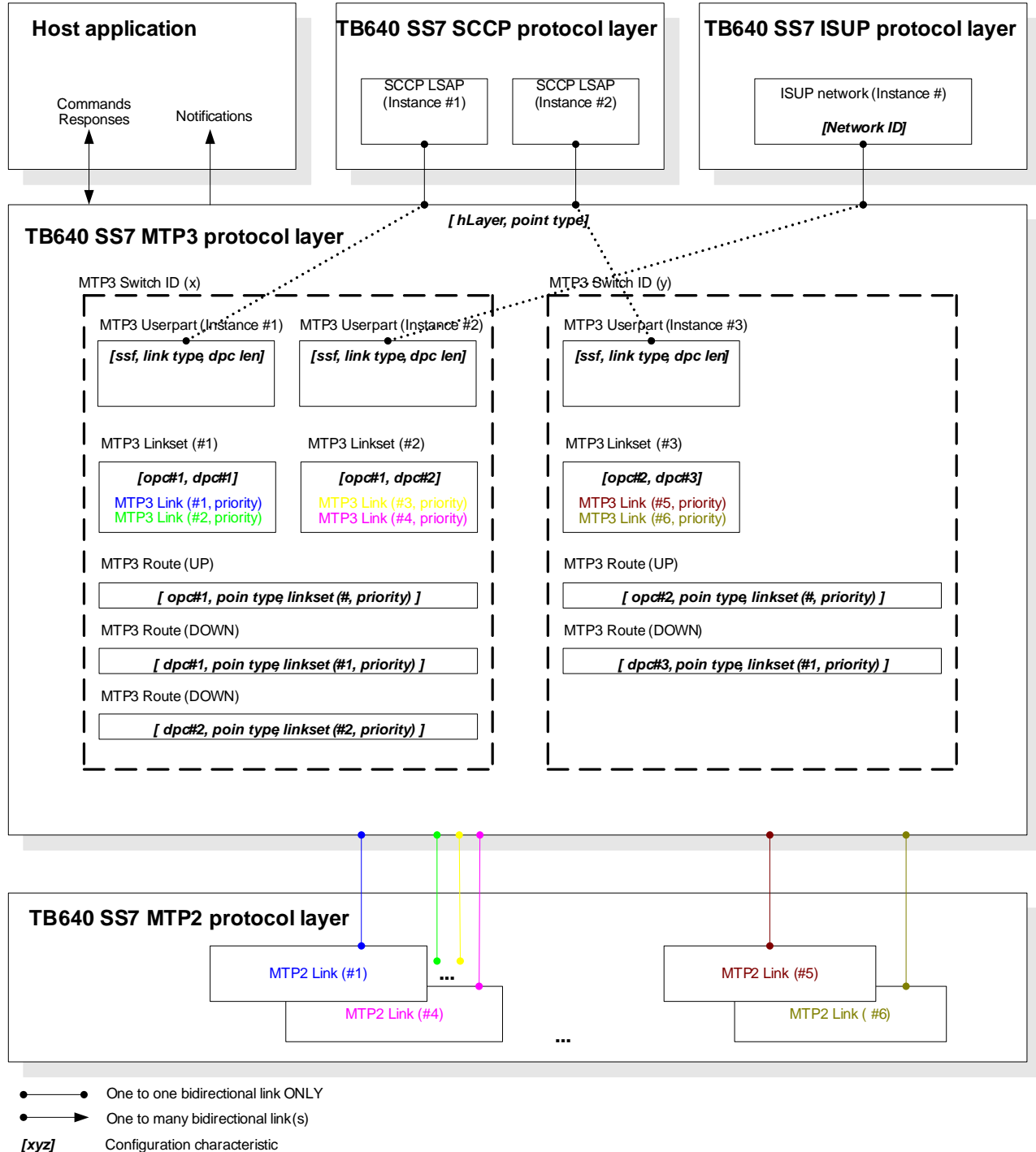


Figure 6 - MTP3 layer hierarchy

A MTP3 Link must be associated with one MTP2 Link. It's a 1 to 1 association. The MTP3 Link defined a physical signaling link between the TX of this node and the adjacent SP (signaling point).

A MTP3 Linkset is a collection of 1 to 16 links with same or different priority, going to the same DPC into a single entity. If you set multiple links with the same priority then you obtain load sharing between these links. The MTP3 always used the highest priority available link.

- ✍ Load sharing requires at least two signaling links for all bit rates, but more may be needed at lower bit rates. When two signaling links are used, each of them should be able to carry the total signaling traffic in case of failure of the other link. When more than two links are used, sufficient reserve link capacity should exist to satisfy the availability requirements specified in Recommendation Q.706 (from: ITU-T Recommendation Q.705 section 4.4).

A MTP3 Route with *Direction* as TB640_SS7_MTP3_ROUTE_DIRECTION_DOWN indicates the destination point code (DPC) that is an accessible from this configuration node. Each linkset within the route's may have an different priority assigned. If you have multiple linksets with the same priority then you obtain load sharing between these linkset. The MTP3 always used the highest priority available linkset.

Relationship scenarios between links, linksets and route:

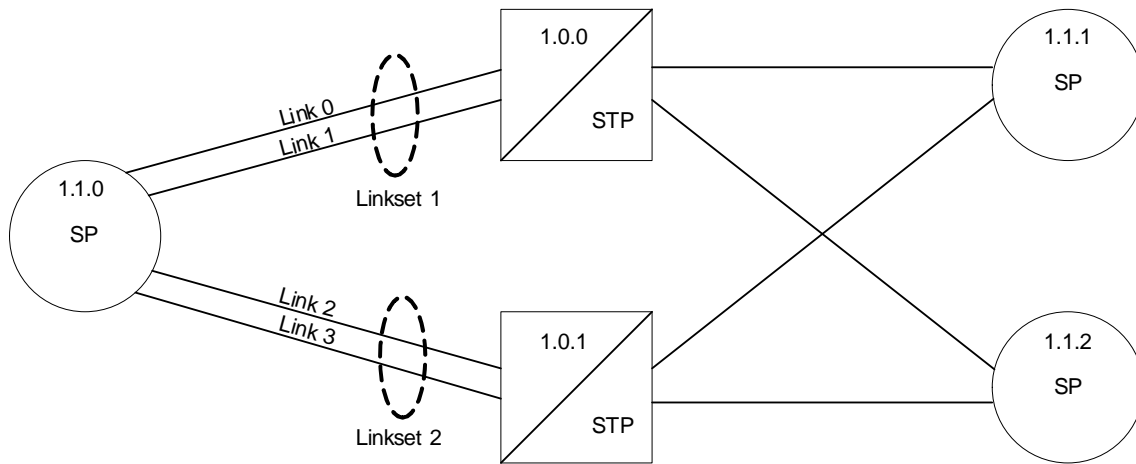


Figure 7- MTP3 relationship between links, linksets and routes

Scenario 1

SP (1.1.0) Configuration

Linkset 1:

Link 0 (priority=0), Link 1 (priority=1)

Linkset 2:

Link 2 (priority=0), Link 3 (priority=1)

Route 0: (self-route)

DPC=1.1.0, SP, UP, Linkset 1 (priority=0)

Route 1:

DPC=1.0.0, STP, DOWN, Linkset 1 (priority=0)

Route 2:

DPC=1.0.1, STP, DOWN, Linkset 2 (priority=0)

Route 3:

DPC=1.1.1, SP, DOWN, Linkset 1 (priority=0), Linkset 2 (priority=1)

Route 4:

DPC=1.1.2, SP, DOWN, Linkset 2 (priority=0), Linkset 1 (priority=1)

Table 14 - MTP3 traffic distribution scenario #1

Originate SP	Linkset 1		Linkset 2		Destination SP
	Link 0	Link 1	Link 2	Link 3	
1.1.0	State: UP		State: UP		1.1.1
	State: UP Traffic: Yes	State: UP Traffic: No	State: UP Traffic: No	State: UP Traffic: No	
1.1.0	State: UP		State: UP		1.1.1
	State: DOWN Traffic: No	State: UP Traffic: Yes	State: UP Traffic: No	State: UP Traffic: No	
1.1.0	State: UP		State: UP		1.1.1
	State: UP Traffic: Yes	State: DOWN Traffic: No	State: UP Traffic: No	State: UP Traffic: No	
1.1.0	State: UP		State: UP		1.1.1
	State: UP Traffic: Yes	State: UP Traffic: No	State: DOWN Traffic: No	State: UP Traffic: No	
1.1.0	State: UP		State: UP		1.1.1
	State: UP Traffic: Yes	State: UP Traffic: No	State: UP Traffic: No	State: DOWN Traffic: No	
1.1.0	State: DOWN		State: UP		1.1.1
	State: DOWN Traffic: No	State: DOWN Traffic: No	State: UP Traffic: Yes	State: UP Traffic: No	
1.1.0	State: DOWN		State: UP		1.1.1
	State: DOWN Traffic: No	State: DOWN Traffic: No	State: DOWN Traffic: No	State: UP Traffic: Yes	
1.1.0	State: DOWN		State: DOWN		1.1.1
	State: DOWN Traffic: No	State: DOWN Traffic: No	State: DOWN Traffic: No	State: DOWN Traffic: No	

Scenario 2

SP (1.1.0) Configuration

Linkset 1: [*load sharing between link 0 and link 1*]
 Link 0 (priority=0), Link 1 (**priority=0**)

Linkset 2:
 Link 2 (priority=0), Link 3 (priority=1)

Route 0: (self-route)
 DPC=1.1.0, SP, UP, Linkset 1 (priority=0)

Route 1:
 DPC=1.0.0, STP, DOWN, Linkset 1 (priority=0)

Route 2:
 DPC=1.0.1, STP, DOWN, Linkset 2 (priority=0)

Route 3:
 DPC=1.1.1, SP, DOWN, Linkset 1 (priority=0), Linkset 2 (priority=1)

Route 4:
 DPC=1.1.2, SP, DOWN, Linkset 2 (priority=0), Linkset 1 (priority=1)

Table 15 - MTP3 traffic distribution scenario #2

Originate SP	Linkset 1		Linkset 2		Destination SP
	Link 0	Link 1	Link 2	Link 3	
1.1.0	State: UP		State: UP		1.1.1
	State: UP Traffic: Yes	State: UP Traffic: Yes	State: UP Traffic: No	State: UP Traffic: No	
1.1.0	State: UP		State: UP		1.1.1
	State: DOWN Traffic: No	State: UP Traffic: Yes	State: UP Traffic: No	State: UP Traffic: No	
1.1.0	State: UP		State: UP		1.1.1
	State: UP Traffic: Yes	State: DOWN Traffic: No	State: UP Traffic: No	State: UP Traffic: No	
1.1.0	State: UP		State: UP		1.1.1
	State: UP Traffic: Yes	State: UP Traffic: Yes	State: DOWN Traffic: No	State: UP Traffic: No	
1.1.0	State: UP		State: UP		1.1.1
	State: UP Traffic: Yes	State: UP Traffic: Yes	State: UP Traffic: No	State: DOWN Traffic: No	
1.1.0	State: DOWN		State: UP		1.1.1
	State: DOWN Traffic: No	State: DOWN Traffic: No	State: UP Traffic: Yes	State: UP Traffic: No	
1.1.0	State: DOWN		State: UP		1.1.1
	State: DOWN Traffic: No	State: DOWN Traffic: No	State: DOWN Traffic: No	State: UP Traffic: Yes	
1.1.0	State: DOWN		State: DOWN		1.1.1
	State: DOWN Traffic: No	State: DOWN Traffic: No	State: DOWN Traffic: No	State: DOWN Traffic: No	

Scenario 3

SP (1.1.0) Configuration

Linkset 1: [load sharing between link 0 and link 1]
Link 0 (priority=0), Link 1 (**priority=0**)

Linkset 2: [load sharing between link 2 and link 3]
Link 2 (priority=0), Link 3 (**priority=0**)

Route 0: (self-route)
DPC=1.1.0, SP, UP, Linkset 1 (priority=0)

Route 1:
DPC=1.0.0, STP, DOWN, Linkset 1 (priority=0)

Route 2:
DPC=1.0.1, STP, DOWN, Linkset 2 (priority=0)

Route 3:
DPC=1.1.1, SP, DOWN, Linkset 1 (priority=0), Linkset 2 (priority=1)

Route 4:
DPC=1.1.2, SP, DOWN, Linkset 2 (priority=0), Linkset 1 (priority=1)

Table 16 - MTP3 traffic distribution scenario #3

Originate SP	Linkset 1		Linkset 2		Destination SP
	Link 0	Link 1	Link 2	Link 3	
1.1.0	State: UP		State: UP		1.1.1
	State: UP Traffic: Yes	State: UP Traffic: Yes	State: UP Traffic: No	State: UP Traffic: No	
1.1.0	State: UP		State: UP		1.1.1
	State: DOWN Traffic: No	State: UP Traffic: Yes	State: UP Traffic: No	State: UP Traffic: No	
1.1.0	State: UP		State: UP		1.1.1
	State: UP Traffic: Yes	State: DOWN Traffic: No	State: UP Traffic: No	State: UP Traffic: No	
1.1.0	State: UP		State: UP		1.1.1
	State: UP Traffic: Yes	State: UP Traffic: Yes	State: DOWN Traffic: No	State: UP Traffic: No	
1.1.0	State: UP		State: UP		1.1.1
	State: UP Traffic: Yes	State: UP Traffic: Yes	State: UP Traffic: No	State: DOWN Traffic: No	
1.1.0	State: DOWN		State: UP		1.1.1
	State: DOWN Traffic: No	State: DOWN Traffic: No	State: UP Traffic: Yes	State: UP Traffic: Yes	
1.1.0	State: DOWN		State: UPs		1.1.1
	State: DOWN Traffic: No	State: DOWN Traffic: No	State: DOWN Traffic: No	State: UP Traffic: Yes	
1.1.0	State: DOWN		State: DOWN		1.1.1
	State: DOWN Traffic: No	State: DOWN Traffic: No	State: DOWN Traffic: No	State: DOWN Traffic: No	

Scenario 4

SP (1.1.0) Configuration

Linkset 1: [load sharing between link 0 and link 1]
 Link 0 (priority=0), Link 1 (**priority=0**)

Linkset 2: [load sharing between link 2 and link 3]
 Link 2 (priority=0), Link 3 (**priority=0**)

Route 0: (self-route)
 DPC=1.1.0, SP, UP, Linkset 1 (priority=0)

Route 1:
 DPC=1.0.0, STP, DOWN, Linkset 1 (priority=0)

Route 2:
 DPC=1.0.1, STP, DOWN, Linkset 2 (priority=0)

Route 3: [load sharing between linkset 1 and linkset 2]
 DPC=1.1.1, SP, DOWN, Linkset 1 (priority=0), Linkset 2 (**priority=0**)

Route 4:
 DPC=1.1.2, SP, DOWN, Linkset 2 (priority=0), Linkset 1 (priority=1)

Table 17 - MTP3 traffic distribution scenario #4

Originate SP	Linkset 1		Linkset 2		Destination SP
	Link 0	Link 1	Link 2	Link 3	
1.1.0	State: UP		State: UP		1.1.1
	State: UP Traffic: Yes	State: UP Traffic: Yes	State: UP Traffic: Yes	State: UP Traffic: Yes	
1.1.0	State: UP		State: UP		1.1.1
	State: DOWN Traffic: No	State: UP Traffic: Yes	State: UP Traffic: Yes	State: UP Traffic: Yes	
1.1.0	State: UP		State: UP		1.1.1
	State: UP Traffic: Yes	State: DOWN Traffic: No	State: UP Traffic: Yes	State: UP Traffic: Yes	
1.1.0	State: UP		State: UP		1.1.1
	State: UP Traffic: Yes	State: UP Traffic: Yes	State: DOWN Traffic: No	State: UP Traffic: Yes	
1.1.0	State: UP		State: UP		1.1.1
	State: UP Traffic: Yes	State: UP Traffic: Yes	State: UP Traffic: Yes	State: DOWN Traffic: No	
1.1.0	State: DOWN		State: UP		1.1.1
	State: DOWN Traffic: No	State: DOWN Traffic: No	State: UP Traffic: Yes	State: UP Traffic: Yes	
1.1.0	State: DOWN		State: UPs		1.1.1
	State: DOWN Traffic: No	State: DOWN Traffic: No	State: DOWN Traffic: No	State: UP Traffic: Yes	
1.1.0	State: DOWN		State: DOWN		1.1.1
	State: DOWN Traffic: No	State: DOWN Traffic: No	State: DOWN Traffic: No	State: DOWN Traffic: No	

5.1.3.1 Addressing


MTP3 fully supports the use of addresses to identify Signaling Points (SPs). The following section describes signaling points in detail.

5.1.3.1.1 Signaling Points

Signaling points are identified using Destination Point Codes (DPCs). DPCs can be a number, up to 24 bits, based on the variant (for example, ITU has 14 bit DPC, ANSI has 24 bit DPC).

In MTP3 software, each point code can be uniquely identified as a local SS7 point code or a remote SS7 point code.

In MTP3 software, each signaling point is additionally identified by a network identifier (ID) which is unique to all signaling point codes within the network. This addressing is useful for simultaneously supporting multiple variants of MTP3 where each signaling point can be addressed by MTP3 using the combination of DPC and network ID.

-  By virtue of the network ID, MTP3 can support the same DPC value for more than one signaling point if each of the signaling points that have the same DPC value belongs to a different network.

The Destination Point Code (DPC), Origination Point Code (OPC), and Signaling Link Selection (SLS) fields are part of the routing label in MTP3.

The service indicator and network indicator are part of the Service Information Octet (SIO) of the MSU. The service indicator is four bits and can indicate an MTP service user or signaling network management service. The network indicator is two bits and can indicate a national or international network.

5.1.3.2 Routing

By directing messages to the appropriate signaling point, the routing mechanism establishes the connection between the calling and the called party.

Message routes pass through a minimum number of intermediate signaling transfer points. Routing at each signaling transfer point is not affected by the message routes used up to the concerned signaling transfer point. If more than one message route is available, signaling traffic is load-shared by the available message routes. Messages related to a given user transaction and sent in a given direction are routed over the same message route.

Routing between functional blocks depends on whether transfer capability is enabled:

- If transfer capability is enabled, messages can be routed between signaling links, or between a signaling link and the upper layer or signaling network management.

- If transfer capability is disabled, messages cannot be routed between signaling links, though they can be routed between a signaling link and MTP3 service users or signaling network management.

5.1.3.3 Flow Control

The MTP3 software aids to data flow control are based on layer 2 inputs. When a link is congested, as indicated by a flow control indication from layer 2, MTP3 starts queuing traffic meant for that link. MTP3 also maintains thresholds for the queued messages. For each threshold, MTP3 performs one of the following actions:

- Send a transfer controlled message at an STP in response to a data received from the adjacent point code for routing to a remote point code
- Send a status indication to its service users at an SP in response to a data transfer request received from user
- Drop the data if the priority with which the data to be sent is lower than the current congestion threshold (only for national networks with multiple congestion priorities).

The above actions allow the data originating entities to take appropriate actions in case of heavy loads and also prevents system crashes due to lack of resources.

5.1.3.4 Multiple OPC Capability

MTP3 allows customers to configure multiple OPCs. The software can route user traffic with different OPCs, where the OPC is supplied by the user in each data request. MTP3 can also receive traffic from the network for different own point codes and distribute the traffic to the users.

5.1.4 Specification

The MTP3 software conforms to the following standards:

- ETS 300 008 – Integrated Services Digital Network; CCITT Signaling System No 7, Message Transfer Part to Support International Interconnection, ETSI.
- GF001-9001 – Technical Specifications of SS7 for National Telephone Network of China, CHINA.
- GR-246, Issue 3, T1.111.4 – Signaling Network Functions and Messages, Bellcore, December 1998.
- GR-246, Issue 3, T1.111.5 - Signaling Network Structure, Bellcore.
- GR-246, Issue 3, T1.111.7 - Testing and Maintenance, Bellcore.
- INTERNET-DRAFT– SS7 MTP3-User Adaptation Layer (M3UA).
- IS 7498 – Open Systems Interconnection – Basic Reference Model, ISO.
- IS 7498 DAD 1 – Open Systems Interconnection – Basic Reference Model Addendum 1: Connectionless, Data Transmission, ISO.
- JT - Q.704 – MTP Signaling Network Functions, Telecommunication Technology Committee (TTC), Japan.
- JT - Q.707 – MTP Testing and Maintenance, Telecommunication Technology Committee (TTC), Japan.
- NTT - Q.704 – MTP Signaling Network Functions, Nippon Telegraph and Telephone (NTT), Japan.

- NTT - Q.707 – MTP Testing and Maintenance, Nippon Telegraph and Telephone (NTT), Japan.
- Q.700 – Introduction to CCITT Signaling System No. 7, CCITT.
- Q.701 – Functional Description of the Message Transfer Part of Signaling System No. 7, CCITT.
- Q.702 – Signaling Data Link, CCITT.
- Q.703 – Signaling Link, CCITT.
- Q.704 – Signaling Network Functions and Messages, CCITT.
- Q.705 – Signaling Network Structure, CCITT.
- Q.706 – Message Transfer Part Signaling Performance, CCITT.
- Q.707 – Testing and Maintenance, CCITT.
- Q.708 – Number of International Signaling Point Codes, CCITT.
- Q.709 – Hypothetical Signaling Reference Connection, CCITT.
- Q.710 – Simplified MTP Version for Small Systems, CCITT.
- Q.752 – Monitoring and Measurements for Signaling Systems No.7 Networks, CCITT, 1997.
- Q.782 – MTP Level 3 Test Specification, CCITT.
- Q.2140 – MTP Level 3 Functions and Messages Using the Services of ITU Q.2140, ITU.
- Q.2210 – MTP Level 3 Functions and Messages Using the Services of CCITT Recommendation Q.2140, ITU.
- RFC 2719 – Framework Architecture for Signaling Transport, IETF.
- T1.111.4 – Signaling Network Functions and Messages, ANSI.
- T1.111.5 - Signaling Network Structure, ANSI.
- T1.111.7 - Testing and Maintenance, ANSI.

MTP3 can support different protocol variants:

- ANSI (used for ANSI88 and ANSI92)
- ANSI96 (used for ANSI96 and TELCORDIA)
- ITU (used for ITU88, ITU92, ITU97, Q767, SINGAPORE and ETSI)
- CHINA
- TTC (*not available yet*)
- NTT (*not available yet*)

5.2 MTP3 Configuration


5.2.1 Configuration of layer


General guidelines for configuration of MTP3 for a proper operation include the following:


1. The MTP3 general allocation³ (TB640_MSG_ID_SS7_MTP3_OP_ALLOC) must precede all other messages (other configuration MTP3 alloc, get, states and stats). The response of this message is a MTP3 handle.


³ All fields of a configuration message alloc must be filled unless explicitly optional or not defined for certain variants.


2. The MTP3 Userpart allocation (TB640_MSG_ID_SS7_MTP3_OP_USERPART_ALLOC) must be made (with the MTP3 handle from **step 1**) for a particular variant. The response of this message is a MTP3 Userpart handle.
3. The MTP3 Linkset allocation (TB640_MSG_ID_SS7_MTP3_OP_LINKSET_ALLOC) must be made (with the MTP3 handle from **step 1** and the Userpart handle from **step 2**) for a particular OPC (Origin Point Code) and a particular adjacent DPC (Destination Point Code). The response of this message is a MTP3 Linkset handle.
4. The MTP3 Link allocation (TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC) must be made (with the MTP3 handle from **step 1**, Linkset handle from **step 3** and the unique identifier of the MTP2 Link) to connect with the MTP2 link layer. The response of this message is a MTP3 Link handle.
5. The MTP3 Route allocation (TB640_MSG_ID_SS7_MTP3_OP_ROUTE_ALLOC) must be made (with the MTP3 handle from **step 1** and Linkset(s) handle from **step 3**) to create a self-route and a route to a specific or to a range of DPC. The response of this message is MTP3 Route handle.


 For any reconfiguration with a MTP3 (TB640_MSG_ID_SS7_MTP3_OP_xyz_SET_PARAMS) message, all reconfigurable parameters must be filled appropriately even if the intention is to modify a single parameter.

 When a reconfiguration with a MTP3 set params is issued, the effect of the set params may not be observed immediately. For example, if the link congestion thresholds (i.e. un32Prio0MsgQueueLen in the TB640_SS7_MTP3_LINK_CFG structure) are modified with a TB640_MSG_ID_SS7_MTP3_OP_LINK_SET_PARAMS message, the current congestion status link may not change.

 The step 2 can be done 1 to maximum of Userpart.

 The step 3 can be done 1 to maximum of Linkset.

 The step 4 can be done 1 to maximum of Links per Linkset.

 The step 5 must be done 1 time for the “self-route” and 1 to maximum of Routes.

5.2.1.1 General Configuration

The TB640_MSG_ID_SS7_MTP3_OP_ALLOC (request/response) message is used to initialize the general parameters of the MTP3 layer.

The **request** part of the message TB640_MSG_ID_SS7_MTP3_OP_ALLOC contains the field:

...

```
TB640_SS7_HANDLE      hLayer; /* Contains the layer handle from system manager module */
TB640_SS7_MTP3_CFG    Cfg;      /* Contains the configuration of the MTP3 layer */
...
```

Structure contains the general configuration parameters for MTP3 layer:

```
typedef struct _TB640_SS7_MTP3_CFG
{
    TBX_UINT32                un32StructVersion;
    TB640_SS7_MTP3_SIGNALING_POINT_TYPE SignalingPointType;
    TB640_SS7_POINT_CODE      DefaultOpc1;
    TB640_SS7_POINT_CODE      DefaultOpc2;
    TBX_BOOL                   fSSFValidation;
    TB640_SS7_MTP3_RESTART_PROCEDURE RestartProcedure;
    TBX_BOOL                   fTransferRestrictRequired;
    TBX_UINT32                 un32T15Timer;
    TBX_UINT32                 un32T16Timer;
    TBX_UINT32                 un32T18Timer;
    TBX_UINT32                 un32T19Timer;
    TBX_UINT32                 un32T21Timer;
    TBX_UINT32                 un32T26Timer;

    TBX_UINT8                  aun8Padding0 [4];
} TB640_SS7_MTP3_CFG, *PTB640_SS7_MTP3_CFG;
```

General explanation of the parameters of configuration:

- The type of signaling parameter (*SignalingPointType*) specifies whether the node being configured acts as a Signaling Point (SP) or as a Signaling Transfer Point (STP). This parameter is reconfigurable. Allowable values:

Table 18 - MTP3 signaling point types

Signaling Point Type	Description
TB640_SS7_MTP3_SIGNALING_POINT_TYPE_SP	MTP3 layer is used as an SP (signaling point) in the SS7 network
TB640_SS7_MTP3_SIGNALING_POINT_TYPE_STP	MTP3 layer is used as an STP (signaling transfer point) in the SS7


- The default point code 1 parameter (*DefaultOpc1*) specifies the default point code of the node being configured for ITU, TTC, and NTT networks. Used as the OPC when the user part does not specify the OPC in the data request. This parameter is not reconfigurable.
- The default point code 2 parameter (*DefaultOpc2*) specifies the default point code for ANSI or CHINA version. This is used as the OPC for ANSI or CHINA networks when the userpart does not specify the OPC in the data request. This parameter is not reconfigurable.
- The SSF validation required flag (*fSSFValidation*). If this flag is TBX_TRUE, the SSF of any message received on a link or from the userpart by MTP3 is checked against the SSF configured for that link or userpart respectively. For example, if a link is configured

to carry international traffic and it receives messages with SSF other than SSF international traffic, the message is discarded. This parameter is reconfigurable.


- The restarting procedure required field (*RestartProcedure*). If this field is not TB640_SS7_MTP3_RESTART_PROCEDURE_NONE, the MTP3 restart procedure is started by MTP3 as soon as the first link becomes in-service. This parameter is reconfigurable. Allowable values:

Table 19 - MTP3 restart procedures

Restart Procedure	Description
TB640_SS7_MTP3_RESTART_PROCEDURE_NONE	Restart procedure disabled.
TB640_SS7_MTP3_RESTART_PROCEDURE_ITU88	ITU88 compliant restart procedure.
TB640_SS7_MTP3_RESTART_PROCEDURE_ITU92	ITU92 compliant restart procedure.
TB640_SS7_MTP3_RESTART_PROCEDURE_ANSI	ANSI compliant restart procedure.


 TTC, NTT and CHINA do not support MTP3 restart procedure. Therefore, this field should be configured with TB640_SS7_MTP3_RESTART_PROCEDURE_NONE.

- The transfer restricts required flag (*fTransferRestrictRequired*). Transfer restrict is a route management procedure of MTP3. It is a national option that can be enabled or disabled through this flag. This field is reconfigurable. Allowable values: TBX_TRUE or TBX_FALSE.

 TTC and NTT do **not** support transfer restrict procedure. Therefore, for TTC and NTT, this field is configured as **TBX_FALSE**.

- The general timers configuration parameters. All timers are reconfigurable. The timers has the following definitions:

un32T15Timer: Waiting to start signaling route set congestion test. Typical value is 2000 milliseconds (2 seconds).
un32T16Timer: Waiting for route set congestion status update. Typical value is 1400 milliseconds (1.4 seconds).
un32T18Timer: Timer within a signaling point whose MTP restarts for supervising link and linkset activation as well as the receipt of routing information. The value is implementation and network dependent. Typical value is 11000 milliseconds (11 seconds) for ANSI and 30000 milliseconds (30 seconds) for ITU.
un32T19Timer: Waiting to receive all traffic restart timer. Typical value is 67000 milliseconds (67 seconds).
un32T21Timer: Waiting to restart traffic routed through adjacent SP timer. Typical value is 63000 milliseconds (63 seconds).
un32T26Timer: Waiting to repeat traffic restart waiting msg timer. Typical value is 12000 milliseconds (12 seconds).

 TTC, NTT and CHINA do not support MTP3 restart procedure. Therefore, the timers related to this procedure need not be configured for these

networks. For TTC and NTT, the route set congestion test is optional, in which case timers **T15** and **T16** need not be configured.

The **response** part of the message TB640_MSG_ID_SS7_MTP3_OP_ALLOC contains the field:

```
...
TB640_SS7_MTP3_HANDLE          hMtp3; /* The handle of the initialized MTP3 layer */
...
```

5.2.1.2 Userpart Configuration

The TB640_MSG_ID_SS7_MTP3_OP_USERPART_ALLOC (request/response) message is used to initialize the userpart parameters of the MTP3 layer.

The **request** part of the message TB640_MSG_ID_SS7_MTP3_OP_USERPART_ALLOC contains the field:

```
...
TB640_SS7_MTP3_HANDLE          hMtp3; /* Handle to the MTP3 general layer */
TB640_SS7_MTP3_USERPART_CFG    Cfg; /* Contains the configuration of the MTP3
                                     * Userpart instance */
...
```

Structure contains the userpart configuration parameters for MTP3 layer:

```
typedef struct _TB640_SS7_MTP3_USERPART_CFG
{
    TB640_SS7_UID                UidMtp3Userpart;
    TB640_SS7_MTP3_CONNECTION_MODE ConnectionMode;
    TB640_SS7_SUBSERVICE_FIELD_TYPE SsfType;
    TB640_SS7_MTP3_LINK_TYPE      LinkType;
    TB640_SS7_MTP3_DPC_LENGTH      DpcLength;
    TBX_UINT32                     un32SwitchId;
} TB640_SS7_MTP3_USERPART_CFG, *PTB640_SS7_MTP3_USERPART_CFG;
```

General explanation of the parameters of configuration:

- Mtp3 user part unique ID (*UidMtp3Userpart*) is used to configure identical MTP3 userpart in each MTP3 module and to connect ISUP network. This field is not reconfigurable.
- The connection mode parameter (*ConnectionMode*) specifies the different modes of connection for an MTP3 userpart. This field is not reconfigurable. Allowable values:

Table 20 - MTP3 connection mode


Connection Mode	Description
TB640_SS7_MTP3_CONNECTION_MODE_SCCP	MPT3 userpart is used in conjunction with an above local SCCP layer and cannot be controlled by the host application.
TB640_SS7_MTP3_CONNECTION_MODE_SCCP_WITH_ALARMS	MPT3 userpart is used in conjunction with an above local SCCP layer and cannot be


	controlled by the host application but the host application will receive ALARM notifications.
TB640_SS7_MTP3_CONNECTION_MODE_ISUP	MPT3 userpart is used in conjunction with an above local ISUP layer and cannot be controlled by the host application.
TB640_SS7_MTP3_CONNECTION_MODE_ISUP_WITH_ALARMS	MPT3 userpart is used in conjunction with an above local ISUP layer and cannot be controlled by the host application but the host application will receive ALARM notifications.
TB640_SS7_MTP3_CONNECTION_MODE_HOST_SCCP ⁴	MTP3 userpart is used as a standalone and communicates with remote SCCP layer (host application).
TB640_SS7_MTP3_CONNECTION_MODE_HOST_TUP ⁵	MTP3 userpart is used as a standalone and communicates with remote TUP layer (host application).
TB640_SS7_MTP3_CONNECTION_MODE_HOST_ISUP	MTP3 userpart is used as a standalone and communicates with remote ISUP layer (host application).

- The sub-service field (*SsfType*) specifies the type of network. If *fSSFValidation* is activated in the general MTP3 configuration, all userpart messages will be compared to this sub-service field type. All non-matching messages will be discarded. This field is not reconfigurable. Allowable values:

Table 21 - SSF values

Sub-service field	Description
TB640_SS7_SUBSERVICE_FIELD_TYPE_INTERNATIONAL	Messages sent by the layer will concern international traffic.
TB640_SS7_SUBSERVICE_FIELD_TYPE_NATIONAL	Messages sent by the layer will concern national traffic.
TB640_SS7_SUBSERVICE_FIELD_TYPE_NAT_RESERVED	Reserved for national use.

 For all national networks (including TTC and NTT), *SsfType* is configured with TB640_SS7_SUBSERVICE_FIELD_TYPE_NATIONAL. For international networks, the *SsfType* is configured with TB640_SS7_SUBSERVICE_FIELD_TYPE_INTERNATIONAL.

 For international networks you must have the same values for *un32Prio1MsgQueueLen* to *un32Prio3MsgQueueLen* in a MTP3 Link Configuration.

- The link type field (*LinkType*), specifies the MTP3 protocol variant required to provide service to the userpart connected through this upper SAP (Service Acces Point). This field is not reconfigurable. Allowable values:

Table 22 - MTP3 link types

Link Type	Description
TB640_SS7_MTP3_LINK_TYPE_ANSI	Used for ANSI88 and ANSI92.
TB640_SS7_MTP3_LINK_TYPE_ITU	Used for ITU88, ITU92, ITU97, Q767, SINGAPORE

⁴ Not available yet.

	and ETSI.
TB640_SS7_MTP3_LINK_TYPE_CHINA	Used for CHINA.
TB640_SS7_MTP3_LINK_TYPE_ANSI96	Used for ANSI96 and TELCORDIA.
TB640_SS7_MTP3_LINK_TYPE_TTC	Link type not available yet.
TB640_SS7_MTP3_LINK_TYPE_NTT	Link type not available yet.

- The DPC length field (*DpcLength*), specifies the length of the DPC for messages communicated through this SAP. This field is not reconfigurable. Allowable values: see **Table 3 - DPC Length**.
- The switch ID parameter (*un32SwitchId*) specifies a value to regroup linksets and user part together. This field is not reconfigurable. Allowable values are from 10 to 255.

The **response** part of the message TB640_MSG_ID_SS7_MTP3_OP_USERPART_ALLOC contains the field:

```

...
TB640_SS7_MTP3_USERPART_HANDLE      hMtp3Userpart;          /* The handle of the instance of
                               * the MTP3 userpart. */
...

```

5.2.1.3 Linkset Configuration

The TB640_MSG_ID_SS7_MTP3_OP_LINKSET_ALLOC (request/response) message is used to initialize an instance of linkset (association of MTP3 links) of the MTP3 layer. A linkset is used to logically combine multiple MTP3 links going to the same DPC into a single entity.

The **request** part of the message TB640_MSG_ID_SS7_MTP3_OP_LINKSET_ALLOC contains the field:

```

...
TB640_SS7_MTP3_HANDLE              hMtp3; /* Handle to the MTP3 general layer */
TB640_SS7_MTP3_LINKSET_CFG         Cfg; /* Contains the configuration of the MTP3 linkset
                               * instance */
...

```

Structure contains the linkset configuration parameters for MTP3 layer:

```

typedef struct _TB640_SS7_MTP3_LINKSET_CFG
{
    TBX_UINT32                un32StructVersion;
    TBX_UINT32                un32SwitchId;
    TB640_SS7_POINT_CODE     Opc;
    TB640_SS7_POINT_CODE     AdjDpc;
    TBX_UINT32                un32NbActiveLinks;


    TBX_UINT8                 aun8Padding0 [4];
} TB640_SS7_MTP3_LINKSET_CFG, *PTB640_SS7_MTP3_LINKSET_CFG;

```

General explanation of the parameters of configuration:

- The switch ID parameter (*un32SwitchId*) specifies a value to regroup linksets and user part together. This field is not reconfigurable. Allowable values are from 10 to 255.

- The originating point code parameter (*Opc*) specifies the origin point code (OPC) of all the links of the linkset. If multiple OPC capability is needed, each OPC can be configured differently per linkset. Otherwise, the OPC is the same as *DefaultOpc1* or *DefaultOpc2* in the general configuration. This field is not reconfigurable.
- The adjacent destination point code parameter (*AdjDpc*) specifies the point code of the adjacent node where the link set terminates. This field is not reconfigurable.
- The number of active links parameter (*un32NbActiveLinks*), specifies the number of the active links required in a linkset. The number of traffic carrying links may be less than the number of active links in a linkset (inactive links, inhibited links, and blocked links are some examples). When an active link goes OOS or becomes unavailable for traffic (due to inhibition or blocking), one of the remaining inactive links is made active. This field is reconfigurable for all the variants **except for NTT**.

 The value of *un32NbActiveLinks* should be greater than or equal to the number of traffic carrying links in that linkset under normal circumstances (no failures) — that is, the number of priority **0** links belonging to this linkset. Violation of this rule may result in non-deterministic mapping of SLSs to links.

The **response** part of the message TB640_MSG_ID_SS7_MTP3_OP_LINKSET_ALLOC contains the field:

```
...
TB640_SS7_MTP3_LINKSET_HANDLE          hMtp3Linkset;          /* The handle of the instance of
* the MTP3 linkset. */
...
```

5.2.1.4 Link Configuration

The TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC (request/response) message is used to initialize an instance of link (also called a data link service access point) of the MTP3 layer. A MTP3 link is used to connect with a MTP2 link, they are connected one to one.

The **request** part of the message TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC contains the field:

```
...
TB640_SS7_MTP3_HANDLE          hMtp3; /* Handle to the MTP3 general layer */
TB640_SS7_MTP3_LINK_CFG        Cfg; /* Contains the configuration of the MTP3 link
* instance */
...
```

Structure contains the link configuration parameters for MTP3 layer:

```
typedef struct _TB640_SS7_MTP3_LINK_CFG
{
    TB640_SS7_UID                UidMtp3Link;
    TB640_SS7_MTP3_LINKSET_HANDLE hLinkset;
    TB640_SS7_UID                UidMtp2Link;
    TB640_SS7_MTP3_PRIORITY      LinkPriority;
    TBX_UINT32                   un32MaxFrameLength;
    TB640_SS7_MTP3_MESSAGE_PRIORITY MessagePriority;
}
```

```

TBX_BOOL                fIsA_C_link;
TBX_UINT32              un32Prio0MsgQueueLen;
TBX_UINT32              un32Prio1MsgQueueLen;
TBX_UINT32              un32Prio2MsgQueueLen;
TBX_UINT32              un32Prio3MsgQueueLen;
TB640_SS7_MTP3_PRIORITY DiscardPriority;
TBX_UINT32              un32MaxSLTMRetry;
TBX_UINT8               un8LinkTestSLC;
TBX_UINT8               un8TestCharacter [ TB640_SS7_MTP3_LINK_TEST_PATTERN_MAX_LENGTH ];
TBX_UINT32              un32TestMsgLen;
TBX_UINT32              un32T1Timer;
TBX_UINT32              un32T2Timer;
TBX_UINT32              un32T3Timer;
TBX_UINT32              un32T4Timer;
TBX_UINT32              un32T5Timer;
TBX_UINT32              un32T7Timer;
TBX_UINT32              un32T12Timer;
TBX_UINT32              un32T13Timer;
TBX_UINT32              un32T14Timer;
TBX_UINT32              un32T17Timer;
TBX_UINT32              un32T22Timer;
TBX_UINT32              un32T23Timer;
TBX_UINT32              un32BsnRequestedTimer;
TBX_UINT32              un32SltTimer;
TBX_UINT32              un32ConnectingTimer;
TBX_UINT32              un32PeriodicSigLinkTstTimer;
TBX_UINT32              un32FalseLinkCongDetectTimer;
TBX_UINT32              un32ProbationLinkOscillationTimer;
TBX_UINT32              un32SuspensionLinkOscillationTimer;
TBX_UINT32              un32LinkReferralCraftTimer;
TBX_UINT32              un32FlowControlRequestTimer;
TBX_UINT32              un32BindConfirmWaitTimer;
TBX_BOOL                fFlushContinueFlags;

TBX_UINT8               aun8Padding0 [4];
} TB640_SS7_MTP3_LINK_CFG, *PTB640_SS7_MTP3_LINK_CFG;

```

General explanation of the parameters of configuration:


- The unique MTP3 link identifier parameter (*UidMtp3Link*) is used to configure identical MTP3 link in each MTP3 module. This field is not reconfigurable.
- The linkset handle parameter (*hLinkset*) specifies the handle of linkset to which this link belongs. This field is not reconfigurable.
- The unique MTP2 link identifier parameter (*UidMtp2Link*) specifies the UID of the MTP2 link to which this link belongs. This field is not reconfigurable.
- The link priority parameter (*LinkPriority*) specifies the priority of the link within a linkset. The highest priority is **0**. There must be at least one link in a linkset with priority **0**. If there are links with different priorities in a linkset, the priorities of the links must be contiguous. For example, there can be no existing links with priority **0**, **1**, and **3** in a linkset (priority **2** missing).

At any given time, only the in-service links with the highest priority are used to carry traffic on that linkset based on the *un32NbActiveLinks* in linkset configuration. For example, if there are five links in a linkset, with two at priority 0, two at priority 1, and one at priority 2 and the value of *un32NbActiveLinks* in the linkset configuration is 5, then the two in-service links with priority 0 are used to carry the traffic. If one priority 0 link goes Out-Of-Service (OOS), one of the in service priority 1 links will be used for traffic. This field is not reconfigurable. Allowable values:

Table 23 - MTP3 link priorities

Link Priority	Description
TB640_SS7_MTP3_PRIORITY_HIGHEST	
TB640_SS7_MTP3_PRIORITY_0	Priority 0, same as highest.
TB640_SS7_MTP3_PRIORITY_1	Priority 1.
TB640_SS7_MTP3_PRIORITY_2	Priority 2.
TB640_SS7_MTP3_PRIORITY_03	Priority 3, same as lowest.
TB640_SS7_MTP3_PRIORITY_LOWEST	


- The maximum frame length parameter (*un32MaxFrameLength*), specifies the maximum frame length for a message signal unit. This field is reconfigurable.

 **MUST** be set to 272 bytes.


- The message priority parameter (*MessagePriority*) specifies the priority at which MTP3 management messages are sent, in case the network allows priorities to be attached with messages (as in ANSI, for example). If the network does not allow priorities to be attached with messages, this is configured to 0. This field is reconfigurable. Allowable values:

Table 24 - MTP3 message priorities


Message Priority	Description
TB640_SS7_MTP3_MESSAGE_PRIORITY_NOT_USED	Used for TTC and NTT variants.
TB640_SS7_MTP3_MESSAGE_PRIORITY_HIGHEST	For international networks or networks without multiple congestion states.
TB640_SS7_MTP3_MESSAGE_PRIORITY_0	Priority 0, same as highest.
TB640_SS7_MTP3_MESSAGE_PRIORITY_1	Priority 1.
TB640_SS7_MTP3_MESSAGE_PRIORITY_2	Priority 2.
TB640_SS7_MTP3_MESSAGE_PRIORITY_3	Priority 3, same as lowest.
TB640_SS7_MTP3_MESSAGE_PRIORITY_LOWEST	

 This field does not have significance for TTC and NTT. For international networks or networks without multiple congestion states, *MessagePriority* should be configured with TB640_SS7_MTP3_MESSAGE_PRIORITY_NOT_USED.


- The c-link flag (*flsA_C_link*), indicates whether the link is a C link (links between the mated STPs are C links). This field is specific to ANSI. This field is reconfigurable. Allowable values: **TBX_TRUE** or **TBX_FALSE**.

 This information is relevant to STPs only. In ANSI networks, the SLS should not be rotated for the data transferred between the C links, and hence need to be configured appropriately.

- The priority 0, 1, 2 and 3 message queue length parameter (*un32Prio0MsgQueueLen*, *un32Prio1MsgQueueLen*, *un32Prio2MsgQueueLen* and *un32Prio3MsgQueueLen*), are used to implement multiple levels of congestion on this link. This specifies four different levels of thresholds and does not represent four different queues for the SAP. When MTP3 starts queuing up messages (for example, due to congestion on a link), the messages are queued up in the link. As the queue length grows, different levels of congestion are reached depending upon the values configured for *un32PrioXMsgQueueLen*. The same value of *un32Prio1MsgQueueLen* to *un32Prio3MsgQueueLen* is acceptable, but will result in only one level of congestion. These values are configured based upon the memory available to queue the messages. These fields are reconfigurable.

 It is essential for international networks to have same values for *un32Prio1MsgQueueLen* to *un32Prio3MsgQueueLen*.

- The discard priority parameter (*DiscardPriority*). In national networks supporting multiple levels of congestion, if a message is received by MTP3 with a priority less than the current congestion priority of the link (in case the link is congested), then the message is dropped. If it is required to override this feature, this parameter can be configured so that before MTP3 drops a message because of congestion, it checks the message priority with *DiscardPriority*. If the message priority is less than the discard priority, the message is dropped. This field is reconfigurable. Allowable values: see **Table 23 - MTP3 link priorities**.
- The maximum times to retry SLTM (*un32MaxSLTMRetry*). When a link is aligned at MTP3, an SLTM (Signaling Link Test Message) message is periodically sent to the peer MTP3 to determine the state of the link and waits for SLTA (Signaling Link Test Acknowledgment) (MTP3 receives SLTA for the SLTM message it sent after level 2 is aligned for an Out-Of-Service (OOS) link). If SLTA is not received, then it repeats the SLTM message *un32MaxSLTMRetry* before declaring the link OOS. This field is reconfigurable.


 TTC and NTT do not support Signaling Link Test using SLTM/SLTA. Therefore, this field is not applicable for TTC and NTT.

- The link selection code for link test (*un8LinkTestSLC*). Each link between Trillium's node and an adjacent node is assigned a unique *un8LinkTestSLC* between **0** and **15**. There can be a maximum of 16 links between two adjacent point codes. This value must be agreed upon by both the ends for each link. This field is reconfigurable.

- ✍ For TTC and NTT, at a signaling point, three MSB bits represent the SLC, and the LSB represents the plane information (**0** = PLANE A and **1** = PLANE B). Also note that since the maximum *un8LinkTestSLC* value possible is 15, there can be only 16 links possible in any linkset terminating on a particular DPC.
- The link test character array (*un8TestCharacter []*), specifies the character pattern for the SLTM message. This field is reconfigurable.
 - ✍ TTC and NTT do not support signaling link test using SLTM/SLTA. Therefore, the field *un8TestCharacter []* is not configured.
- The link test message length (*un32TestMsgLen*). When MTP3 gets a connection confirm from layer 2 (as part of a link activation procedure), it sends an SLTM message to align the link at MTP3. In the SLTM message, it sends a character pattern, and expects the same pattern to come back in an SLTA message from the peer. *un32TestMsgLen* specifies the length of the character pattern. This field is reconfigurable.
 - ✍ TTC and NTT do not support signaling link test using SLTM/SLTA. Therefore, the field *un32TestMsgLen* is not configured.
- The link timers configuration parameters. All timers are reconfigurable.

The timers have the following definitions:

<i>un32T1Timer:</i>	Delay to avoid mis-sequencing on changeover. Typical value is 800 milliseconds (0.8 second).
<i>un32T2Timer:</i>	Waiting for changeover ack. Typical value is 1400 milliseconds (1.4 seconds).
<i>un32T3Timer:</i>	Delay to avoid mis-sequencing on changeback. Typical value is 800 milliseconds (0.8 second).
<i>un32T4Timer:</i>	Waiting for changeback ack (first). Typical value is 800 milliseconds (0.8 second).
<i>un32T5Timer:</i>	Waiting for changeback ack (second). Typical value is 800 milliseconds (0.8 second).
<i>un32T7Timer:</i>	Waiting for link connection ack. Typical value is 1500 milliseconds (1.5 seconds).
<i>un32T12Timer:</i>	Waiting for uninhibit ack. Typical value is 1150 milliseconds (1.15 seconds).
<i>un32T13Timer:</i>	Waiting for force uninhibit. Typical value is 1150 milliseconds (1.15 seconds).
<i>un32T14Timer:</i>	Waiting for inhibition ack. Typical value is 2500 milliseconds (2.5 seconds).
<i>un32T17Timer:</i>	Delay to avoid oscillation of initial alignment failure. Typical value is 1150 milliseconds (1.15 seconds).
<i>un32T22Timer:</i>	Local inhibit test timer. Typical value is 30000 milliseconds (30 seconds) for ANSI and 300000 milliseconds (5 minutes) for ITU.
<i>un32T23Timer:</i>	Remote inhibit test timer. Typical value is 30000 milliseconds (30 seconds) for ANSI and 300000 milliseconds (5 minutes) for ITU.

-  TTC and NTT do not support inhibition and Signaling Link Test. Therefore, the timers related to these procedures are not configured. Also, timer **T5** and **T17** are **not** configured, as they have been removed from TTC and NTT specifications.

The non-standard timers has the following definitions:

- un32BsnRequestedTimer:* This is a Trillium-specific timer. As part of the changeover procedure on a link, MTP3 sends a status request to layer 2 to return the BSN for that link. The value of this timer depends on the customer implementation. For example, if layer 2 and MTP3 are on separate boards, this value is more than when MTP3 and layer 2 exist together on the same board. Typical value is 5000 milliseconds (5 seconds).
- un32SlrTimer:* This is the same as timer T1 in Q.707. This is also the same as T10 in JT Q.707 (TTC Japan). Typical value is 4000 milliseconds (4 seconds).
- un32ConnectingTimer:* This is a Trillium-specific timer to take care of loss of connect request or connect confirm. To activate a link, MTP3 sends a connect request to layer 2 and starts this timer *un32ConnectingTimer*. If layer 2 does not send either a connection confirms or disconnects indication, and *un32ConnectingTimer* expires, MTP3 repeats the connection request and restarts *un32ConnectingTimer*. *un32ConnectingTimer* is used by MTP3 to periodically send the connect request to layer 2, in case layer 2 is not responding. Its typical value is between 120000 milliseconds (2 minutes) to 600000 milliseconds (10 minutes).
- un32PeriodicSigLinkTstTimer:* This is the same as timer T2 in Q.707. Typical value is 30000 milliseconds (30 seconds).
- un32FalseLinkCongDetectTimer:* This is the same as timer T31 in ANSI'96. Used for ANSI 88, 92 and 96. Typical value is 10000 milliseconds (10 seconds).
- un32ProbationLinkOscillationTimer:* This is the same as timer T33 in ANSI'96. Used for ANSI 88, 92 and 96. Typical value is 60000 milliseconds (1 minute).
- un32SuspensionLinkOscillationTime:* This is the same as timer T34 in ANSI'96. Used for ANSI 88, 92 and 96. Typical value is 5000 milliseconds (5 seconds).
- un32LinkReferralCraftTimer:* This is the same as timer T19 in ANSI'96. Used for ANSI 88, 92 and 96. Typical value is 480000 milliseconds (8 minutes).
- un32FlowControlRequestTimer:* This is a proprietary Trillium timer. When layer 2 sends a flow control indication to MTP3 indicating onset of flow control, MTP3 sends a status request to layer 2 periodically at the time-out of the *un32FlowControlRequestTimer* timer. The recommended value for timer *un32FlowControlRequestTimer* is 30000 milliseconds (30 seconds).
- un32BindConfirmWaitTimer:* This is a proprietary Trillium timer. When MTP3 sends a bind request to layer 2, it starts this timer to wait for the bind confirm. The recommended value for timer *un32BindConfirmWaitTimer* is from 2000 milliseconds (2 seconds) to 5000 milliseconds (5 seconds).

- The flush continue flag parameter (*fFlushContinueFlags*), specifies whether FLUSH BUFFER and CONTINUE requests are required to be sent to layer 2 on processor outage recovery. It is **TBX_FALSE** for ITU88, CHINA, TTC and NTT variants of MTP3. This field is reconfigurable. Allowable values: **TBX_TRUE** or **TBX_FALSE**.

The **response** part of the message TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC contains the field:

```
...
TB640_SS7_MTP3_LINK_HANDLE    hMtp3Link;    /* The handle of the instance of the MTP3 link */
...
```

You can add a new link (with TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC) to a linkset and changed the number active links of the linkset (with B640_MSG_ID_SS7_MTP3_OP_LINKSET_SET_PARAMS). So, you DON'T need to clear and redo the linkset.



If you remove a link from a linkset you must verify if the *LinkPriority* of the remaining links in the linkset do not become discontinuous after deletion of this link.

5.2.1.5 Route Configuration

The TB640_MSG_ID_SS7_MTP3_OP_ROUTE_ALLOC (request/response) message is used to create a route to a specific DPC.

The **request** part of the message TB640_MSG_ID_SS7_MTP3_OP_ROUTE_ALLOC contains the field:

```
...
TB640_SS7_MTP3_HANDLE        hMtp3;    /* Handle to the MTP3 general layer */
TB640_SS7_MTP3_ROUTE_CFG    Cfg;    /* Contains the configuration of the MTP3 route */
...
```

Structure contains the route configuration parameters for MTP3 layer:

```
typedef struct _TB640_SS7_MTP3_ROUTE_CFG
{
    TBX_UINT32                un32StructVersion;
    TBX_UINT8                 aun8Padding0 [4];

    TB640_SS7_POINT_CODE     Dpc;
    TB640_SS7_MTP3_SIGNALING_POINT_TYPE SignalingPointType;
    TB640_SS7_MTP3_ROUTE_DIRECTION Direction;
    TBX_UINT32                un32T6Timer;
    TBX_UINT32                un32T8Timer;
    TBX_UINT32                un32T10Timer;
    TBX_UINT32                un32T11Timer;
    TBX_UINT32                un32T19Timer;
    TBX_UINT32                un32T21Timer;
    TBX_UINT32                un32T18Timer;
    TBX_UINT32                un32T25Timer;
    TBX_UINT32                un32TcTimer;
    TBX_BOOL                  fRouteToAdjacentSP;
    TBX_BOOL                  fBroadcast;
}
```


```

TB640_SS7_MTP3_RESTART_PROCEDURE      RestartProcedure;
TB640_SS7_MTP3_SLS_RANGE              SlsRange;
TB640_SS7_MTP3_STP_SLS_SELECTOR       StpSlsLinksetSelector;
TBX_BOOL                               fSupportsMultiMsgPriority;
TBX_BOOL                               fSupportRouteSetCongestionTest;
TBX_UINT32                             un32NbLinkset;
TBX_UINT8                              aun8Padding1 [4];
TB640_SS7_MTP3_LINKSET_HANDLE         ahLinkset [TB640_SS7_MTP3_MAX_LINKSETS_PER_ROUTE];
TB640_SS7_MTP3_PRIORITY               aLinksetPriority [TB640_SS7_MTP3_MAX_LINKSETS_PER_ROUTE];
} TB640_SS7_MTP3_ROUTE_CFG, *PTB640_SS7_MTP3_ROUTE_CFG;

```

General explanation of the parameters of configuration:


- The destination point code parameter (*Dpc*) specifies the point code of the node where the route is terminating. This field is not reconfigurable.

 **Default routing:** Note that destination point code 0.0.0 is reserved for default routing. When no explicit route exists for a destination point code and default route 0.0.0 is present, the message is passed to MTP 2 for delivery in place of being dropped. Default routing can be used for routing of outbound messages only (direction parameter (*Direction*) equal to TB640_SS7_MTP3_ROUTE_DIRECTION_DOWN).

- The signaling point type parameter (*SignalingPointType*) specifies whether the point code under configuration is an SP or an STP. This field is reconfigurable. Allowable values: see **Table 18 - MTP3 signaling point types**.
- The direction parameter (*Direction*) specifies whether the point code under configuration is self-point code or remote SS7 point code. A route configuration exists for each point code, including Trillium's; when a route for Trillium's own point code is configured, TB640_SS7_MTP3_ROUTE_DIRECTION_UP is used. For remote point codes in the SS7 network TB640_SS7_MTP3_ROUTE_DIRECTION_DOWN is used. This field is not reconfigurable. Allowable values:

Table 25 - MTP3 directions

Direction	Description
TB640_SS7_MTP3_ROUTE_DIRECTION_UP	Local DPC.
TB640_SS7_MTP3_ROUTE_DIRECTION_DOWN	Remote DPC.

 If multiple OPC capability is required, the number of routes configured with direction as TB640_SS7_MTP3_ROUTE_DIRECTION_UP should be equal to the number of OPCs representing the Trillium node/Gateway.

- The route timers configuration parameters. All timers are reconfigurable.

The timers have the following definitions:

un32T6Timer: Delay to avoid missequencing during controlled rerouting. Typical value is 800 milliseconds (0.8 seconds).

- un32T8Timer:* Transfer prohibited inhibition timer. Typical value is 800 milliseconds (0.8 seconds).
- un32T10Timer:* Route set test timer. Typical value is 30000 milliseconds (30 seconds).
- un32T11Timer:* Transfer restrict timer. Typical value is 30000 milliseconds (30 seconds).
- un32T19Timer:* TRA (Traffic Restart Allowed) timer. Not required for ITU 88. Same as T29 of ANSI. Typical value is 67000 milliseconds (67 seconds).
- un32T21Timer:* Waiting to restart traffic routed through adjacent SP.
- un32T18Timer:* Transfer restricted inhibition. Used for ANSI 88, 92 and 96 only. Typical value is 11000 milliseconds (11 seconds).
- un32T25Timer:* Waiting for traffic restart allowed message" timer. Used for ANSI 88, 92 and 96 only. Typical value is 30000 milliseconds (30 seconds).
- un32TcTimer:* Delay timer for remote congestion abatement. Used for TTC and NTT only. Typical value is 500 milliseconds (0.5 second).

- The flag indicating route to adjacent SP (*fRouteToAdjacentSP*) specifies whether the point code being configured is adjacent to Trillium node. This field is not reconfigurable. Allowable values: **TBX_TRUE** or **TBX_FALSE**.

 Must be **TBX_FALSE** if *Direction* is UP.

- The broadcast required flag (*fBroadcast*), specifies whether the route management messages for this destination are sent in a broadcast manner to other destinations in the network. This field is reconfigurable. Allowable values: **TBX_TRUE** or **TBX_FALSE**.
- The restarting procedure required field (*RestartProcedure*). Configured only for the point codes adjacent to Trillium's node. When MTP3 restores connectivity with an adjacent point code, it starts the adjacent restart procedure as recommended by this field. This field is reconfigurable. Allowable values: see **Table 19 - MTP3 restart procedures**.

 TTC and NTT do not support MTP3 restart procedure.

- The number of SLSs for this point code (*SlsRange*). The number of SLSs depends on the network for which the route is configured. This field is not reconfigurable. Allowable values:

Table 26 - MTP3 SLS ranges

SLS ranges	Description
TB640_SS7_MTP3_SLS_RANGE_ITU	ITU, TTC and CHINA.
TB640_SS7_MTP3_SLS_RANGE_ANSI_5BITS	For networks which use 5 bits SLS.
TB640_SS7_MTP3_SLS_RANGE_ANSI_8BITS	For networks which use 8 bits SLS.
TB640_SS7_MTP3_SLS_RANGE_NTT	NTT.

- The linkset selector bit(s) in SLS (*StpSlsLinksetSelector*). This field need to be configured only when the own point code is an STP. At an STP, for any incoming message which needs to be rerouted to the DPC indicated in the message, MTP3 chooses a linkset based on these bit(s) of SLS in the received message. This field is reconfigurable. Allowable values are:

Table 27 - MTP3 SLS selectors

Linkset Selector
TB640_SS7_MTP3_STP_SLS_SELECTOR_BIT1
TB640_SS7_MTP3_STP_SLS_SELECTOR_BIT2
TB640_SS7_MTP3_STP_SLS_SELECTOR_BIT3
TB640_SS7_MTP3_STP_SLS_SELECTOR_BIT4



If more than one bit needs to be used to choose a linkset, the above values can be **ORed**. For example, if at an ITU STP, the 2 LSBs of SLS need to be used for selecting a linkset, the **lsetSel** value should be configured as:

TB640_SS7_MTP3_STP_SLS_SELECTOR_BIT1 | TB640_SS7_MTP3_STP_SLS_SELECTOR_BIT1

The value should be chosen based on the incoming SLSs such that a fair distribution of load happens among the linksets towards the given destination.

Refer to Section A.5 in ITU Q.705 specification, "Explanatory Note from the Implementors Forum for Clarification of Load Sharing," for details on load sharing at an STP.

- ✍ For an NTT STP, this field should always be configured with TB640_SS7_MTP3_STP_SLS_SELECTOR_BIT2. For ANSI STPs, the last bit is used to choose a linkset (as per the specifications), hence this field should be configured to TB640_SS7_MTP3_STP_SLS_SELECTOR_BIT1.
- ✍ Reconfiguring this field in a live system while the point code is available may result in mis-sequencing.
- The multi message priority flag (*fSupportsMultiMsgPriority*), indicates whether multiple congestion priorities are supported for the destination or not. Allowable values are **TBX_TRUE** or **TBX_FALSE**. If the value is set to **TBX_FALSE**, all the outgoing messages will be encoded with priority **0** irrespective of the priority received from the user part. This field is reconfigurable. If this field is **TBX_FALSE**, no messages are dropped due to link congestion. Also if this field is **TBX_FALSE**, the route set congestion cannot be performed and hence the *fSupportRouteSetCongestionTest* flag should be set to **TBX_FALSE**.
 - ✍ Should be set to **TBX_TRUE** for ANSI networks and all national variants of ITU which require the support for multiple congestion states.
 - ✍ Setting this option to **TBX_TRUE** with remote equipment NOT supporting it will result in some SS7 primitives to be ignored by the remote switches. For example, outgoing IAMs will work (as its priority is always 0) but REL and RLC will be ignored. Thus, be very careful when activating this configuration parameter.

- The route set congestion parameter (*fSupportRouteSetCongestionTest*), indicates whether the route-set-congestion test needs to be performed or not on reception of a transfer controlled message. Allowable values are `TBX_TRUE` or `TBX_FALSE`. This field needs to be configured only for the networks which support multiple congestion priorities (ITU with national option, CHINA, TTC, NTT, and ANSI). This field is reconfigurable.
 -  For TTC and NTT networks, if this flag is set to **TBX_FALSE**, the congestion status received in the transfer controlled message is stored and the timer **Tc** is started. On expiry of timer **Tc** the congestion status is cleared. For all other networks, the congestion status received in the transfer controlled message is not stored if this flag is set to **TBX_FALSE**. *fSupportRouteSetCongestionTest* should be **TBX_TRUE** for ANSI and all other country variants where this is not an option.
- The number of linkset parameter (*un32NbLinkset*), specifies the number of linksets required in the route. This field is reconfigurable.
- The linkset handle array parameter (*ahLinkset []*), specifies the list handle of linksets to access this route. This field is reconfigurable.
 -  For a self-route configuration (when the *Direction* parameter is set to `TB640_SS7_MTP3_ROUTE_DIRECTION_UP`), we only need one linkset in such a route. It has no real meaning in the SS7 networks but the Trillium SS7 stack needs to get configuration parameter *LinkType* from Userpart to validate the *SlsRange*.
- The linkset priority array parameter (*aLinksetPriority []*), specifies the priority of the linksets for these route. This field is reconfigurable. The highest priority is **0**. There must be at least one linkset with priority **0**. If there are multiple linksets with different priorities for a node, then their priorities must be contiguous. For example, there can be no existing linksets with priority **0**, **1**, and **3** for a node (priority **2** missing).

The **response** part of the message `TB640_MSG_ID_SS7_MTP3_OP_ROUTE_ALLOC` contains the field:

```

...
TB640_SS7_MTP3_ROUTE_HANDLE          hMtp3Route;      /* The handle of the instance of
...                                     the MTP3 route. */

```

5.2.2 Compatibility

5.2.2.1 Variants

See section 4.2.2.1 Variants.

5.2.2.2 Timers

MTP3 uses the standard timers defined in the specifications. As presented in the table below, standard timers map to the timers defined in various control blocks. The following notations are used in the table:

- Not defined: The timer is not provided in the specifications.
- Provided: The timer is defined in the specifications and provided.
- Not provided: The timer is defined in the specifications but not provided.
- N/A: The description does not apply to this cell of the table.


 The timers for CHINA are identical to ITU 88 timers. The timers for TTC/NTT are identical to ITU92 timers except for timers **T5**, **T7**, **T11**, **T12**, **T13**, **T14**, **T17**, **T18**, **T19**, **T20**, **T21**, **T22**, **T23**, which are not applicable for TTC/NTT. TTC/NTT defined timer **Tc** is same as timer **Tc** defined in the route configuration.

Table 28 - MTP3 timers

Timer Number / Our Timer	ITU 88	ITU 92 to ITU 96	ANSI 92 to ANSI 96	xyzAlloc message
T1 / T1	Provided	Provided	Provided	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC
T2 / T2	Provided	Provided	Provided	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC
T3 / T3	Provided	Provided	Provided	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC
T4 / T4	Provided	Provided	Provided	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC
T5 / T5	Provided	Provided	Provided	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC
T6 / T6	Provided	Provided	Provided	TB640_MSG_ID_SS7_MTP3_OP_ROUTE_ALLOC
T7 / T7	Provided	Provided	Provided	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC
T8 / T8	Provided	Provided	Provided	TB640_MSG_ID_SS7_MTP3_OP_ROUTE_ALLOC
T9 /	Not defined	Not defined	Not defined	N/A
T10 / T10	Provided	Provided	Provided	TB640_MSG_ID_SS7_MTP3_OP_ROUTE_ALLOC
T11 / T11	Provided	Provided	Provided	TB640_MSG_ID_SS7_MTP3_OP_ROUTE_ALLOC
T12 / T12	Provided	Provided	Provided	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC
T13 / T13	Provided	Provided	Provided	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC
T14 / T14	Provided	Provided	Provided	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC
T15 / T15	Provided	Provided	Provided	TB640_MSG_ID_SS7_MTP3_OP_ALLOC
T16 / T16	Provided	Provided	Provided	TB640_MSG_ID_SS7_MTP3_OP_ALLOC
T17 / T17	Provided	Provided	Provided	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC
T18 / T18	Provided	Provided	Provided	TB640_MSG_ID_SS7_MTP3_OP_ALLOC
T18 / T18	N/A	N/A	Provided ANSI 96	TB640_MSG_ID_SS7_MTP3_OP_ROUTE_ALLOC
T19 / T19	Provided	N/A	N/A	TB640_MSG_ID_SS7_MTP3_OP_ALLOC
T19 / T19	N/A	Provided	N/A	TB640_MSG_ID_SS7_MTP3_OP_ROUTE_ALLOC
T19 / LinkReferralCraftTimer	N/A	N/A	Provided	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC
T20 / T20	Not provided	Not provided	N/A	Timer T20 is used to limit the time MTP3 uses to send TFPs and PAUSEs at end of restart. Trillium implementation does not limit this time and starts traffic only when all status has been sent.
T20 / T22	Provided	N/A	N/A	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC

T21 / T21	Provided	N/A	N/A	TB640_MSG_ID_SS7_MTP3_OP_ALLOC Timer T21 of general configuration and route configuration. Both timers have the same value as T21 in the specifications.
T21 / T21	N/A	Provided	N/A	TB640_MSG_ID_SS7_MTP3_OP_ROUTE_ALLOC
T21 / T23	N/A	N/A	Provided	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC
T22 / T22	Provided	Provided	N/A	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC
T22 / T18	N/A	N/A	Provided	TB640_MSG_ID_SS7_MTP3_OP_ALLOC
T23 / T23	Provided	Provided	N/A	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC
T23 / T19	N/A	N/A	Provided	TB640_MSG_ID_SS7_MTP3_OP_ALLOC
T24 /	Not provided	Not provided	N/A	In Trillium's implementation, Timer T24 is used to limit the time MTP3 uses to send TFPs and PAUSEs at the end of restart. It is not possible to control the time that MTP3 sends status information because Trillium's code is monolithic and non-reentrant. The restart ends only after all status requests are sent.
T24 /	N/A	N/A	Not provided	In Trillium's implementation, Timer T24 is used to limit the time MTP3 uses to send TFPs and PAUSEs at the end of restart. It is not possible to control the time that MTP3 sends status information because Trillium's code is monolithic and non-reentrant. The restart ends only after all status requests are sent.
T25 / T25	Not defined	Not defined	Provided	TB640_MSG_ID_SS7_MTP3_OP_ROUTE_ALLOC
T26 / T26	Not defined	Not defined	Provided	TB640_MSG_ID_SS7_MTP3_OP_ALLOC
T27	Not defined	Not defined	Not provided	Trillium's implementation of MTP3 does not have any minimum duration of availability during restart. However, Trillium's implementation does not initiate restart during short term isolation, for example, when MTP3 is unable to uninhibit a link when all other links fail.
T28 / T21	Not defined	Not defined	Provided	TB640_MSG_ID_SS7_MTP3_OP_ROUTE_ALLOC
T29 / T19	Not defined	Not defined	Provided	TB640_MSG_ID_SS7_MTP3_OP_ROUTE_ALLOC
T30 /	Not defined	Not defined	Not provided	In Trillium's implementation, Timer T30 is used to limit the time MTP3 uses to send TFPs and PAUSEs at the end of restart. It is not possible to control the time that MTP3 sends status information because Trillium's code is monolithic and non-reentrant. The restart ends only after all status requests are sent.
T31 / FalseLinkCongDetectTimer (same as T35)	Not defined	Not defined	Provided ANSI 96	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC
T32 /	Not defined	Not defined	Not provided	Trillium MTP3 implements Link Oscillation Filtering procedure B which does not use T32
T33 / ProbationLinkOscillationTimer (same as T36)	Not defined	Not defined	Provided ANSI 96	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC

T34 / SuspensionLinkOscillationTimer (same as T37)	Not defined	Not defined	Provided ANSI 96	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC
SLT T1 / SlTTimer (same as T32)	Provided Q.707	Provided Q.707	Provided T1.111.7	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC
SLT T2 / PeriodicSigLinkTstTimer (same as T34)	Provided Q.707	Provided Q.707	Provided T1.111.7	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC
Internal / BsnRequestedTimer (same as T31)	N/A	N/A	N/A	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC
Internal / ConnectingTimer (same as T33)	N/A	N/A	N/A	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC
Internal / FlowControlRequestTimer	N/A	N/A	N/A	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC
Internal / BindConfirmWaitTimer	N/A	N/A	N/A	TB640_MSG_ID_SS7_MTP3_OP_LINK_ALLOC

5.3 MTP3 Alarms

5.3.1 Link Alarms

The TB640_MSG_ID_SS7_MTP3_NOTIF_LINK_ALARM (event) notification message is received by the host application when a MTP3 link is reporting an error.

Structure contains the **event** notification link alarm for a MTP3 link:

```
typedef struct _TB640_EVT_SS7_MTP3_NOTIF_LINK_ALARM
{
    TBX_MSG_HEADER                Header;
    TBX_UINT32                    un32MsgVersion;
    TB640_SS7_MTP3_LINK_HANDLE    hMtp3Link;
    TB640_SS7_MTP3_LINK_ALARM     Alarm;
    TB640_SS7_MTP3_LINK_ALARM_CAUSE Cause;
} TB640_EVT_SS7_MTP3_NOTIF_LINK_ALARM, *PTB640_EVT_SS7_MTP3_NOTIF_LINK_ALARM;
```

General explanation of the field of event notification link alarm:

- The link handle field (*hMtp3Link*), specifies the handle of the MTP3 link instance.
- The alarm field (*Alarm*), indicates an alarm that can be reported at the link level.
Possible values:

Table 29 - MTP3 link alarms

Link Alarm	Description
TB640_SS7_MTP3_LINK_ALARM_UP	Link activated or restored.
TB640_SS7_MTP3_LINK_ALARM_DOWN	Link deactivated or failed.
TB640_SS7_MTP3_LINK_ALARM_MTP2_NOT_AVAILABLE	Mtp2 link has been removed.
TB640_SS7_MTP3_LINK_ALARM_INHIBITED_LOCALLY	Link inhibited locally (management initiated).
TB640_SS7_MTP3_LINK_ALARM_INHIBITED_REMOTELY	Link inhibited remotely (remote node initiated action).

TB640_SS7_MTP3_LINK_ALARM_UNINHIBITED_LOCALLY	Link uninhibited locally (Manager ctrl request).
TB640_SS7_MTP3_LINK_ALARM_UNINHIBITED_REMOTELY	Link uninhibited by remote end.
TB640_SS7_MTP3_LINK_ALARM_INHIBITED_DENIED	Link inhibit request from management denied.
TB640_SS7_MTP3_LINK_ALARM_UNINHIBITED_DENIED	Link uninhibit request from management denied.
TB640_SS7_MTP3_LINK_ALARM_BLOCKED_LOCALLY	Link locally blocked (Manager ctrl request).
TB640_SS7_MTP3_LINK_ALARM_BLOCKED_REMOTELY	Link remotely blocked (unknown cause).
TB640_SS7_MTP3_LINK_ALARM_UNBLOCKED_LOCALLY	Link unblocked (Manager ctrl request).
TB640_SS7_MTP3_LINK_ALARM_UNBLOCKED_REMOTELY	Link unblocked by remote end (unknown cause).
TB640_SS7_MTP3_LINK_ALARM_CONGESTED	Local link congested.
TB640_SS7_MTP3_LINK_ALARM_CONGESTION_STOPPED	Local link congestion abated.
TB640_SS7_MTP3_LINK_ALARM_EMERGENCY	Link is in emergency state.
TB640_SS7_MTP3_LINK_ALARM_EMERGENCY_STOPPED	Link is no more in emergency state.
TB640_SS7_MTP3_LINK_ALARM_INVALID_SLC_REMOTE_END	A signaling link test message (SLT/SLA) received with an invalid SLC.
TB640_SS7_MTP3_LINK_ALARM_INVALID_OPC_REMOTE_END	A signaling link test message (SLT/SLA) received with an invalid OPC.
TB640_SS7_MTP3_LINK_ALARM_SDT_DATA_DROP	Data dropped, The cause will also be set (TB640_SS7_MTP3_LINK_ALARM_CAUSE).

- The cause field (*Cause*), specifies the alarm cause according to alarm type. Possible values:

Table 30 – MTP3 Link alarm cause values

Link Alarm Cause	Description
TB640_SS7_MTP3_LINK_ALARM_CAUSE_DECODE_ERR	Decode error.
TB640_SS7_MTP3_LINK_ALARM_CAUSE_INV_MSG_LENGTH	Invalid message length.
TB640_SS7_MTP3_LINK_ALARM_CAUSE_LNK_DEACT	Link is inactive.
TB640_SS7_MTP3_LINK_ALARM_CAUSE_T17_EXPIRED	Timer T17 expired.
TB640_SS7_MTP3_LINK_ALARM_CAUSE_INV_OPC	Invalid OPC.
TB640_SS7_MTP3_LINK_ALARM_CAUSE_RMT_NEG_ACK	Remote end negatively acknowledged.
TB640_SS7_MTP3_LINK_ALARM_CAUSE_UNKNOWN	Unknown cause
TB640_SS7_MTP3_LINK_ALARM_CAUSE_INV_TST_PTRN	Invalid test pattern in SRT/SRA.
TB640_SS7_MTP3_LINK_ALARM_CAUSE_UNINH_IN_PROG	Uninhibition in progress.
TB640_SS7_MTP3_LINK_ALARM_CAUSE_NO_RMT_ACK	No acknowledgment received from remote end.
TB640_SS7_MTP3_LINK_ALARM_CAUSE_PATH_UNAVAIL	Route to the point code not available.
TB640_SS7_MTP3_LINK_ALARM_CAUSE_RSTR_FAILED	Link restoration failed.
TB640_SS7_MTP3_LINK_ALARM_CAUSE_T32_EXPIRED	Timer T32 (SLT Timer) expired.
TB640_SS7_MTP3_LINK_ALARM_CAUSE_RMT_INIT	Remote node initiated action.
TB640_SS7_MTP3_LINK_ALARM_CAUSE_MGMT_INIT	Layer manager initiated action.
TB640_SS7_MTP3_LINK_ALARM_CAUSE_LOC_RTE_MGMT_INIT	Route management initiated action.
TB640_SS7_MTP3_LINK_ALARM_CAUSE_SLF_RST	Action due to self-restart.
TB640_SS7_MTP3_LINK_ALARM_CAUSE_ADJ_RST	Action due to adjacent restart.
TB640_SS7_MTP3_LINK_ALARM_CAUSE_NULL_NIFOPC	-
TB640_SS7_MTP3_LINK_ALARM_CAUSE_INVALID_DPC	Invalid DPC.
TB640_SS7_MTP3_LINK_ALARM_CAUSE_INH_IN_PROG	Inhibition in progress.
TB640_SS7_MTP3_LINK_ALARM_CAUSE_MTP2_DISC	MTP2 layer disconnected.
TB640_SS7_MTP3_LINK_ALARM_CAUSE_LNKSET_DEACT	Linkset deactivated.
TB640_SS7_MTP3_LINK_ALARM_CAUSE_DPC_UNAVAIL	DPC unavailable.
TB640_SS7_MTP3_LINK_ALARM_CAUSE_SLAVE_SYNC	Slave synchronization.

5.3.2 Linkset Alarms

The TB640_MSG_ID_SS7_MTP3_NOTIF_LINKSET_ALARM (event) notification message is received by the host application when a MTP3 linkset is reporting an error.

Structure contains the **event** notification linkset alarm for a MTP3 linkset:

```
typedef struct _TB640_EVT_SS7_MTP3_NOTIF_LINKSET_ALARM
{
    TBX_MSG_HEADER                Header;
    TBX_UINT32                    un32MsgVersion;
    TB640_SS7_MTP3_LINKSET_HANDLE hMtp3Linkset;
    TB640_SS7_MTP3_LINKSET_ALARM Alarm;
    TBX_UINT32                    un32SupplementalInfo;
} TB640_EVT_SS7_MTP3_NOTIF_LINKSET_ALARM,
*PTB640_EVT_SS7_MTP3_NOTIF_LINKSET_ALARM;
```

General explanation of the field of event notification linkset alarm:

- The linkset handle field (*hMtp3Linkset*) specifies the handle of the MTP3 linkset instance.
- The alarm field (*Alarm*), indicates an alarm that can be reported at the linkset level.
Possible values:

Table 31 - MTP3 Linkset alarms

Linkset Alarm	Description
TB640_SS7_MTP3_LINKSET_ALARM_LSET_ACTIVE	Linkset active.
TB640_SS7_MTP3_LINKSET_ALARM_LSET_INACTIVE	Linkset inactive.
TB640_SS7_MTP3_LINKSET_ALARM_LSET_NTT_ABNORMAL ⁵	Linkset abnormal.
TB640_SS7_MTP3_LINKSET_ALARM_LSET_NTT_HALF_NORMAL ⁶	Linkset half-normal.
TB640_SS7_MTP3_LINKSET_ALARM_LSET_NTT_NORMAL ⁶	Linkset normal.
TB640_SS7_MTP3_LINKSET_ALARM_LSET_NTT_FULL_NORMAL ⁶	Linkset full-normal.

- The cause field (*un32SupplementalInfo*), specifies the alarm supplemental value according to alarm type.

5.3.3 Route Alarms

The TB640_MSG_ID_SS7_MTP3_NOTIF_ROUTE_ALARM (event) notification message is received by the host application when a MTP3 route is reporting an error.

Structure contains the **event** notification route alarm for a MTP3 route:

```
typedef struct _TB640_EVT_SS7_MTP3_NOTIF_ROUTE_ALARM
{
    TBX_MSG_HEADER                Header;
    TBX_UINT32                    un32MsgVersion;
    TB640_SS7_MTP3_ROUTE_HANDLE  hMtp3Route;
}
```

⁵ Not available yet.

```
TB640_SS7_MTP3_ROUTE_ALARM      Alarm;
TBX_UINT32                       un32SupplementalInfo;

} TB640_EVT_SS7_MTP3_NOTIF_ROUTE_ALARM, *PTB640_EVT_SS7_MTP3_NOTIF_ROUTE_ALARM;
```

General explanation of the field of event notification route alarm:

- The route handle field (*hMtp3Route*) specifies the handle of the MTP3 route instance.
- The alarm field (*Alarm*), indicates an alarm that can be reported at the route level.
Possible values:

Table 32 - MTP3 alarms

Route Alarm	Description
TB640_SS7_MTP3_ROUTE_ALARM_DPC_PAUSE	Reported when a concerned point code becomes inaccessible.
TB640_SS7_MTP3_ROUTE_ALARM_DPC_RESUME	Reported when a concerned point code becomes accessible.
TB640_SS7_MTP3_ROUTE_ALARM_DPC_NETWORK_CONGESTED	Reported when a concerned point code becomes congested at level priority. The congestion can be due to a local link congestion used to transport data towards the destination or due to congestion at intermediate nodes in the network.
TB640_SS7_MTP3_ROUTE_ALARM_DPC_NETWORK_CONGESTION_STOPPED	Reported when a concerned point code is out of congestion.
TB640_SS7_MTP3_ROUTE_ALARM_RESTART_BEGIN	Reported by MTP3 to its users when our point code is restarting.
TB640_SS7_MTP3_ROUTE_ALARM_RESTART_END	Reported by MTP3 to its users when our point code finishes the restarting procedure.
TB640_SS7_MTP3_ROUTE_ALARM_DROP_MESSAGE	Concerned DPC drop message (SP routing).
TB640_SS7_MTP3_ROUTE_ALARM_REMOTE_USER_UNAVAILABLE	Concerned DPC remote user unavailable.
TB640_SS7_MTP3_ROUTE_ALARM_RESTRICTED	Concerned DPC restricted.

- The cause field (*un32SupplementalInfo*) specifies the alarm supplemental value according to alarm type.

5.4 MTP3 States

States information, which can be gathered at any time by MTP3, indicates the current state of the software or Service Access Point (SAP). This information indicates the Quality of Service (QoS) parameters and can be used to assist in SS7 software debugging. The collection of status information does not affect any of the information examined.

5.4.1 Link States Get

The TB640_MSG_ID_SS7_MTP3_STATES_LINK_GET (request/response) message is used to obtain states from a MTP3 link.

The **request** part of the message TB640_MSG_ID_SS7_MTP3_STATES_LINK_GET contains the field:

```
...
TB640_SS7_MTP3_LINK_HANDLE    hMtp3Link;    /* The handle of the MTP3 link instance */
...
```

The **response** part of the message TB640_MSG_ID_SS7_MTP3_STATES_LINK_GET contains the field:

```
...
TB640_SS7_MTP3_LINK_STATES    States;        /* Structure containing different readable
* states of the link */
...
```

Structure contains the link states parameters in the message:

```
typedef struct _TB640_SS7_MTP3_LINK_STATES
{
    TB640_SS7_MTP3_LINK_STATE    LinkState;
    TBX_BOOL                     fLocallyInhibited;
    TBX_BOOL                     fRemotelyInhibited;
    TBX_BOOL                     fLocallyBlocked;
    TBX_BOOL                     fRemotelyBlocked;
    TBX_BOOL                     fCongested;
    TBX_BOOL                     fEmergencyIndication;
    TBX_UINT32                   un32RTxQueueSize;
    TBX_UINT32                   un32TxQueueSize;
    TB640_SS7_MTP3_PRIORITY      CongestionPriority;
} TB640_SS7_MTP3_LINK_STATES, *PTB640_SS7_MTP3_LINK_STATES;
```

General explanation of the link states parameters:

- The link state parameter (*LinkState*) indicates the current state of a MTP3 link. Possible values:

Table 33 - MTP3 link states

Link State	Description
TB640_SS7_MTP3_LINK_STATE_UP	Link in UP state (aligned and ready to carry traffic). A link in UP state can carry traffic provided it is not inhibited or blocked. MTP3 chooses an active, unblocked and uninhibited link to carry traffic based on the link priority and the requirements for load sharing. Refer to the SS7 specifications for details on load sharing algorithms.
TB640_SS7_MTP3_LINK_STATE_DOWN	Link in DOWN state. An active link moves to DOWN state when the alignment is lost or when the layer manager deactivates the link or inhibition or blocking is initiated for this link.
TB640_SS7_MTP3_LINK_MTP2_LINK_NOT_AVAILABLE	MTP3 link is not configured and is waiting for MTP2 link configuration.

- The locally inhibited flag (*fLocallyInhibited*) indicates whether or not the link is locally inhibited. Possible values are **TBX_TRUE** or **TBX_FALSE**.
- The remotely inhibited flag (*fRemotelyInhibited*) indicates whether or not the link is remotely inhibited. Possible values are **TBX_TRUE** or **TBX_FALSE**.
- The locally blocked flag (*fLocallyBlocked*) indicates whether or not the link is locally blocked. Possible values are **TBX_TRUE** or **TBX_FALSE**.
- The remotely blocked flag (*fRemotelyBlocked*) indicates whether or not the link is remotely blocked. Possible values are **TBX_TRUE** or **TBX_FALSE**.
- The congested flag (*fCongested*) indicates whether or not the link is in congestion state. Possible values are **TBX_TRUE** or **TBX_FALSE**.
- The emergency indication flag (*fEmergencyIndication*) indicates we are in emergency condition. Possible values are **TBX_TRUE** or **TBX_FALSE**.
- The retransmit queue size field (*un32RTxQueueSize*) gives the number of messages presents in link retransmit queue. Messages in retransmit queue are present only for a short duration until the buffer retrieval is completed during the changeover procedure.
- The transmit queue size field (*un32TxQueueSize*), gives the number of messages queued in the link transmit queue.
- The congestion priority field (*CongestionPriority*). Allowable values: see **Table 23 - MTP3 link priorities**.

The current congestion status is updated based on the configured thresholds (*un32PrioXMsgQueueLen*) hit in the link transmit queue after queuing is initiated. For international networks and networks without multiple congestion states (where all *un32PrioXMsgQueueLe* are same), the *CongestionPriority* can take only two values: **TB640_SS7_MTP3_PRIORITY_3** (link congested) or **TB640_SS7_MTP3_PRIORITY_0** (no congestion).

5.4.2 Link States Set

The **TB640_MSG_ID_SS7_MTP3_STATES_LINK_SET** (request/response) message is used to request a states change for a MTP3 link.

The **request** part of the message **TB640_MSG_ID_SS7_MTP3_STATES_LINK_SET** contains the field:


```
...
TB640_SS7_MTP3_LINK_HANDLE          hMtp3Link;    /* The handle of the MTP3 link instance */
TB640_SS7_MTP3_LINK_WRITEABLE_STATES States;
...
```

Structure contains the link states parameters in the message:

```
typedef struct _TB640_SS7_MTP3_LINK_WRITEABLE_STATES
{
    TBX_BOOL                fLocallyInhibited;
    TBX_UINT8               aun8Padding0 [4]; /* Align on 64 bits */
} TB640_SS7_MTP3_LINK_WRITEABLE_STATES, *PTB640_SS7_MTP3_LINK_WRITEABLE_STATES;
```

General explanation of the link states parameters:

- The locally inhibited flag (*fLocallyInhibited*) indicates whether or not the link is locally inhibited. Possible values are **TBX_TRUE** or **TBX_FALSE**.


 A link under congestion cannot be inhibited. If the link is the only one available for traffic, it cannot be inhibited.

The **response** part of the message TB640_MSG_ID_SS7_MTP3_STATES_LINK_SET contains the field:

```
...
TBX_RESULT                Result;
...
```

General explanation of the states response parameters:

- Result code of the request operation.

 The result code can be successful and you can received an TB640_SS7_MTP3_LINK_ALARM_INHIBITED_DENIED or TB640_SS7_MTP3_LINK_ALARM_UNINHIBITED_DENIED alarm.

5.4.3 Route Linkset States Get

The TB640_MSG_ID_SS7_MTP3_STATES_ROUTE_LINKSET_GET (request/response) message is used to obtain states from a MTP3 route linkset.

The **request** part of the message TB640_MSG_ID_SS7_MTP3_STATES_ROUTE_LINKSET_GET contains the fields:

```
...
TB640_SS7_MTP3_LINKSET_HANDLE hMtp3Linkset; /* The handle of the MTP3 linkset instance */
TB640_SS7_MTP3_ROUTE_HANDLE   hMtp3Route; /* The handle of the MTP3 route */
...
```

The **response** part of the message TB640_MSG_ID_SS7_MTP3_STATES_ROUTE_LINKSET_GET contains the field:

```
...
TB640_SS7_MTP3_LINKSET_STATES States; /* Structure containing different readable
* states of the linkset
...
```

Structure contains the linkset states parameters in the message:


```
typedef struct _TB640_SS7_MTP3_LINKSET_STATES
{
    TB640_SS7_MTP3_LINKSET_STATUS          LinksetStatus;
    TBX_BOOL                                fTransmitProhibitState;
    TBX_BOOL                                fTransmitRestrictedState;
    TBX_UINT8                               aun8Padding0 [4];
} TB640_SS7_MTP3_LINKSET_STATES, *PTB640_SS7_MTP3_LINKSET_STATES;
```

General explanation of the linkset states parameters:

- The linkset status parameter (*LinksetStatus*) indicates the current status of a MTP3 linkset. Possible values:

Table 34 - MTP3 linkset status

Linkset State	Description
TB640_SS7_MTP3_LINKSET_STATUS_ACTIVE	A link set in active state can carry traffic.
TB640_SS7_MTP3_LINKSET_STATUS_INACTIVE	A link set in inactive state cannot carry traffic.


-  Link capability state for NTT (JAPAN). This field gives the number of active links in the link set. Possible values:

Table 35 - MTP3 linkset status for NTT

Linkset State	Description
TB640_SS7_MTP3_LINKSET_STATUS_ABNORMAL	No links are active.
TB640_SS7_MTP3_LINKSET_STATUS_HALF_NORMAL	Less than half links are active.
TB640_SS7_MTP3_LINKSET_STATUS_NORMAL	More than half links are active.
TB640_SS7_MTP3_LINKSET_STATUS_FULL_NORMAL	All the links are active.

- The transmit prohibit state flag (*fTransmitProhibitState*), is set to TBX_TRUE when a transfer prohibit message (TFP) is received from the network on this linkset for the point code represented by the route of the linkset. This flag is reset when a transfer allowed message (TFA) or a transfer restrict message (TFR) is received from the network. Possible values are TBX_TRUE or TBX_FALSE.
- The transmit restricted state flag (*fTransmitRestrictedState*), is set to TBX_TRUE when a TFR is received from the network on this linkset for the point code represented by the route of the linkset. This flag is reset when a TFA or TFP is received from the network. Possible values are TBX_TRUE or TBX_FALSE.

5.4.4 Route Linkset States Set

The TB640_MSG_ID_SS7_MTP3_STATES_ROUTE_LINKSET_SET (request/response) message is used to request a states change for a MTP3 route linkset.

The **request** part of the message TB640_MSG_ID_SS7_MTP3_STATES_ROUTE_LINKSET_SET contains the fields:

```
...
TB640_SS7_MTP3_LINKSET_HANDLE          hMtp3Linkset; /* The handle of the MTP3 linkset
instance */
TB640_SS7_MTP3_ROUTE_HANDLE           hMtp3Route; /* The handle of the MTP3 route*/
```

```
TB640_SS7_MTP3_LINKSET_WRITEABLE_STATES      States;  
...
```

Structure contains the linkset writeable states parameters in the message:

```
typedef struct _TB640_SS7_MTP3_LINKSET_WRITEABLE_STATES  
{  
    TBX_BOOL          fActive;  
    TBX_UINT8         aun8Padding0 [4];          /* Align on 64 bits */  
} TB640_SS7_MTP3_LINKSET_WRITEABLE_STATES, *PTB640_SS7_MTP3_LINKSET_WRITEABLE_STATES;
```

General explanation of the linkset states parameters:

- The active parameter (*fActive*) indicates whether or not the linkset is active. Possible values are TBX_TRUE or TBX_FALSE.

The **response** part of the message TB640_MSG_ID_SS7_MTP3_STATES_ROUTE_LINKSET_SET contains the field:

```
...  
TBX_RESULT          Result;  
...
```

General explanation of the states response parameters:

- Result code of the request operation.

5.4.5 Route States Get

The TB640_MSG_ID_SS7_MTP3_STATES_ROUTE_GET (request/response) message is used to request a states change for a MTP3 route.

The **request** part of the message TB640_MSG_ID_SS7_MTP3_STATES_ROUTE_GET contains the field:

```
...  
TB640_SS7_MTP3_ROUTE_HANDLE          hMtp3Route;  /* The handle of the MTP3 route*/  
...
```

The **response** part of the message TB640_MSG_ID_SS7_MTP3_STATES_ROUTE_GET contains the fields:

```
...  
TB640_SS7_MTP3_ROUTE_STATES          States;  /* Structure containing different readable  
                                           states of the route */  
...
```

Structure contains the route states parameters in the message:

```
typedef struct _TB640_SS7_MTP3_ROUTE_STATES  
{  
    TBX_UINT8         aun8Padding1 [4];  
    TBX_BOOL          fPaused;  
    TBX_BOOL          fCongested;  
    TBX_BOOL          fDpcRestart;  
  
    /* States non related to alarms */  
}
```

```

    TB640_SS7_MTP3_PRIORITY    CongestionPriority;

    TBX_UINT8                  aun8Padding0 [4];
} TB640_SS7_MTP3_ROUTE_STATES, *PTB640_SS7_MTP3_ROUTE_STATES;

```

General explanation of the route states parameters:

- The paused parameter (*fPaused*) indicates whether or not the route (destination DPC) is reachable. Possible values are TBX_TRUE or TBX_FALSE.
- The congested parameter (*fCongested*) indicates whether or not the route (destination DPC) is congested. Possible values are TBX_TRUE or TBX_FALSE.
- The destination point code restart parameter (*fDpcRestart*) indicates whether or not the route (destination DPC) is doing restart procedures. Possible values are TBX_TRUE or TBX_FALSE.
- The congestion priority parameter (*CongestionPriority*) indicates the current congestion priority. Allowable values: see **Table 23 - MTP3 link priorities**.

5.5 MTP3 Statistics

The MTP3 can gather statistics information at any time to measure the performance of the MTP3 software. This information can be used to determine the distribution of traffic loads and Quality of Service (QoS) parameters, and to assist in MTP3 software debugging. The collection of statistics information may or may not result in the counters being reset.

5.5.1 General Statistics

The TB640_MSG_ID_SS7_MTP3_STATS_GET (request/response) message is used to obtain statistics from a MTP3 layer.

The **request** part of the message TB640_MSG_ID_SS7_MTP3_STATS_GET contains the field:

```

...
TB640_SS7_MTP3_HANDLE          hMtp3;          /* Handle of the MTP3 layer */
...

```

The **response** part of the message TB640_MSG_ID_SS7_MTP3_STATS_GET contains the field:

```

...
TB640_SS7_MTP3_STATISTICS      Statistics;
...

```

Structure contains the statistics for general layer MTP3:

```

typedef struct _TB640_SS7_MTP3_STATISTICS
{
    TBX_UINT32                  un32UnavailableRxMsg;
    TBX_UINT32                  un32UnavailableTxMsg;
    TBX_UINT32                  un32TrafficRestartTxMsg;
    TBX_UINT32                  un32TrafficRestartRxMsg;
    TBX_UINT32                  un32TrafficRestartWaitingTxMsg;
    TBX_UINT32                  un32TrafficRestartWaitingRxMsg;
}

```

```
TBX_UINT32      un32MsuDroppedDueToRoutingError;  
TBX_UINT8      aun8Padding0 [4];  
  
} TB640_SS7_MTP3_STATISTICS, *PTB640_SS7_MTP3_STATISTICS;
```

General explanation of the general layer MTP3 statistics parameters:

- The unavailable received message counter (*un32UnavailableRxMsg*) indicates the number of userpart unavailable message received for a user of this layer.
- The unavailable transmit message counter (*un32UnavailableTxMsg*), indicates the number of userpart unavailable transmitted because of an unavailable user at this layer.
- The traffic restart transmit message counter (*un32TrafficRestartTxMsg*), indicates the number of traffic restart allowed transmitted.
- The traffic restart received message counter (*un32TrafficRestartRxMsg*) indicates the number of traffic restart allowed received.
- The traffic restart waiting transmit message counter (*un32TrafficRestartWaitingTxMsg*), indicates the number of traffic restart waiting transmitted for **ANSI version**.
- The traffic restart waiting received message counter (*un32TrafficRestartWaitingRxMsg*), indicates the number of traffic restart waiting received for **ANSI version**.
- The MSU dropped due to routing error counter (*un32MsuDroppedDueToRoutingError*), indicates the number of drop may occur for the following reasons:
 - Self or adjacent restart is in progress
 - SLS value is wrong
 - Circular routing
 - MSU is received at an SP for some other node (SP cannot route)
 - The destination is unreachable or unavailable
 - The layer encounters an illegal routing configuration or a bad packet.

5.5.2 Link Statistics

The TB640_MSG_ID_SS7_MTP3_STATS_LINK_GET (request/response) message is used to obtain statistics from a MTP3 link.

The **request** part of the message TB640_MSG_ID_SS7_MTP3_STATS_LINK_GET contains the field:

```
...  
TB640_SS7_MTP3_LINK_HANDLE  hMtp3Link;      /* Handle of the MTP3 link instance */  
...
```

The **response** part of the message TB640_MSG_ID_SS7_MTP3_STATS_LINK_GET contains the field:

```
...
```

```
TB640_SS7_MTP3_LINK_STATISTICS    Statistics;
...
```

Structure contains the statistics for a MTP3 link:

```
typedef struct _TB640_SS7_MTP3_LINK_STATISTICS
{
    TBX_UINT32    un32NbChangeoverTx;
    TBX_UINT32    un32NbChangeoverRx;
    TBX_UINT32    un32NbChangeoverAckTx;
    TBX_UINT32    un32NbChangeoverAckRx;
    TBX_UINT32    un32NbChangebackTx;
    TBX_UINT32    un32NbChangebackRx;
    TBX_UINT32    un32NbChangebackAckTx;
    TBX_UINT32    un32NbChangebackAckRx;
    TBX_UINT32    un32NbEmergencyChangeoverTx;
    TBX_UINT32    un32NbEmergencyChangeoverRx;
    TBX_UINT32    un32NbEmergencyChangeoverAckTx;
    TBX_UINT32    un32NbEmergencyChangeoverAckRx;
    TBX_UINT32    un32NbLinkInhibitTx;
    TBX_UINT32    un32NbLinkInhibitRx;
    TBX_UINT32    un32NbLinkInhibitAckTx;
    TBX_UINT32    un32NbLinkInhibitAckRx;
    TBX_UINT32    un32NbLinkInhibitDeniedTx;
    TBX_UINT32    un32NbLinkInhibitDeniedRx;
    TBX_UINT32    un32NbLinkUninhibitTx;
    TBX_UINT32    un32NbLinkUninhibitRx;
    TBX_UINT32    un32NbLinkUninhibitAckTx;
    TBX_UINT32    un32NbLinkUninhibitAckRx;
    TBX_UINT32    un32NbLinkForceUninhibitTx;
    TBX_UINT32    un32NbLinkForceUninhibitRx;
    TBX_UINT32    un32NbLocalLinkInhibitTestTx;
    TBX_UINT32    un32NbLocalLinkInhibitTestRx;
    TBX_UINT32    un32NbRemoteLinkInhibitTestTx;
    TBX_UINT32    un32NbRemoteLinkInhibitTestRx;
    TBX_UINT32    un32NBLinkConnectionOrderTx;
    TBX_UINT32    un32NBLinkConnectionOrderRx;
    TBX_UINT32    un32NBLinkConnectionAckTx;
    TBX_UINT32    un32NBLinkConnectionAckRx;
    TBX_UINT32    un32NBLinkTestTx;
    TBX_UINT32    un32NBLinkTestRx;
    TBX_UINT32    un32NBLinkTestAckTx;
    TBX_UINT32    un32NBLinkTestAckRx;
    TBX_UINT32    un32TxMsdropped;
    TBX_UINT32    un32TxMsdroppedDueToCongestion;
    TBX_UINT32    un32NbSifByteTx;
    TBX_UINT32    un32NbSifByteRx;
    TBX_UINT32    un32NbSioByteTx;
    TBX_UINT32    un32NbSioByteRx;
    TBX_UINT32    un32NbMsdTx;
    TBX_UINT32    un32NbMsdRx;
    TBX_UINT32    un32NbLinkCongestionSetLevel1;
    TBX_UINT32    un32NbLinkCongestionSetLevel2;
    TBX_UINT32    un32NbLinkCongestionSetLevel3;
    TBX_UINT32    un32LinkUnavailabilityMsec;
    TBX_UINT32    un32LinkCongestedMsec;
    TBX_UINT32    un32NbInvalidPduRx;
} TB640_SS7_MTP3_LINK_STATISTICS, *PTB640_SS7_MTP3_LINK_STATISTICS;
```

General explanation of the MTP3 link statistics parameters:

- The number of changeover transmit counter (*un32NbChangeoverTx*), indicates the number of changeover order transmitted for this link.
- The number of changeover received counter (*un32NbChangeoverRx*), indicates the number of changeover order received for this link.
- The number of changeover acknowledgement transmit counter (*un32NbChangeoverAckTx*), indicates the number of changeover acknowledgment transmitted for this link.
- The number of changeover acknowledgement received counter (*un32NbChangeoverAckRx*), indicates the number of changeover acknowledgment received for this link.
- The number of changeback transmit counter (*un32NbChangebackTx*), indicates the number of changeback declaration transmitted on this link.
- The number of changeback received counter (*un32NbChangebackRx*), indicates the number of changeback declaration received on this link.
- The number of changeback acknowledgement transmit counter (*un32NbChangebackAckTx*), indicates the number of changeback acknowledgment transmitted for this link.
- The number of changeback acknowledgement received counter (*un32NbChangebackAckRx*), indicates the number of changeback acknowledgment received for this link.
- The number of emergency changeover transmit counter (*un32NbEmergencyChangeoverTx*), indicates the number of emergency changeover transmitted for this link.
- The number of emergency changeover received counter (*un32NbEmergencyChangeoverRx*), indicates the number of emergency changeover received for this link.
- The number of emergency changeover acknowledgement transmit counter (*un32NbEmergencyChangeoverAckTx*), indicates the number of emergency changeover acknowledgment transmitted for this link.
- The number of emergency changeover acknowledgement received counter (*un32NbEmergencyChangeoverAckRx*), indicates the number of emergency changeover acknowledgment received for this link.

- The number of link inhibit transmit counter (*un32NbLinkInhibitTx*), indicates the number of link inhibit transmitted for this link.
- The number of link inhibit received counter (*un32NbLinkInhibitRx*), indicates the number of link inhibit received for this link.
- The number of link inhibit acknowledgement transmit counter (*un32NbLinkInhibitAckTx*), indicates the number of link inhibit acknowledgement transmitted for this link.
- The number of link inhibit acknowledgement received counter (*un32NbLinkInhibitAckRx*), indicates the number of link inhibit acknowledgement received for this link.
- The number of link inhibit denied transmit counter (*un32NbLinkInhibitDeniedTx*), indicates the number of link inhibit denied transmitted for this link.
- The number of link inhibit denied received counter (*un32NbLinkInhibitDeniedRx*), indicates the number of link inhibit denied received for this link.
- The number of link uninhibit transmit counter (*un32NbLinkUninhibitTx*), indicates the number of link uninhibit transmitted for this link.
- The number of link uninhibit received counter (*un32NbLinkUninhibitRx*), indicates the number of link uninhibit received for this link.
- The number of link uninhibit acknowledgement transmit counter (*un32NbLinkUninhibitAckTx*), indicates the number of link uninhibit acknowledgement transmitted for this link.
- The number of link uninhibit acknowledgement received counter (*un32NbLinkUninhibitAckRx*), indicates the number of link uninhibit acknowledgement received for this link.
- The number of link force uninhibit transmit counter (*un32NbLinkForceUninhibitTx*), indicates the number of link force uninhibit transmitted for this link.
- The number of link force uninhibit received counter (*un32NbLinkForceUninhibitRx*), indicates the number of link force uninhibit received for this link.
- The number of local link inhibit test transmit counter (*un32NbLocalLinkInhibitTestTx*), indicates the number of local link inhibit test transmitted for this link.
- The number of local link inhibit test received counter (*un32NbLocalLinkInhibitTestRx*), indicates the number of local link inhibit test received for this link.

- The number of remote link inhibit test transmit counter (*un32NbRemoteLinkInhibitTestTx*), indicates the number of remote link inhibit test transmitted for this link.
- The number of remote link inhibit test received counter (*un32NbRemoteLinkInhibitTestRx*), indicates the number of remote link inhibit test received for this link.
- The number of link connection order transmit counter (*un32NBLinkConnectionOrderTx*), indicates the number of link connection order transmitted for this link.
- The number of link connection order received counter (*un32NBLinkConnectionOrderRx*), indicates the number of link connection order received for this link.
- The number of link connection acknowledgement transmit counter (*un32NBLinkConnectionAckTx*), indicates the number of link connection acknowledgement transmitted for this link.
- The number of link connection acknowledgement received counter (*un32NBLinkConnectionAckRx*), indicates the number of link connection acknowledgement received for this link.
- The number of link test transmit counter (*un32NBLinkTestTx*), indicates the number of signaling link test message transmitted on this link.
- The number of link test received counter (*un32NBLinkTestRx*), indicates the number of signaling link test message received on this link.
- The number of link test acknowledgement transmit counter (*un32NBLinkTestAckTx*), indicates the number of signaling link test acknowledgement message transmitted on this link.
- The number of link test acknowledgement received counter (*un32NBLinkTestAckRx*), indicates the number of signaling link test acknowledgement message received on this link.
- The number of transmit MSU dropped counter (*un32TxMsuDropped*). A message to be transmitted cannot be sent and consequently is dropped. This may occur for the following reasons:
 - There is an error in encoding the message.
 - The size of the message is illegal.
 - The link is congested and a message is a COO or ECO. Changeover

messages are not queued, as congestion abatement at a later time may bring down the link for which the changeover was previously sent.

- A message retrieved from layer2 during changeover cannot be decoded or contains an illegal field.
- The number of transmit MSU dropped due to congestion counter (*un32TxMsuDroppedDueToCongestion*), indicates the number of MSUs dropped because of congestion on this link.
- The number of SIF byte transmit counter (*un32NbSifByteTx*), indicates the number of SIF octets transmitted on this link.
- The number of SIF byte received counter (*un32NbSifByteRx*), indicates the number of SIF octets received on this link.
- The number of SIO byte transmit counter (*un32NbSioByteTx*), indicates the number of SIO octets transmitted on this link.
- The number of SIO byte received counter (*un32NbSioByteRx*), indicates the number of SIO octets received on this link.
- The number of MSU transmit counter (*un32NbMsuTx*), indicates the number of MSUs transmitted on this link.
- The number of MSU received counter (*un32NbMsuRx*), indicates the number of MSUs received on this link.
- The number of link congestion set level 1 counter (*un32NbLinkCongestionSetLevel1*), indicates the number of link congestion set at threshold 1.
- The number of link congestion set level 2 counter (*un32NbLinkCongestionSetLevel2*), indicates the number of link congestion set at threshold 2.
- The number of link congestion set level 3 counter (*un32NbLinkCongestionSetLevel3*), indicates the number of link congestion set at threshold 3.
- The number of link unavailability counter (*un32LinkUnavailabilityMsec*), indicates the duration of link unavailability in Msec.
- The number of link congested counter (*un32LinkCongestedMsec*), indicates the duration of link congestion in Msec.
- The number of invalid PDU received counter (*un32NbInvalidPduRx*), indicates the number of invalid PDU received on this link. This may reflect one of the following conditions:

- There is an error in encoding the message.
- PDU has a wrong size. That is, less than the minimum allowed or more than the maximum allowed.
- Length of PDU cannot be found.
- PDU cannot be decoded.
- PDU contains illegal SLS value.
- PDU is received in an invalid link state.

5.5.3 Route Linkset Statistics

The TB640_MSG_ID_SS7_MTP3_STATS_ROUTE_LINKSET_GET (request/response) message is used to obtain statistics from a MTP3 route linkset.

The **request** part of the message TB640_MSG_ID_SS7_MTP3_STATS_ROUTE_LINKSET_GET contains the fields:

```
...
TB640_SS7_MTP3_LINKSET_HANDLE      hMtp3Linkset;      /* Handle of the MTP3
                               linkset instance */
TB640_SS7_MTP3_ROUTE_HANDLE        hMtp3Route;        /* Handle of the MTP3
                               route */
...
```

The **response** part of the message TB640_MSG_ID_SS7_MTP3_STATS_ROUTE_LINKSET_GET contains the field:

```
...
TB640_SS7_MTP3_LINKSET_STATISTICS  Statistics;
...
```

Structure contains the statistics for a MTP3 route linkset:

```
typedef struct _TB640_SS7_MTP3_LINKSET_STATISTICS
{
    TBX_UINT32      un32NbStartOfLinksetFailure;
    TBX_UINT32      un32NbStopOfLinksetFailure;
    TBX_UINT32      un32LinksetUnavailabilityMsec;
    TBX_UINT8       aun8Padding0 [4];
} TB640_SS7_MTP3_LINKSET_STATISTICS, *PTB640_SS7_MTP3_LINKSET_STATISTICS;
```

General explanation of the MTP3 route linkset statistics parameters:

- The number of start of linkset failure counter (*un32NbStartOfLinksetFailure*) indicates the number of start of linkset failure.
- The number of stop of linkset failure counter (*un32NbStopOfLinksetFailure*), indicates the number of stop of linkset failure.
- The number of linkset unavailability counter (*un32LinksetUnavailabilityMsec*), indicates the duration of link set unavailability in Msec.

5.5.4 Route Statistics

The TB640_MSG_ID_SS7_MTP3_STATS_ROUTE_GET (request/response) message is used to obtain statistics from a MTP3 route.

The **request** part of the message TB640_MSG_ID_SS7_MTP3_STATS_ROUTE_GET contains the field:

```
...
TB640_SS7_MTP3_ROUTE_HANDLE    hMtp3Route;    /* Handle of the MTP3 route */
...
```

The **response** part of the message TB640_MSG_ID_SS7_MTP3_STATS_ROUTE_GET contains the field:

```
...
TB640_SS7_MTP3_ROUTE_STATISTICS    Statistics;
...
```

Structure contains the statistics for a MTP3 route:

```
typedef struct _TB640_SS7_MTP3_ROUTE_STATISTICS
{
    TBX_UINT32    un32NbRouteTestMsgTx;
    TBX_UINT32    un32NbRouteTestMsgRx;
    TBX_UINT32    un32NbCongestionTestMsgTx;
    TBX_UINT32    un32NbCongestionTestMsgRx;
    TBX_UINT32    un32NbTransferProhibitedMsgTx;
    TBX_UINT32    un32NbTransferProhibitedMsgRx;
    TBX_UINT32    un32NbTransferRestrictMsgTx;
    TBX_UINT32    un32NbTransferRestrictMsgRx;
    TBX_UINT32    un32NbTransferControlledMsgTx;
    TBX_UINT32    un32NbTransferControlledMsgRx;
    TBX_UINT32    un32NbRouteUnavailable;
    TBX_UINT32    un32NbSifByteTx;
    TBX_UINT32    un32NbSioByteTx;
    TBX_UINT32    un32RouteUnavailabilityMsec;
    TBX_UINT32    un32NbUsnRx;
    TBX_UINT8     aun8Padding0 [4];
} TB640_SS7_MTP3_ROUTE_STATISTICS, *PTB640_SS7_MTP3_ROUTE_STATISTICS;
```

General explanation of the MTP3 route statistics parameters:

- The number of route test message transmit counter (*un32NbRouteTestMsgTx*), indicates the number of route set test message transmitted to this route.
- The number of route test message received counter (*un32NbRouteTestMsgRx*), indicates the number of route set test message received to this route.
- The number of congestion test message transmit counter (*un32NbCongestionTestMsgTx*), indicates the number of route set congestion test message transmitted to this route.

- The number of congestion test message received counter (*un32NbCongestionTestMsgRx*), indicates the number of route set congestion test message received to this route.
- The number of transfer prohibited message transmit counter (*un32NbTransferProhibitedMsgTx*), indicates the number of transfer prohibited transmitted to this route.
- The number of transfer prohibited message received counter (*un32NbTransferProhibitedMsgRx*), indicates the number of transfer prohibited received to this route.
- The number of transfer restrict message transmit counter (*un32NbTransferRestrictMsgTx*), indicates the number of transfer restricted transmitted to this route.
- The number of transfer restrict message received counter (*un32NbTransferRestrictMsgRx*), indicates the number of transfer restricted received to this route.
- The number of transfer controlled message transmit counter (*un32NbTransferControlledMsgTx*), indicates the number of transfer controlled transmitted to this route.
- The number of transfer controlled message received counter (*un32NbTransferControlledMsgRx*), indicates the number of transfer controlled received to this route.
- The number of route unavailable counter (*un32NbRouteUnavailable*), indicates the number of route became unavailable.
- The number of SIF byte transmit counter (*un32NbSifByteTx*), indicates the number of SIF octets transmitted to this route.
- The number of SIO byte transmit counter (*un32NbSioByteTx*), indicates the number of SIO octets transmitted to this route.
- The number of route unavailability counter (*un32RouteUnavailabilityMsec*), indicates the duration of route unavailability in Msec.
- The number of USN received counter (*un32NbUsnRx*), indicates the number of USN (Unallocated Signaling Number) messages received. **Specific to NTT and TTC networks.**

5.6 MTP3 Standalone mode

The MTP3 standalone mode is only active when the MTP3 Userpart connection mode is set to TB640_SS7_MTP3_CONNECTION_MODE_HOST_ISUP (or _TUP or _SCCP).

5.6.1 Send Data

The TB640_MSG_ID_SS7_MTP3_OP_SEND_DATA (request/response) message is used to transmit a buffer (or set of buffers) from the host application to a MTP3 userpart. The message can contain 16 different frames. Each frames of the array will be sent with the same buffer order. If one frame generates an error while being enqueued, all subsequent frames in the same message will NOT be sent to prevent frames to be sent in the wrong order.

The **request** part of the message TB640_MSG_ID_SS7_MTP3_OP_SEND_DATA contains the field:

```

...
TB640_SS7_MTP3_USERPART_HANDLE      hMtp3Userpart;
TBX_UINT32                          un32NbFrameInMsg;
TB640_SS7_MTP3_SEND_DATA_CFG        aCfg [TB640_SS7_MTP3_MAX_FRAME_IN_MESSAGE];
TBX_UINT8                            aun8Payload [1];
...

```

Structure contains the send data configuration parameters for one frame in the message:

```

typedef struct _TB640_SS7_MTP3_SEND_DATA_CFG
{
    TBX_UINT32          un32StructVersion;
    TBX_UINT8           aun8Padding0 [4];
    TB640_SS7_POINT_CODE Opc;
    TB640_SS7_POINT_CODE Dpc;
    TB640_SS7_MTP3_MESSAGE_PRIORITY MessagePriority;
    TBX_UINT32          un32Sls;
    TBX_UINT64          un64UserContext;
    TBX_UINT32          un32FrameOffset;
    TBX_UINT32          un32PayloadSize;
} TB640_SS7_MTP3_SEND_DATA_CFG, *PTB640_SS7_MTP3_SEND_DATA_CFG;

```

General explanation of the parameters of send data configuration:

- The userpart handle parameter (*hMtp3Userpart*) specifies the handle of the MTP3 userpart instance.
- The number of frame (*un32NbFrameInMsg*) indicates the number of frames included in the *aun8Payload*.
- The originating point code parameter (*Opc*). If *Opc* is NULL, MTP3 uses a configured default point code (see *DefaultOpc1* in MTP3 general configuration) before transferring the message.
- The destination point code parameter (*Dpc*).

- The priority of the message parameter (*MessagePriority*). Allowable values: see **Table 24 - MTP3 message priorities**. The priority value should be set to TB640_SS7_MTP3_MESSAGE_PRIORITY_0 if the message priority is not being used. For international networks, the priority value should be set to TB640_SS7_MTP3_MESSAGE_PRIORITY_0. For national networks which do not support multiple congestion priorities, MTP3 does not use the supplied priority.
- The Signaling link selector field (*un32Sls*) specifies a unique number used by MTP users in a signaling network to associate all messages belonging to a particular connection or transaction. MTP3 transmits all messages, with a specific SLS value, on the same signaling link under normal circumstances. The range of SLS for a given network type is given in the table below:

Table 36 - MTP3 SLS ranges value

Network Type	Range of SLS
ITU	0 to 15 (16 SLSs)
ANSI	0 to 31 or 0 to 255 (based on whether 5 bit SLS or 8 bit SLS is used)
TTC	0 to 15 (16 SLSs)
NTT	0 to 31 (32 SLSs)
CHINA	0 to 15 (16 SLSs)

- The user context parameter (*un64UserContext*), it's an user contexts (user value) that are going to be returned in the response for a frame of the message.
- The frame offset parameter (*un32FrameOffset*), specifies the offset from the start of the *aun8Payload* (from the TB640_REQ_SS7_MTP3_OP_SEND_DATA structure of the TB640_MSG_ID_SS7_MTP3_OP_SEND_DATA message) field to which the frame start.
- The payload size parameter (*un32PayloadSize*) specifies the byte count of the frame in the *aun8Payload*.
- The payload parameter (*aun8Payload*) contains all sent frames.

The **response** part of the message TB640_MSG_ID_SS7_MTP3_OP_SEND_DATA contains the fields:

```

...
TBX_UINT32    un32NbFrameInMsg;    /* Number of frames included in the
                                   response */

TBX_UINT64    aun64UserContext [TB640_SS7_MTP3_MAX_FRAME_IN_MESSAGE];
                                   /* Array containing user-values associated
                                   with the buffers */

TBX_RESULT    aResult [TB640_SS7_MTP3_MAX_FRAME_IN_MESSAGE];
                                   /* Array of result code for every frames.
                                   An error value means the frames hasn't
                                   been queued for transmission.*/
...

```

5.6.2 Received Data

The TB640_MSG_ID_SS7_MTP3_NOTIF_RECEIVED_DATA (event) notification message is received by the host application when a buffer is received on an MTP3 userpart. The notification message can contain 16 different frames. Each frames of the array will be received with the same MTP3 received order.

Structure contains the **event** notification received data from a MTP3 Userpart:

```
typedef struct _TB640_EVT_SS7_MTP3_NOTIF_RECEIVED_DATA
{
    TBX_MSG_HEADER                Header;
    TBX_UINT32                    un32MsgVersion;
    TB640_SS7_MTP3_USERPART_HANDLE hMtp3Userpart;
    TBX_UINT32                    un32NbFrameInMsg;
    TBX_UINT8                    aun8Padding0 [4];
    TB640_SS7_MTP3_RECV_DATA_CFG aRxPkt[TB640_SS7_MTP3_MAX_FRAME_IN_MESSAGE];
    TBX_UINT8                    aun8Payload [1];
} TB640_EVT_SS7_MTP3_NOTIF_RECEIVED_DATA,
*PTB640_EVT_SS7_MTP3_NOTIF_RECEIVED_DATA;
```

Structure contains the receive data parameters for one frame in the message:

```
typedef struct _TB640_SS7_MTP3_RECV_DATA_CFG
{
    TB640_SS7_MTP3_MESSAGE_PRIORITY MessagePriority;
    TBX_UINT32                    un32Sls;
    TB640_SS7_POINT_CODE          Opc;
    TB640_SS7_POINT_CODE          Dpc;
    TBX_UINT32                    un32FrameOffset;
    TBX_UINT32                    un32PayloadSize;
} TB640_SS7_MTP3_RECV_DATA_CFG, *PTB640_SS7_MTP3_RECV_DATA_CFG;
```

General explanation of the field of event notification received data:

- The userpart handle parameter (*hMtp3Userpart*) specifies the handle of the MTP3 userpart instance.
- The number of frame (*un32NbFrameInMsg*) indicates the number of frames included in the *aun8Payload*.
- The array of frame received (*aRxPkt*).
- The priority of the message parameter (*MessagePriority*).
- The SLS number (*un32Sls*) indicates the signaling link selector in the original message.
- The originating point code parameter (*Opc*).
- The destination point code parameter (*Dpc*).

- The frame offset parameter (*un32FrameOffset*), specifies the offset from the start of the *aun8Payload* (from the TB640_EVT_SS7_MTP3_NOTIF_RECEIVED_DATA message) field to which the frame start.
- The payload size parameter (*un32PayloadSize*) specifies the byte count of the frame in the *aun8Payload*.
- The payload parameter (*aun8Payload*) contains all received data frames.

6 ISUP

6.1 Overview

Starting forty years ago, the public telephone networks were built using analog technologies and multiple in-band signaling methods. The growth of common fixed telephony end-users and the increasing demand for more advance services quickly made old protocols such as R1 CAS, R2 CAS to become less attractive for service providers. The ISDN protocol revolution was a big step in the telecom industry going from an in-band signaling protocol (using tones, stolen bits from voice channel or dedicated channel) to an out-of-band signaling (separated from voice channels) with message based primitives fitted for digital networks. Even if ISDN was fixing many problems of older protocols, its mass deployment in core networks never really occurred thus leaving providers with a complex hierarchy of equipments interworking with each other (or at least trying to). Although ISDN offered an easier way to provide supplementary services (i.e. call hold, call transfer) to the end-users, ISDN progressively migrated toward the edge of the network closer to the end-user. In addition, no equipment provider really implemented the ISDN base specifications thus creating lots of local variants and headaches to system integrators.

At that point, international consortiums had already started working on a unified way of establishing calls, to provide numerous services (including those already supported by ISDN and older protocols) and to be able to build new applications (services) without having to change the whole network architecture. SS7 (or Signaling System 7) took many years before being deployed all over the world but is now achieving what it was designed to do: offer an overall communication standard providing many services and being able to grow with public's demands. Even if local variants were still implemented and multiple revisions of the protocol occurred, this signaling protocol currently connects most major telephone networks all over the world. It also added a very critical feature to every network all over the world: robustness and fault tolerance.

Being layered in different functional groups, SS7 was built so that different nodes in the network don't need to implement every portion of the protocol stack if they don't require to. Following sections will describe more in details those layers and particularly the call control layer ISUP.

6.1.1 ISUP summary

In the SS7 architecture, every layer has very specific task and responsibilities to provide to upper layers. Looking at Figure 8, we see that an SS7 stack covers all levels of the OSI model from physical to application layer. Within this model, it is possible to replace one of more layers by another protocol as long as it provides the same functionalities. In a base call control application over a TDM network, the SS7 stack required would include layers MTP1, MTP2, MTP3 and ISUP or TUP. Let's take each of those layers one by one.

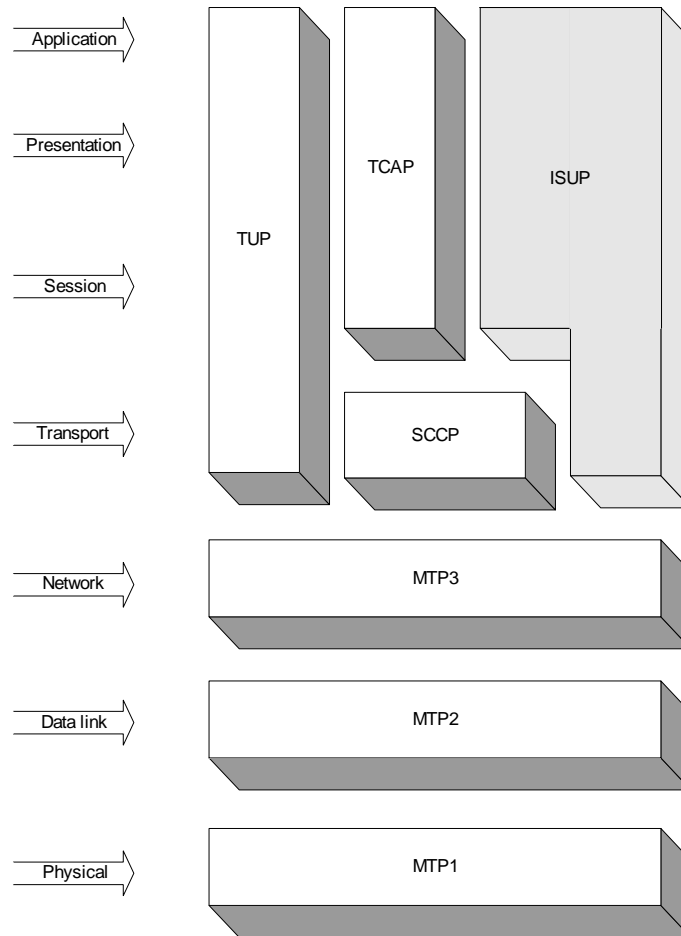


Figure 8 - ISUP OSI model

The MTP1 layer is the physical layer and is responsible of the actual HDLC framing of SS7 packets (usually called MSU – Message signaling unit). Beside the transmission and reception of frames, calculation of CRC, this layer is also responsible to monitor and report the quality of the link based on the number of failures detected (CRC errors, unexpected packets). Using specially formatted packets (LSSU and FISU), this layer makes sure the communication with the peer side is always valid and synchronized, even when the communication link is supposedly idle (no MSU being sent).

The MTP2 layer is used to create a flow control and buffering mechanism over the MTP1 layer. The flow control works both ways (toward upper layer and toward remote side) to avoid losing packets because of a lack of buffer or because of a temporary resource outage on the local/remote side (i.e CPU too busy to process packets). Because it contains the buffering scheme, it allows upper layers to retrieve buffers that were not sent in the event of a MTP1 link failure. It is also responsible to establish the connection with the peer MTP2 layer upon physical link activation.

Using the services of MTP2, the MTP3 layer really controls the networking aspect of SS7. This layer knows every reachable destination (represented by a “point code”) within the SS7 networks and knows about the path (called “routes”) to reach them. A point code can be compared to an address in the IP world. It also knows the state of every destination at any time in addition of knowing the state of every path to them. This last functionality allows this layer to perform re-

routing live, without any packet loss, in case of a link failure. This layer is the base of the SS7 redundancy architecture and makes the SS7 network fault-tolerant. This robustness however has a price which is the rigidity of the MTP3 layout. Adding a new node (point-code) to the network needs a reconfiguration of every other node that needs to talk to it. There is no automatic discovery mechanism to do that. Special nodes, called STP, in the SS7 networks are designated to act as MTP3 relay to route messages from one SS7 end-node to another. Those nodes are usually deployed in pair to allow for failure recovery of the network (see Figure 9).

At this point, stacks composed of layers MTP1-3 are only able to send packets to each other going through STPs. There is no concept of 'voice call' yet. Two layers are available to offer this functionality over MTP3: the TUP and ISUP layers. TUP is a call-control layer which is mostly used to control analog calls and equipments. The ISUP layer offers a virtual control over 'circuits' (which are really TDM timeslots) and to associate calls to them. As shown in Figure 9, those 'circuits' conveying the end-to-end voice conversation are truly separated from the signaling links that MTP1-MTP3 are transmitting onto. This allows an SS7 end-point to control equipments not even located at the same facility. Once the virtual call is establish, the application interfacing with the ISUP layer is responsible to send appropriate commands to the systems to actually connect the voice path. Thus, the responsibility of the ISUP is to negotiate the call parameters and also to negotiate with other SS7 nodes within the network the path through which TDM voice data will be connected. Of course, this is the basic function of ISUP. All other supplementary services are also support (call forward/transfer, call hold, etc). In other words, the ISUP layer does NOT actually do the TDM connections to connect both ends of a call, it only negotiate the parameters and the path. The actual connection is not the responsibility of SS7 as it is dependent on the equipment, services or even the medium. The application sitting on top of SS7 is bounded to do the connections.



ISUP is responsible to negotiate circuit assignation/reservation. It does not actually open the TDM channels to connect the physical timeslots.

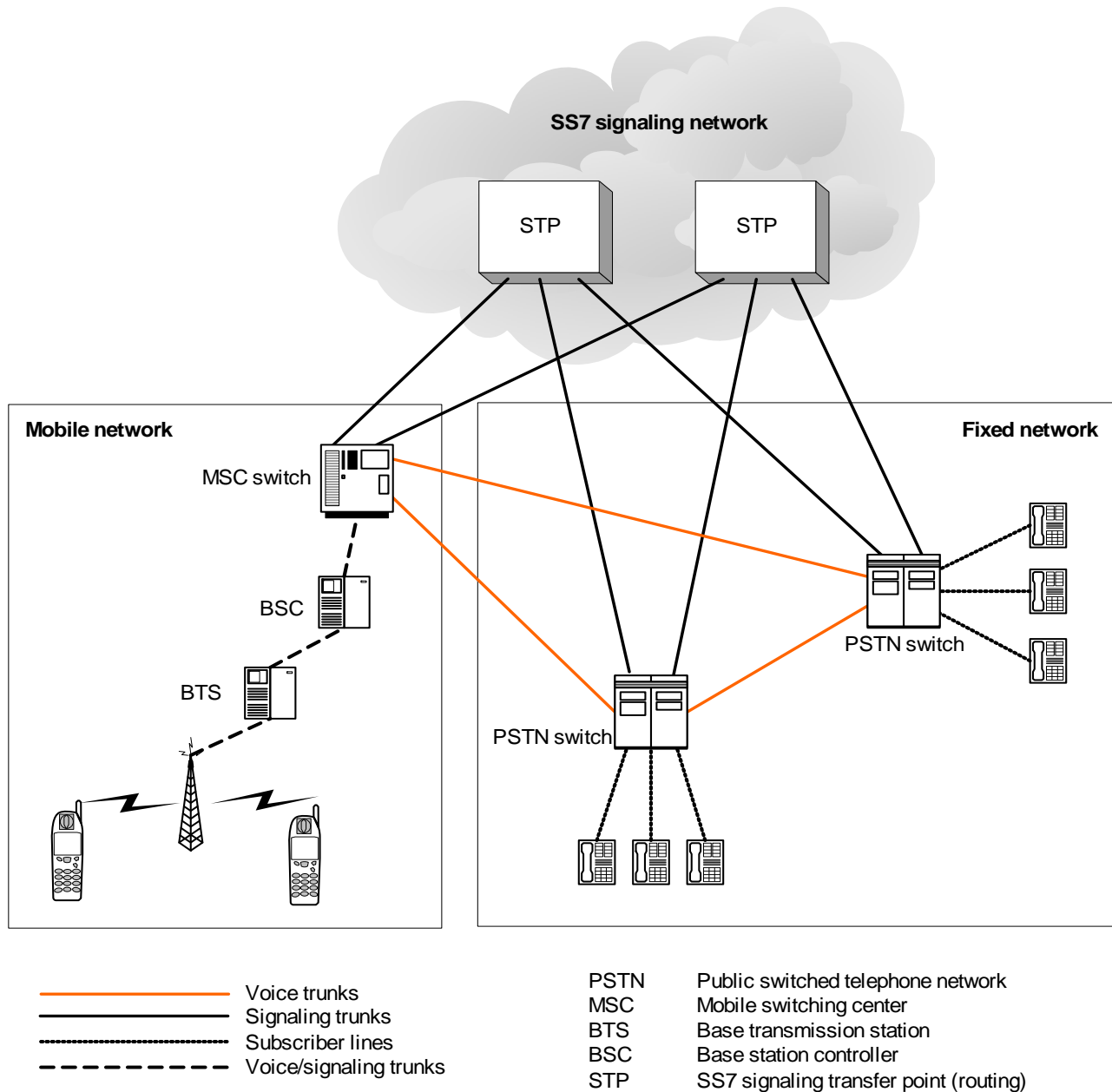


Figure 9 - Example of an SS7 network

Figure 9 shows an SS7 network containing mobile switches (for cellular phones) and fixed switch (for residential or commercial phones) that are all linked to each other through the STPs (SS7 routers). Those switches are also connected to each other with trunks dedicated to transfer voice conversation using TDM technologies. Those voice trunks are statically configured and the SS7 stack contained in every switch knows this configuration. Thus, a mobile switch SS7 stack knows how many physical timeslot it has to reach a specific PSTN switch. Every timeslot within those trunks are assigned a CIC (circuit identification code) and is referred to as a 'circuit'. It is not necessary for a switch to have a circuit toward every other switch present in the network as an SS7 call can be routed through many switches before reaching its final destination. This is where the

'negotiation' function of ISUP is so critical: to be able to manage multi-path circuit reservation, allocation and teardown.

6.1.2 Architecture within the TB640

The TB640 SS7 implementation is segmented into the various protocol layers of the SS7. The combination of those layers forms a complete SS7 stacks allowing a host application to connect to a service provider network and to provide a specific service (e.g. calling cards, conferences, etc).

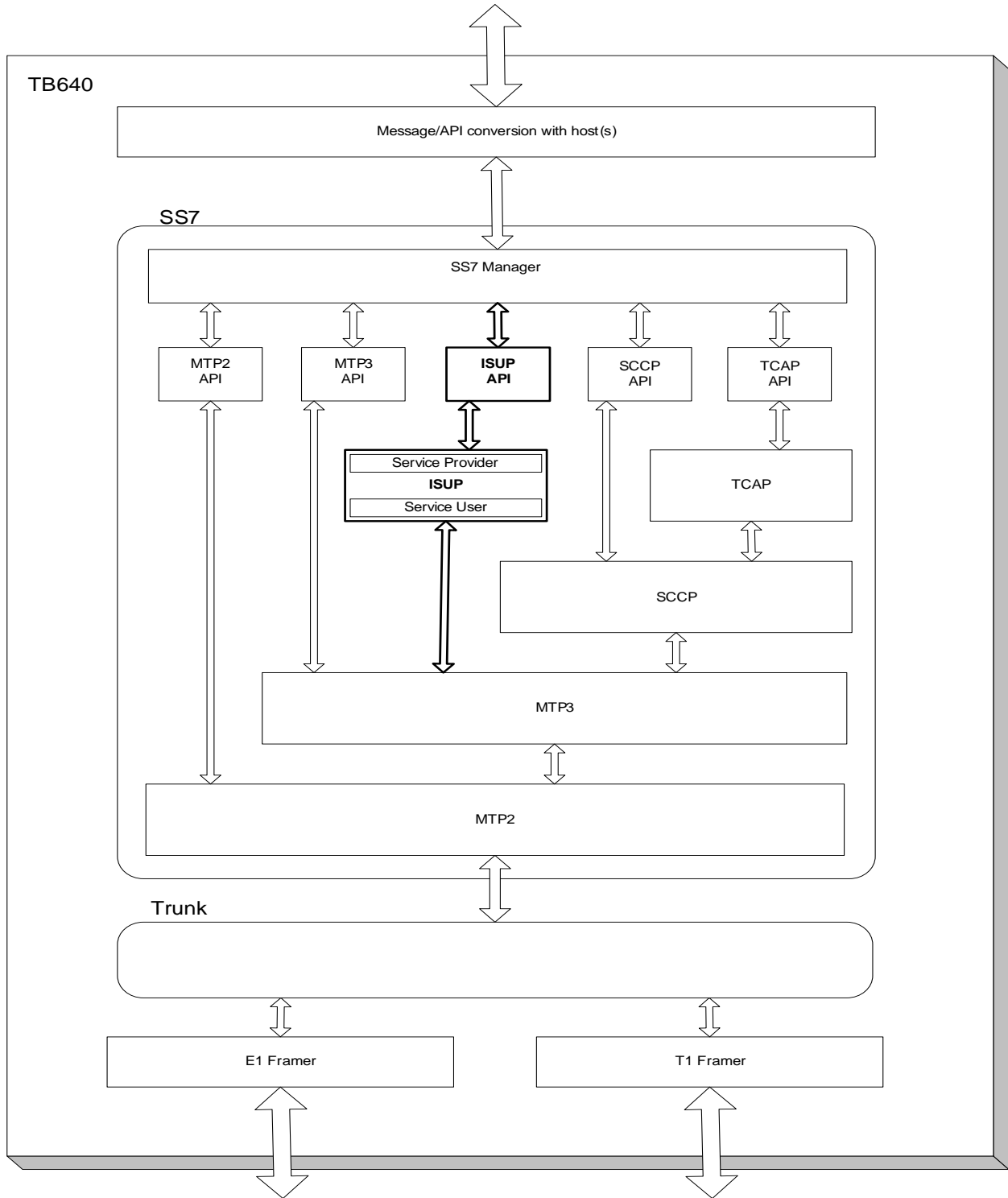


Figure 10 - ISUP layer organization within TB640

As shown in Figure 10, there is an independent API for every layer, which enables a user to either use the full potential of the stack or only a portion of it. For example, a customer already having a host implementation of ISUP could directly link it to the TB640 MTP3 and MTP2 layers through the appropriate APIs.

Assuming the customer wants to use the complete SS7 stack, the host application needs to configure the underlying layers starting from the ones closer to physical interface first. In a typical scenario, the user application would first configure the MTP2 layer followed by the MTP3 layer and then followed by the ISUP layer. This sequence does not prevent the application to dynamically add new configuration to the stack while the system is running but it does enforce a sequence to do it. The APIs are structured as such that handles need to be passed from one layer (e.g. MTP2 link handle) to the other (e.g. MTP3 link resource) during resource allocation. Thus, the user application does not really have a choice to allocate the resources in the dependency order. Those dependencies are always from upper layers toward lower layers (ISUP depends on MTP3 resources while MTP3 depends on MTP2 resources). The reader should refer to the MTP2 and MTP3 sections in this user guide to get a more precise description of capabilities and configuration for those specific layers.

As described before, an SS7 node has an ‘address’ within the SS7 network to be accessible from other nodes. This address is called a point code and is mostly defined by the MTP3 layer as it is inherently part of the ‘network protocol layer’ of an OSI model. This point code concept is also present in ISUP since it is used to specify with which SS7 node the call is to be established. The destination point code for a call is not directly related to the ‘called number’ since the destination node is rather a service box (i.e. switch) knowing what to do with a call to that specific ‘called number’. So, at this point, we know an SS7 node as its own point code (called OPC – Origin point code) and that remote nodes also have point codes (called DPC – Destination point code). There is no limitation in a single SS7 node to have more than one OPC. We’ll see later that an OPC has a limitation in terms of circuits it can handle due to the nature of the ISUP standard itself.



SS7 addresses are called ‘point-codes’. An OPC (origin point code) represents the current SS7 node and a DPC (destination point code) represents another SS7 node accessible in the SS7 network.


Since ISUP calls are established between two ISUP layers “talking” to each other, we can say that ISUP is a peer-to-peer protocol on a per-call basic (since a single ISUP layer can talk to different SS7 nodes depending on the DPC used). As in any standardized protocol, there are multiple revisions of ISUP and those are not always fully compatible between them. Therefore, two ISUP layers talking to each other must understand a common language. This language is known as the protocol variant (i.e. ANSI 92, ANSI 95, ITU, etc). So, at this point, there is three characteristic that need to be taken in account by the ISUP layer for call establishment: OPC, DPC and variants.




For two SS7 nodes to have their ISUP layers to exchange with each other, they must share the same protocol variant.

There is another characteristic that needs to be known for a call to be successful. Often in this guide, we refer to ‘the SS7 network’. A network is defined as an agglomeration of nodes that can understand and address each other. With the three characteristics previously mentioned, there is still a confusion possibility. Let’s imagine a customer who wants to build a SS7 gateway node to route calls from a provider’s network to another provider’s network. Both of those providers may have used the same ISUP variants (say ANSI 92) and may have assigned the same point code to multiple nodes in their respective networks. This works fine in two separate network but will fail if

you try to join two those networks together. How can a single point code refer to two different nodes at the same time (it is equivalent to have an IP address collision). This is why the gateway SS7 stack needs to consider those two networks separately. Thus, the ISUP layer needs a fourth characteristic to properly establish calls: a network ID.

 An SS7 network is a collection of SS7 nodes that have unique point codes.

The last remaining characteristic required to establish a call is to associate a physical timeslot to carry the voice conversation (or data) on top of the “virtual” ISUP call. As mentioned before, an ISUP layer knows to which remote node it is talking. As shown in Figure 9, the layer also knows which circuits (timeslots) are available toward that destination node as well. Combining all of those characteristics (OPC, DPC, variant, network ID and circuit ID), the ISUP layer can establish a call to any specific remote ISUP node in any network.

 A CIC (circuit identification code) represents a physical timeslot between two SS7 nodes. Both of those nodes must commonly agree upon CIC numbering scheme.

6.1.3 Configuration sequence

Within the TB640 ISUP layer, there are four resource categories that need to be instantiated in order to have a working layer (refer to Figure 11). Those four resources correspond to characteristics mentioned earlier. The user application must first create an instance of an ISUP network (see definition of a network in section 6.1.2) and will be used to reference a set of SS7 nodes having unique point code values.

Once the ISUP network resource handle is available, the application must create an ISUP userpart instance which contains the switch variant it wishes to interface with. A single network handle can be used by multiple ISUP userparts at the same time (thus it is possible for a single OPC to support multiple ISUP variants toward different DPCs).

 A single SS7 node can support multiple switch variants (as long as OPC/DPC/network combination differs from one variant to the other).

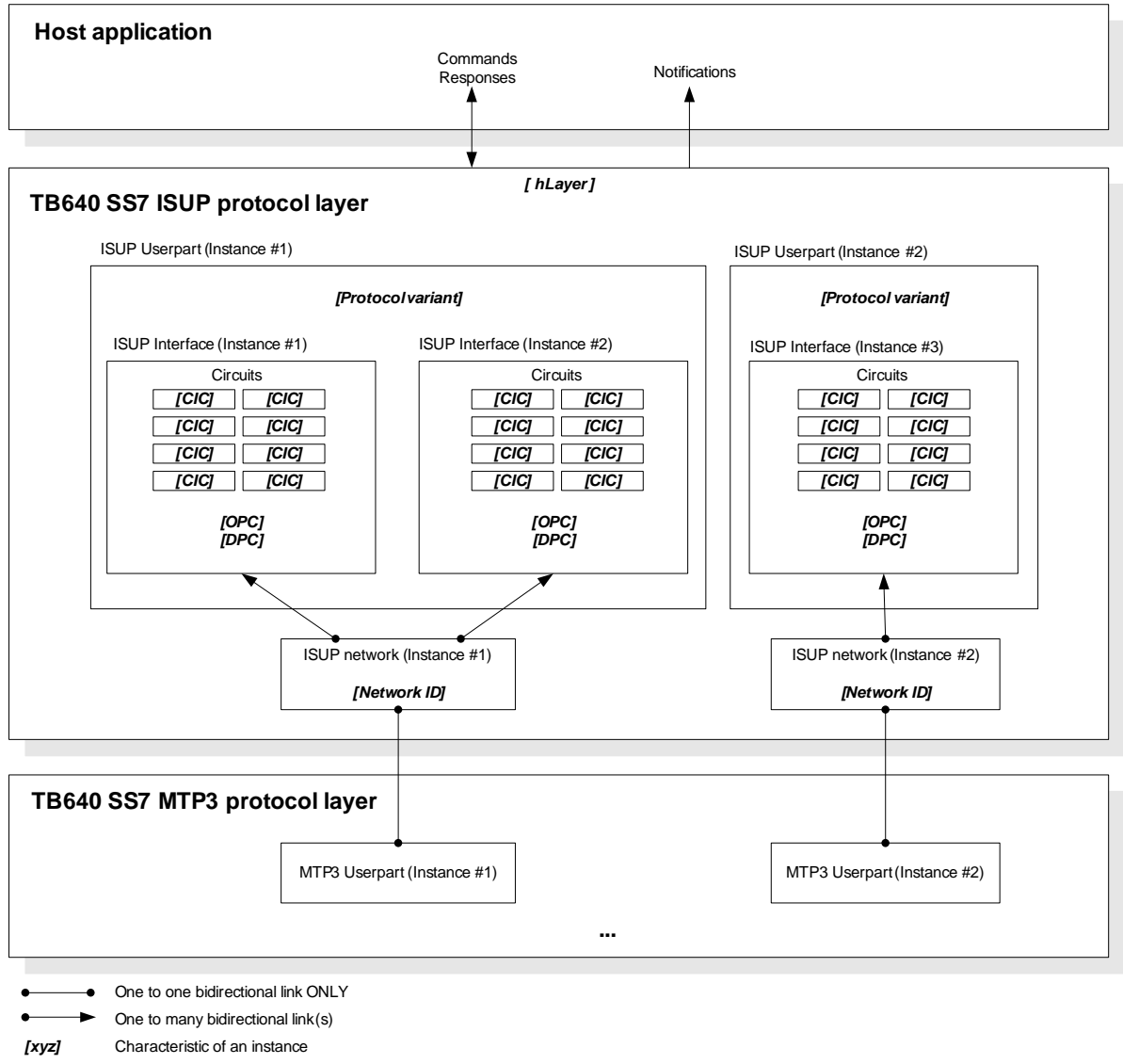


Figure 11 - ISUP layer hierarchy

Once an ISUP userpart handle is available, the user application needs to instantiate one or multiple ISUP interface(s) associated with the userpart. The interface is an association of OPC and DPC which will have circuits assigned to it. This association is required because a CIC value has a limited range. This range is determined by the ISUP standard used (i.e. ANSI/ITU/etc) which varies from 12 bits (base ITU), 14 bits (base ANSI) or 16 bits (ITU and ANSI extended). Thus, to build very large system (more than $2^{(12, 14, 16)}$ circuits linking two specific SS7 nodes) using the same ISUP switch variant, the network administrator needs to define multiple point codes for a single node. The user application would then have to create other interfaces to which the new circuits would be attached. So, logically, once an interface is instantiated, the new step is to instantiate circuits (identified by a unique CIC within an interface).





Multiple ISUP interfaces for a same network are required if...

- i. two SS7 nodes are linked with an high number of circuits (larger than what the protocol can support in an MSU)

- ii. an SS7 node is linked to multiple DPCs (at least 1 interface per OPC/DPC is required).
- iii. if more than one OPC is supported by the SS7 node (at least 1 interface per OPC/DPC is required)

During this last step (circuit allocation), the TB640 ISUP API requires the user application to provide a unique number (handle) that will be used to precisely identify the network/OPC/DPC/CIC combination. That host-provided handle is called a “circuit ID” and will be used afterward for all other messages of the ISUP API (making and receiving calls, etc). This 32 bits handle is not interpreted by the TB640 and, thus, could be anything meaningful to the host application (index, hash table key, etc).

 A “circuit ID” is a 32 bits, host-application, provided number to represent a network/OPC/DPC/CIC combination.

 A “circuit ID” is not the same as a circuit identification code (CIC).

6.1.4 Features of the TB640 ISUP layer

The ISUP protocol layer supports for following features:

- Supports multi-rate connection establishment, circuit maintenance, connection release, and abnormal conditions for multi-rate calls. Multi-rate connections are circuit switched connections requiring more than one bearer channel. This feature is part of the ANSI 95, ITU 97, and ETSI v3 specifications.
- Supports working in a decomposed media network. ISUP supports multiple variant and SS7 networks within the same stack.
- Supports the recovery of lost primitives crossing the upper, lower, and layer management interfaces
- Supports the configurable call validation testing behavior during MTP3 Resume primitive for the ANSI 95 variant.
- Supports enbloc and overlap signaling.
- Supports international and national capabilities.
- Supports link-by-link signaling using the pass-along method.
- Each link may be configured to support a variant independent of any other link. For example, one link may support ITU 1988 while another link supports ANSI 1992.
- Supports supplementary services, including user access to calling party address id, user access to called party address id, user-to-user signaling, and call forwarding.
- Supports Local Number Portability (LNP) for ANSI and Telcordia (formerly Bellcore).
- Supports multiple Originating Point Codes (OPCs).
- Support message compatibility, parameter compatibility, and wrong parameter values procedures.
- Supports circuit management procedures including blocking, unblocking, and reset.
- Supports circuit group management procedures including blocking, unblocking, reset, and query.
- Supports message segmentation procedures for ITU and ETSI variants.
- Supports configurable call clearing behavior for MTP Level 3 Pause/Resume priorities.
- Supports passing proprietary parameters transparent between upper and lower layers.

6.1.5 Specifications

The ISUP protocol layer conforms to the following standards:

- Common Channel Signaling System N.7 (National), Singapore Telecom, 1988.
- EN 300 356-1 v3.2.2, ISUP Version 3 for the International Interfaces Basic Services, 1998.
- ETS 300 356 ISDN User Part (ISUP), Version 2 for the International Interface, 1995.
- GR-317 -- CORE, Generic Requirement for Call Control Using ISDNUP. Issue 2, December 1997.
- GR-394 -- CORE, Generic Requirement for ICI Using ISDNUP. Issue 2, December 1997. Revision 1, November 1998.
- IS 7498 - Open Systems Interconnection - Basic Reference Model, ISO.
- IS 7498 DAD 1 - Open Systems Interconnection - Basic Reference Model Addendum 1: Connectionless Data Transmission, ISO.
- I.200 Guidance to the I.200 Series of Recommendations, ITU.
- I.220 Common Dynamic Description of Basic Telecommunications Services, ITU.
- I.221 Common Specific Characteristics of Services, ITU.
- I.230 Definition of Bearer Service Categories, ITU.
- I.231 Circuit Mode Bearer Service Categories, ITU.
- I.232 Packet Mode Bearer Service Categories, ITU.
- I.240 Definition of Teleservices, ITU.
- I.250 Definition of Supplementary Services, ITU.
- I.251 Number Identification Supplementary Services, ITU.
- I.320 ISDN Protocol Reference Model, ITU.
- Q.210 Principles of Telecommunication Services Supported by an ISDN and the Means to Describe Them, ITU.
- Q.700 – Introduction
 - to ITU Signaling System No. 7, 1993.
- Q.730 - ISDN Supplementary Services, 1993.
- Q.752 - Specifications of Signaling System No. 7 - Signaling System No. 7 Management, ITU.
- Q.761 - Functional Description of the ISDN User Part of Signaling System No. 7, ITU, 1993 and 1997.
- Q.762 - General Function of Messages and Signals, ITU, 1993 and 1997.
- Q.763 - Formats and Codes, ITU, 1993 and 1997.
- Q.763 - Formats and Codes, ITU, Addendum 1, 1998
- Q.764 - Signaling Procedures, ITU, 1993 and 1997.
- Q.765 - Application Transport Mechanism, ISUP ASE Module, 1998.
- Q.766 - Performance Objectives in the Integrated Services Digital Network Application, ITU.
- Q.767 - Application of the ISUP of ITU SS7 for International ISDN Interconnections, ITU, 1991.
- SS7 Integrated Services Digital Network User Part, ANSI T1.113.1-1995.
- SS7 Integrated Services Digital Network User Part, ANSI T1.113.2-1995.
- SS7 Integrated Services Digital Network User Part, ANSI T1.113.3-1995.
- SS7 Integrated Services Digital Network User Part, ANSI T1.113.4-1995.
- T1.113 - Signaling System Number 7 - Integrated Services Digital Network (ISDN) User Part, 1988.
- T1.113 - Signaling System Number 7 - Integrated Services Digital Network (ISDN) User Part, 1992..

The TB640 ISUP API supports the following switch variants:

- ANSI 88, T1.113 (1998)
- ANSI 92, T1.113 (1992)

- ANSI 95, T1.113.1 – T1.113.4 (1995)
- ITU 92, Q.761 - Q.764 (1993)
- ITU 97, Q.761 – Q.764 (1997), Q.763 Addendum 1 (05/1998), Q.765 ISUP ASE module (05/1998)
- Q.767, ITU 1991
- Singapore Telecom, Common Channel Signaling System N.7, Singapore Telecom, 1988
- Telcordia 97, GR-317, GR-394
- CHINA ISUP - Technical Specification for national No. 7 Signalling - Integrated Service Digital Network User Part (ISUP) - YDN038.1-1999
- CHINA ISUP - Technical Specification for national No. 7 Signalling – Integrated Service Digital Network User Part (ISUP) (supplementary) – Supplementary Amendment YDN 038-1997.
- ETSIv2 - ETS 300 356 (1995).
- ETSIv3 - EN 300 356-1 V3.2.2 (1998).
- UK ISUP - ND 1007:2006/04 TSG/SPEC/007.

6.2 ISUP Configuration

6.2.1 Layout of the ISUP network

As explained in section 6.1.3, the configuration of an SS7 ISUP layers depends on the setup of the network itself. In the case presented by Figure 12, the different nodes would be configured differently. All of those nodes are part of the same network as they are attached to the same STP and all have unique OPCs amongst all nodes.

The SS7 node on the left of Figure 12 (OPC 1.0.1) would require one ISUP network instance, one ISUP userpart (we assume all nodes have the same protocol variant), two ISUP interfaces (one for each destination DPC) and a certain number of ISUP circuit instances associated to each interface. Those circuits correspond to physical TDM timeslot available between the two peer nodes. Note that the CIC values are unique only within a specific interface. For example CIC=100 means a different TDM trunk/timeslot when used between 1.0.1/1.0.2 and 1.0.1/1.0.4.



A CIC (Circuit identification code) is unique only within an interface.

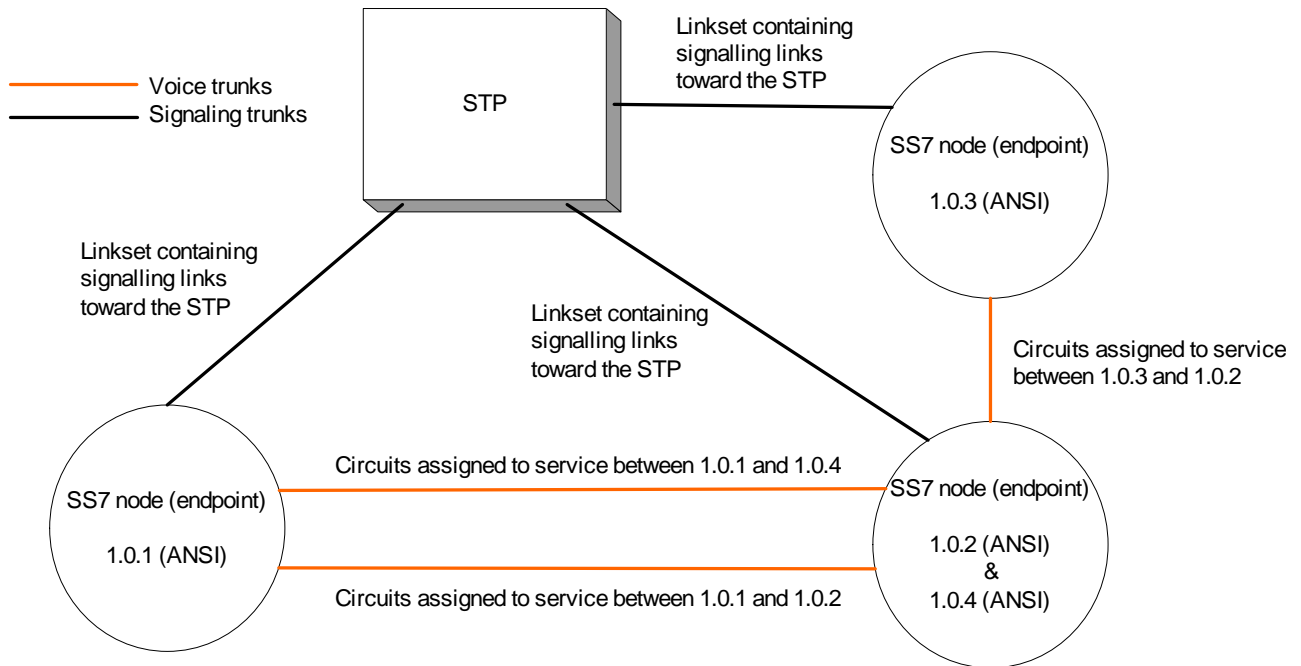


Figure 12 - Circuit setup between multiple SS7 nodes

The SS7 node located on the lower-right of Figure 12 (OPC 1.0.2 and 1.0.4) would require one ISUP network instance, one ISUP userpart, three interfaces (one for each destination DPC) and a certain number of ISUP circuit instances associated to each interface.

The SS7 node located on the upper-right of Figure 12 (OPC 1.0.3) would require one ISUP network instance, one ISUP userpart, one interface and a certain number of ISUP circuit instances associated to that interface.

6.2.2 Configuration of layer


As mentioned in section 6.1.3, the configuration of the ISUP layer (or addition of new elements to an existing layer) must follow a specific sequence. This sequence is enforced by the API through the use of handles assigned upon allocation of the different required resources. The following sections will detail the different resource configuration, their parameters and their respective use.


General guidelines for configuration of ISUP for a proper operation include the following:

1. The ISUP general allocation⁶ (TB640_MSG_ID_SS7_ISUP_OP_ALLOC) must precede all other messages (other configuration ISUP allocation, get, states and stats). The response of this message is an ISUP handle.
2. The ISUP Network allocation TB640_MSG_ID_SS7_ISUP_OP_NETWORK_ALLOC) must be made (with the ISUP handle from **step 1**) for a particular network. The response of this message is an ISUP network handle.

⁶ All fields of a configuration (allocation) message must be filled unless explicitly optional or not defined for certain variants.

3. The ISUP userpart allocation (TB640_MSG_ID_SS7_ISUP_OP_USERPART_ALLOC) must be made (with the ISUP handle from **step 1**) for a particular switch variant. The response of this message is an ISUP userpart handle.
4. The ISUP interface allocation (TB640_MSG_ID_SS7_ISUP_OP_INTERFACE_ALLOC) must be made (with the ISUP handle from **step 1**, network handle from **step 2** and the userpart handle from **step 3**) for a particular OPC/DPC. The response of this message is an ISUP interface handle.
5. The ISUP circuit allocation (TB640_MSG_ID_SS7_ISUP_OP_CIRCUIT_ALLOC) must be made (with the ISUP handle from **step 1** and the interface handle from **step 4**). The response of this message is an ISUP circuit handle associated with a host-provided handle called the “circuit ID”.

 For any reconfiguration with an ISUP (TB640_MSG_ID_SS7_ISUP_OP_xyz_SET_PARAMS) message, all reconfigurable parameters must be filled appropriately even if the intention is to modify only single parameter unless specified otherwise.

 When a reconfiguration is requested through a ISUP API, the effect may not be observed immediately. This is especially true for timers and threshold as the targeted event might already be active when the reconfiguration occurs.

6.2.2.1 General

The TB640_MSG_ID_SS7_ISUP_OP_ALLOC (request/response) message is used to initialize the general parameters of the ISUP layer. These parameters are common to every other instance created afterward (i.e. networks, userparts, interfaces, circuits).

The **request** part of the message TB640_MSG_SS7_ISUP_OP_ALLOC contains the field:

```
...
TB640_SS7_HANDLE    hLayer; /* Contains the layer handle from system manager
                        Module */
TB640_SS7_ISUP_CFG  Cfg;    /* Contains the configuration of the ISUP layer */
...
```

This structure contains the general configuration parameters for ISUP layer:


```
typedef struct _TB640_SS7_ISUP_CFG
{
    TBX_UINT32          un32StructVersion;
    TBX_UINT32          un32T18Timer;
    TBX_UINT32          un32T19Timer;
    TBX_UINT32          un32T20Timer;
    TBX_UINT32          un32T21Timer;
    TBX_UINT32          un32T22Timer;
    TBX_UINT32          un32T23Timer;
    TBX_UINT32          un32T28Timer;
    TBX_UINT32          un32FGRTimer;
    TBX_UINT8           aun8Padding0 [4];
}
```


```
} TB640_SS7_ISUP_CFG, *PTB640_SS7_ISUP_CFG;
```


The following enumeration lists the different configuration parameters and their description. Unless specifically noted, all parameters are reconfigurable:

- The general timer configuration parameters. All timer values must be expressed in milliseconds. The timers have the following definitions:

un32T18Timer:	GROUP BLOCKING message sent. Terminated normally when GROUP BLOCKING ACKNOWLEDGMENT is received. Typical values are 15 to 60 seconds.
un32T19Timer:	INITIAL GROUP BLOCKING message sent. Terminated normally when GROUP BLOCKING ACKNOWLEDGMENT is received. Typical values are 5 to 15 minutes.
un32T20Timer:	GROUP UNBLOCKING message sent. Terminated normally when GROUP UNBLOCKING ACKNOWLEDGMENT is received. Typical values are 15 to 60 seconds.
un32T21Timer:	INITIAL GROUP UNBLOCKING message sent. Terminated normally when GROUP UNBLOCKING ACKNOWLEDGMENT is received. Typical values are 5 to 15 minutes.
un32T22Timer:	GROUP RESET sent timer. Terminated normally at the receipt of the acknowledgment. Typical values are 15 to 60 seconds.
un32T23Timer:	INITIAL GROUP RESET sent timer. Terminated normally when acknowledgment is received. Typical values are 5 to 15 minutes.
un32T28Timer:	CIRCUIT GROUP QUERY sent timer. Terminated normally when CIRCUIT GROUP QUERY RESPONSE message is received. When this alarm expires, an alarm is generated. Typical value is 10 seconds.
un32FGRTimer:	ANSI and Telcordia FIRST GROUP received timer. A typical value is 5 seconds.

 ITU variants do not support un32FGRTimer. Therefore, the timer needs not to be configured for these variants.

 UK: The following timers are not required for the UK: T28

 The macro TB640_SS7_ISUP_SET_DEFAULT_ISUP_CFG is available to set default values into the configuration structure. It is highly recommended to use the macro to properly set all elements of the structure and then change individual settings according to the user application.

The **response** part of the message TB640_MSG_ID_SS7_ISUP_OP_ALLOC contains the field:

```
...
TB640_SS7_ISUP_HANDLE      hIsup;          /* Handle of the initialized ISUP layer */
...
```

6.2.2.2 Network

The TB640_MSG_ID_SS7_ISUP_OP_NETWORK_ALLOC (request/response) message is used to initialize (create) an instance of an ISUP network. Creating such entity tells the ISUP protocol layer about a collection of SS7 nodes that are accessible through MTP3.

The **request** part of the message TB640_MSG_SS7_ISUP_OP_NETWORK_ALLOC contains the field:

```
...  
TB640_SS7_ISUP_HANDLE          hIsup; /* Handle of the ISUP layer */  
TB640_SS7_ISUP_NETWORK_CFG    Cfg;   /* Contains the configuration of a ISUP  
                                   network instance */  
...
```

This structure contains the configuration parameters for ISUP network instance:


```
typedef struct _TB640_SS7_ISUP_NETWORK_CFG  
{  
    TB640_SS7_UID                UidIsupNetwork;  
    TB640_SS7_UID                UidMtp3Userpart;  
    TBX_UINT32                   un32NetworkId;  
    TB640_SS7_SUBSERVICE_FIELD_TYPE SsfType;  
}  
TB640_SS7_ISUP_NETWORK_CFG, *PTB640_SS7_ISUP_NETWORK_CFG;
```

The following enumeration lists the different configuration parameters and their description. Unless specifically noted, all parameters are not reconfigurable:

- The unique identifier ISUP network parameter (*UidIsupNetwork*) specified the unique ISUP network Id for a system SS7.
- The unique identifier MTP3 userpart parameter (*UidMtp3Userpart*) specified the UID of the MTP3 userpart to which this network is connected.
- The network ID parameter (*un32NetworkId*) is a unique identifier that is used by the ISUP layer to differentiate between multiple MTP3 networks. This number is only used locally and is not included in any MSUs sent or received on the SS7 network.
- The subservice field type parameter (*SsfType*) specified the type of sub-service for a specific user-part.

The **response** part of the message TB640_MSG_SS7_ISUP_OP_NETWORK_ALLOC contains the field:

```
...  
TB640_SS7_ISUP_NETWORK_HANDLE  hIsupNetwork; /* Handle of the instance of the  
                                           ISUP network */  
...
```


 As specified in section 6.1.3, an ISUP network can be used by multiple ISUP interfaces even if they are members of different ISUP userpart (e.g. to support multiple switch variants for a single SS7 network).

6.2.2.3 Userpart

The TB640_MSG_ID_SS7_ISUP_OP_USERPART_ALLOC (request/response) message is used to initialize (create) an instance of an ISUP userpart. A userpart can easily be seen as a protocol variant that a host application wants to use on the SS7 network.

The **request** part of the message TB640_MSG_SS7_ISUP_OP_USERPART_ALLOC contains the field:

```

...
TB640_SS7_ISUP_HANDLE          hIsup; /* Handle of the ISUP layer */
TB640_SS7_ISUP_USERPART_CFG    Cfg;     /* Contains the configuration of the ISUP
...
                                     userpart */

```

This structure contains the configuration parameters for a ISUP userpart instance:

```

typedef struct _TB640_SS7_ISUP_USERPART_CFG
{
    TBX_UINT32                un32StructVersion;
    TB640_SS7_ISUP_PROTOCOL_VARIANT    ProtocolVariant;
    TB640_SS7_SUBSERVICE_FIELD_TYPE    SsfType;
    TBX_BOOL                  fCallingNbInsertion;
    TBX_BOOL                  fCallingNbVerification;
    TB640_SS7_ISUP_ADDRESS_INDICATOR_TYPE    CallingNbType;
    TB640_SS7_ISUP_NUMBERING_PLAN    NumberingPlan;
    TBX_CHAR                  szSig [TB640_SS7_ISUP_MAX_SID_DIGIT];

    TBX_BOOL                  fForceSidPresIndRestricted;
    TBX_BOOL                  fForceIncomingSidPresRestricted;
    TBX_BOOL                  fOutgoingSidPresRestricted;
    TBX_BOOL                  fRequestCallingNb;
    TBX_BOOL                  fCallModificationAllowed;
    TBX_UINT8                un8MaxLenUserToUserMessages;
    TBX_UINT8                aun8Padding0 [3];
    TBX_BOOL                  fAllowPassOn;
    TB640_SS7_ISUP_RELEASE_LOCATION    ReleaseLocation;
    TBX_UINT32                un32T1Timer;
    TBX_UINT32                un32T2Timer;
    TBX_UINT32                un32T5Timer;
    TBX_UINT32                un32T6Timer;
    TBX_UINT32                un32T7Timer;
    TBX_UINT32                un32T8Timer;
    TBX_UINT32                un32T9Timer;
    TBX_UINT32                un32T27Timer;
    TBX_UINT32                un32T31Timer;
    TBX_UINT32                un32T33Timer;
    TBX_UINT32                un32T34Timer;
    TBX_UINT32                un32T36Timer;
    TBX_UINT32                un32CCrTimer;
    TBX_UINT32                un32ExTimer;
    TBX_UINT32                un32CCRTimer;
    TBX_UINT32                un32CRMTimer;
    TBX_UINT32                un32CRATimer;

```

```

TBX_UINT32          un32ECTTimer;
TBX_UINT32          un32RELRSPTimer;
TBX_UINT32          un32FNLRELRSPTimer;
TBX_UINT8           aun8Padding1 [4];
} TB640_SS7_ISUP_USERPART_CFG, *PTB640_SS7_ISUP_USERPART_CFG;

```

The following enumeration lists the different configuration parameters and their description. Unless specifically noted, all parameters are reconfigurable:

- The protocol variant parameter (*ProtocolVariant*) is used specify which protocol is going to be used to interact with peer ISUP layers. This parameter is not reconfigurable. Allowed values are:

Table 37 - ISUP protocol variants

Protocol variant	Description
TB640_SS7_ISUP_PROTOCOL_VARIANT_ANSI88	Protocol follows specification “T1.113 (1998)”
TB640_SS7_ISUP_PROTOCOL_VARIANT_ANSI92	Protocol follows specification “ T1.113 (1992)”
TB640_SS7_ISUP_PROTOCOL_VARIANT_ANSI95	Protocol follows specifications “T1.113.1 – T1.113.4 (1995)”
TB640_SS7_ISUP_PROTOCOL_VARIANT_TELCORDIA	Protocol follows specifications “GR-317 and GR-394”
TB640_SS7_ISUP_PROTOCOL_VARIANT_ITU	Protocol follows specifications “Q.761 - Q.764 (1993)”
TB640_SS7_ISUP_PROTOCOL_VARIANT_ITU97	Protocol follows specifications “Q.761 – Q.764 (1997), Q.763 Addendum 1 (05/1998), Q.765 ISUP ASE module (05/1998) Q.767, ITU 1991”
TB640_SS7_ISUP_PROTOCOL_VARIANT_SINGAPORE	Protocol follows specification “Singapore Telecom, Common Channel Signaling System N.7, Singapore Telecom, 1988”
TB640_SS7_ISUP_PROTOCOL_VARIANT_Q767	Protocol follows specification “Q.767 (1991)”
TB640_SS7_ISUP_PROTOCOL_VARIANT_NTT	Available on demand only.

- The sub-service field parameter (*SsfType*) is used to specify which type of network the protocol layer stands on (national or international). The SSF parameter is included in every ISUP MSUs sent and received from the SS7 network. Filtering according to this field can be activated/deactivated using the MTP3 general configuration parameter *fSSFValidation*. This parameter is not reconfigurable. Allowed values are:

Table 38 - ISUP SSF values

Sub-service field	Description
TB640_SS7_SUBSERVICE_FIELD_TYPE_INTERNATIONAL	Messages sent by the layer will concern international traffic.
TB640_SS7_SUBSERVICE_FIELD_TYPE_NATIONAL	Messages sent by the layer will concern national traffic.
TB640_SS7_SUBSERVICE_FIELD_TYPE_NAT_RESERVED	Reserved for national use.

- The automatic calling number insertion parameter (*fCallingNbInsertion*) is used to instruct the ISUP userpart to insert the “calling party number IE” automatically in outgoing IAM MSUs if the host request does not already includes it. The default data is taken from the station ID configuration parameter *szSig*.

- The calling number verification parameter (*fCallingNbVerification*) is used to instruct the ISUP userpart to validate if all calling party numbers IEs received from the host are equal to the default data from the station ID configuration parameter *szSig*. If the IE is not identical, the call will be refused by the ISUP layer.
- The calling address indicator parameter (*CallingNbType*) is used to identify the type of number used in the station ID parameter *szSig*. This value is mapped to the “Nature of address indicator” field of the “called party number” IE (refer to specification Q.763E, section 3.10 b). Allowed values are:

Table 39 - ISUP calling address indicator values

Calling address indicator	Description
TB640_SS7_ISUP_ADDRESS_INDICATOR_TYPE_NOT_PRESENT	Used only when the default station ID is not configured
TB640_SS7_ISUP_ADDRESS_INDICATOR_TYPE_SUBSCRIBER_NUMBER	Number represents a subscriber number (national use).
TB640_SS7_ISUP_ADDRESS_INDICATOR_TYPE_NATIONAL_NUMBER	Number represents a national (significant) number.
TB640_SS7_ISUP_ADDRESS_INDICATOR_TYPE_INTERNATIONAL_NUMBER	Number represents an international number

- The numbering plan parameter (*NumberingPlan*) is used to identify the numbering scheme for the station ID parameter *szSig*. This value is mapped to the “Numbering plan indicator” field of the “called party number” IE (refer to specification Q.763E, section 3.9 d). Allowed values are:

Table 40 - ISUP numbering plan values

Calling numbering plan	Description
TB640_SS7_ISUP_NUMBERING_PLAN_UNKNOWN	Unknown numbering scheme
TB640_SS7_ISUP_NUMBERING_PLAN_ISDN	ISDN/Telephony numbering according to E.164/E.163
TB640_SS7_ISUP_NUMBERING_PLAN_TEL	Telephony numbering according to E.163
TB640_SS7_ISUP_NUMBERING_PLAN_DATA	Data numbering according to X.121 (national use)
TB640_SS7_ISUP_NUMBERING_PLAN_TELEX	Telex numbering according to F.69 (national use)
TB640_SS7_ISUP_NUMBERING_PLAN_NATIONAL	National standard numbering
TB640_SS7_ISUP_NUMBERING_PLAN_PRIVATE	Private numbering
TB640_SS7_ISUP_NUMBERING_PLAN_EXT	Reserved for extension

- The station ID parameter (*szSig*) contains the default “calling party number” signals that is used in automatic CID insertion (see *fCallingNbInsertion*) and calling number verification (see *fCallingNbVerification*). The digit string is coded in ASCII and will be converted to BCD encoding by the stack as specified in Q.763E, section 3.10 g).
- The presentation restricted indication parameter (*fForceSidPresIndRestricted*) instructs the ISUP userpart to force (set) the presentation restricted flag of the calling party number IE of all incoming call events.
- The presentation restricted incoming parameter (*fForceIncomingSidPresRestricted*) instructs the ISUP userpart to force (set) the presentation restricted flag of the calling party number IE of all incoming IAM MSUs.

- The presentation restricted outgoing parameter (*fOutgoingSidPresRestricted*) instructs the ISUP userpart to force (set) the presentation restricted flag of the calling party number IE of all outgoing IAM MSUs.
- The request calling number parameter (*fRequestCallingNb*) instructs the ISUP userpart to send an INR MSUs when it receive incoming call where there is no calling number IE. This option must be disabled for variants not supporting INR (i.e. NTT).
- The call modification parameter (*fCallModificationAllowed*) is used for the ISUP userpart to allow for call modification.
- The maximum user to user message length parameter (*un8MaxLenUserToUserMessages*) is used to set the maximum size (in bytes) allowed in MSUs. If set to zero, the user-to-user information is removed from all MSUs. Allowed values are from 0 to 255.
- The pass-on allow parameter (*fAllowPassOn*) is used instruct the ISUP layer to allow unrecognized MSUs to be passed up or down. This field is not used by ANSI variants.
- The release location parameter (*ReleaseLocation*) is used to fill the release location indicator of the cause IE when the ISUP layer automatically releases calls (i.e. when there is a timer expiry). Allowed values are:


Table 41 - ISUP release locations


Release location	Description
TB640 SS7 ISUP RELEASE LOCATION USER	User
TB640 SS7 ISUP RELEASE LOCATION PRIVNETLU	Private network serving local user
TB640 SS7 ISUP RELEASE LOCATION PUBNETLU	Public network serving local user
TB640 SS7 ISUP RELEASE LOCATION TRANNET	Transit network
TB640 SS7 ISUP RELEASE LOCATION PUBNETRU	Public network serving remote user
TB640 SS7 ISUP RELEASE LOCATION PRIVNETRU	Private network serving remote user
TB640 SS7 ISUP RELEASE LOCATION LOCALIF	Local interface controlled by this link
TB640 SS7 ISUP RELEASE LOCATION INTNET	International network
TB640 SS7 ISUP RELEASE LOCATION NETINTER	Network beyond internetworking point
TB640 SS7 ISUP RELEASE LOCATION NOINFOAV	No information

- The timer configuration parameters. All timers are reconfigurable. All timer values must be expressed in milliseconds. The timers have the following definitions:
 - un32T1Timer: RELEASE message sent. Terminated normally at the receipt of an RLC message. Typical values are 15 to 60 seconds.
 - un32T2Timer: SUSPEND message receive by controlling exchange. Terminated normally when RESUME (user) is received at the controlling exchange. Typical value is 3 minutes.
 - un32T5Timer: INITIAL RELEASE message sent. Terminated normally at the receipt of the RLC message. Typical values are 5 to 15 minutes.

un32T6Timer:	SUSPEND (network) received by controlling exchange. Terminated normally at the receipt of RESUME (network) or RELEASE. Typical value is 1 second.
un32T7Timer:	LATEST ADDRESS message sent. Terminated normally when the condition for normal release of address and routing information is met. Typical values are 20 to 30 seconds.
un32T8Timer:	INITIAL ADDRESS message received by incoming international exchange. Typical values are 10 to 15 seconds.
un32T9Timer:	LATEST ADDRESS message sent by the outgoing international exchange. Terminated normally at the receipt of answer. Typical values are 90 to 180 seconds.
un32T27Timer:	Waiting for continuity rechecks. Terminated normally when the CONTINUITY CHECK REQUEST message is received. . Typical value is 4 minutes.
un32T31Timer:	Release of ISUP signaling connection based on connection oriented SCCP. Call reference is frozen during this period and is reusable after the timer expiry. Typical values are more than 6 minutes.
un32T33Timer:	Information request sent. Terminated normally when INF message is received. Typical values are 12 to 15 seconds.
un32T34Timer:	Waiting for continuity after rechecks (ANSI88, ANSI92 and Telcordia). For the ITU segmentation timer, see <i>un32T36Timer</i> . Typical values are 2 to 4 seconds.
un32T36Timer:	Awaiting for another segmented message. For ITU-T, ETSI and FTZ, this timer substitute for the T34 (ANSI). Typical values are 10 to 15 seconds.
un32CCrTimer:	Continuity recheck timer. Typical value is 20 seconds.
un32ExTimer:	Exit to be sent (ANSI and Telcordia). The values are dependent upon the network.
un32CCRTimer:	Awaiting LPA message after having sent CCR (ANSI88, ANSI92 and Telcordia). Typical value is 2 seconds.
un32CRMTimer:	Circuit reservation sent, waiting for reservation acknowledge (ANSI92 and Telcordia). Typical values are from 3 to 4 seconds.
un32CRATimer:	Circuit reservation acknowledge sent, waiting for IAM (ANSI92 and Telcordia). Terminated normally when IAM is received.. Typical value is 10 seconds.
un32ECTimer:	Explicit call transfer, waiting for LOOP PREVENTION RESPONSE message (ETSI, ETSI V3 and ITU97). Typical value is 1 second.
un32RELRSPTimer:	Waiting for release response timer. It is used to account for loss of release response primitive. Typical values are system-dependent.
un32FNLRELRSPTimer:	Waiting for release response timer. It is used to account for the loss of release response primitive. If the release response from the host


application is not received before the expiry of this timer, circuit reset is locally initiated by the ISUP userpart. Typical values are system-dependant.

 UK: The following timers are not required for the UK: T8, T27, T31, T36. The value of T33 is 5-15s. If Timer (T33) matures, the call shall continue in the normal way.

 The macro `TB640_SS7_ISUP_SET_DEFAULT_ISUP_USERPART_CFG` is available to set default values into the configuration structure. It is highly recommended to use the macro to properly set all elements of the structure and then change individual settings according to the user application.

The **response** part of the message `TB640_MSG_SS7_ISUP_OP_USERPART_ALLOC` contains the field:

```
...  
TB640_SS7_ISUP_USERPART_HANDLE    hIsupUserpart; /* handle of the instance of the  
                                     ISUP user-part */  
...
```

 As specified in section 6.1.3, multiple ISUP userpart can be instantiated within a single ISUP protocol layer. This allows a user application to interface with multiple SS7 ISUP protocol variants at the same time. Only one ISUP userpart is required for a specific variant as it can be assigned to multiple ISUP networks.

6.2.2.4 Interface

The `TB640_MSG_ID_SS7_ISUP_OP_INTERFACE_ALLOC` (request/response) message is used to initialize (create) an instance of an ISUP interface. This entity creates a binding within the ISUP layer of a particular variant, a specific network and an OPC/DPC pair to which circuits can be assigned to.

The **request** part of the message `TB640_MSG_SS7_ISUP_OP_INTERFACE_ALLOC` contains the field:

```
...  
TB640_SS7_ISUP_HANDLE              hIsup; /* Handle of the ISUP layer */  
TB640_SS7_ISUP_INTERFACE_CFG      Cfg; /* Contains the configuration of a ISUP  
                                     interface instance */  
...
```

This structure contains the configuration parameters for ISUP interface instance:

```
typedef struct _TB640_SS7_ISUP_INTERFACE_CFG  
{  
    TBX_UINT32                          un32StructVersion;  
    TB640_SS7_ISUP_USERPART_HANDLE      hIsupUserpart;  
    TB640_SS7_ISUP_NETWORK_HANDLE       hIsupNetwork;  
    TB640_TRUNK_TYPE                     TrunkType;  
    TB640_SS7_POINT_CODE                 Opc;  
    TB640_SS7_POINT_CODE                 Dpc;  
}
```

```
TBX_UINT32                un32T4Timer;
TBX_UINT32                un32PauseTimer;
TBX_UINT32                un32StaEnqTimer;
TB640_SS7_ISUP_PAUSE_ACTION  PauseAction;
TBX_BOOL                  fAvailTest;
TB640_SS7_ISUP_MULTIRATE_TABLE_CHK  MultiRateTableCheck;
TB640_SS7_ISUP_SLS_SELECTOR  SlsSelector;
TB640_SS7_ISUP_SLS_RANGE    SlsRange;


} TB640_SS7_ISUP_INTERFACE_CFG, *PTB640_SS7_ISUP_INTERFACE_CFG;
```

The following enumeration lists the different configuration parameters and their description. Unless specifically noted, all parameters are not reconfigurable:

- The ISUP userpart parameter (*hIsupUserpart*) is the handle provided by the TB640_MSG_SS7_ISUP_OP_USERPART_ALLOC message. It is used to specify within which userpart this interface instance is to be created.
- The ISUP network parameter (*hIsupNetwork*) is the handle provided by the TB640_MSG_SS7_ISUP_OP_NETWORK_ALLOC message. It is used to specify within which userpart this interface instance is to be created.
- The trunk type parameter (*TrunkType*) represents the type of digital path used between the near-end exchange and far-end exchange. This field is used when ISUP validate a received CAM parameter, or when it fills a CAM parameter and checks the Q.763 table 3 for a non-since rate connections. ISUP uses this field for ANS95, ITU97 and ETSI v3 variants. Allowed values are:

Table 42 - ISUP trunk types

Trunk type	Description
TB640_TRUNK_TYPE_E1	Digital trunk type is E1
TB640_TRUNK_TYPE_T1	Digital trunk type is T1

 In ANSI95, the trunk type can only be T1.

- The origin point code parameter (*Opc*) contains the point code (address) of the local SS7 node to which the ISUP interface is bounded to. The point code format must follow the switch variant configured in the ISUP userpart. Proper point code length is: see **Table 3 - DPC Length**.
- The destination point code parameter (*Dpc*) contains the point code (address) of the remote SS7 node to which the ISUP interface is bounded to. The point code format must follow the switch variant configured in the ISUP userpart. Proper point code length is: see **Table 3 - DPC Length**.
- The pause action parameter (*PauseAction*) instructs the ISUP interface what to do with calls when the destination SS7 node becomes ‘paused’ (not accessible). This parameter is reconfigurable. Allowed values are:

Table 43 - ISUP pause actions

Pause action	Description
TB640_SS7_ISUP_PAUSE_ACTION_CLEAR_ALL	All calls on the interface’s circuits are released

	then the destination point code becomes unavailable.
TB640_SS7_ISUP_PAUSE_ACTION_CLEAR_TRANSIENT	All calls with a non active state on the interface's circuits are released then the destination point code becomes unavailable.
TB640_SS7_ISUP_PAUSE_ACTION_CLEAR_AFTER_TIMER	All calls on the interface's circuits are released after the <i>un32PauseTimer</i> expiry.

- The availability test parameter (*fAvailTest*) tells whether or not the ISUP interface should generate a circuit validation test on a circuit upon reception of an MTP-RESUME indication. This option is only valid for ANSI95.
- The multi rate table check parameter (*MultiRateTableCheck*) tells whether or not the ISUP interface should check the Table 3 (part 1 or part 2) in ITU97 Q.763 for multi-rate connection type calls and fixed contiguous Nx64 calls. The ISUP userpart uses this field to validate the starting CIC, in case of multi-rate connection type calls and fixed contiguous Nx64 calls, if the calls begin on the specific circuits. This field is used in ITU97, UK and ETSI v3 variants. Allowed values are:

Table 44 - ISUP multi-rate check values

Multi-rate check	Description
TB640_SS7_ISUP_MULTIRATE_TABLE_CHK_TABLE3_PART1	Validate the start CIC for multi-rate calls as per Table 3 part 1 Q.763.
TB640_SS7_ISUP_MULTIRATE_TABLE_CHK_TABLE3_PART2	Validate the start CIC for fixed contiguous calls as per Table 3 part 2 Q.763.

- The SLS selector parameter (*SlsSelector*) instructs the ISUP interface about the SLS selection algorithm to use. Allowed values are:

Table 45 - ISUP SLS selector values

SLS selector	Description
TB640_SS7_ISUP_SLS_SELECTOR_LOAD_DISTRIBUTION	SLS is selected to optimize load distribution
TB640_SS7_ISUP_SLS_SELECTOR_BY_CIC	SLS is selected according to CIC

- The SLS range parameter (*SlsRange*) instructs the ISUP interface to use a defined range of SLS values. This parameter is only used for ANSI88, ANSI92, ANSI95 and Telcordia. Allowed values are:

Table 46 - ISUP SLS range values

SLS range	Description
TB640_SS7_ISUP_SLS_RANGE_5BITS	SLS values will be from 0-31
TB640_SS7_ISUP_SLS_RANGE_8BITS	SLS values will be from 0-255

- The timer configuration parameters. All timers are reconfigurable. All timer values must be expressed in milliseconds. The timers have the following definitions:


un32T4Timer: RELEASE message sent. Terminated normally at the receipt of an RLC message. Typical values are 15 to 60 seconds.

un32PauseTimer: SUSPEND message receive by controlling exchange. Terminated normally when RESUME (user) is received at the controlling exchange. Typical value is 3 minutes.

un32StaEnqTimer: Query DPC state from MTP3 layer when ISUP interface is not available.
Typical value is 15 seconds.

The **response** part of the message TB640_MSG_SS7_ISUP_OP_INTERFACE_ALLOC contains the field:

```
...
TB640_SS7_ISUP_INTERFACE_HANDLE  hIsupInterface; /* Handle of the instance of the
                                                    ISUP interface */
...
```

 As specified in section 6.1.3, an ISUP interface can only be part of a single ISUP userpart. On the other hand, multiple circuits can be attached to the same interface instance. Those circuits represent the voice channels available between the local SS7 node and the remote SS7 node.


6.2.2.5 Circuits

The TB640_MSG_ID_SS7_ISUP_OP_CIRCUIT_ALLOC (request/response) message is used to initialize (create) an instance representing one or many ISUP circuits. Since a circuit represents a physical voice (or data) channel between this SS7 node and a remote SS7 node, it is likely that a system will have thousands of circuits created.

The **request** part of the message TB640_MSG_SS7_ISUP_OP_CIRCUIT_ALLOC contains the field:

```
...
TB640_SS7_ISUP_HANDLE             hIsup; /* Handle of the ISUP layer */
TB640_SS7_ISUP_CIRCUIT_CFG       Cfg; /* Circuit configuration structure */
TBX_UINT32                        un32NbCircuitAlloc;
...
```

As mentioned before, a message TB640_MSG_SS7_ISUP_OP_CIRCUIT_ALLOC can instantiate more than one circuit at the same time. This is to help the host application to reduce the number of messages to send toward the TB640 when it wants to allocate thousand of circuits. The parameter *un32NbCircuitAlloc* tells the TB640 the number of consecutive circuit to allocate. All configuration parameters for all circuits will be identical with the exception of *un32CircuitId* (which will be incremented by one for each circuit instance) and *un16Cic* (which will also be incremented by one for each circuit instance). See configuration parameter description below for a better understanding of those two parameters.

 During multiple circuit instantiation, only the *un32CircuitId* and *un16Cic* will differ from the configuration from one circuit to the other.

The *cfg* structure contains the configuration parameters for ISUP interface instance:

```
typedef struct _TB640_SS7_ISUP_CIRCUIT_CFG
{
    TBX_UINT32                un32StructVersion;
    TBX_UINT32                un32CircuitId;
    TB640_SS7_ISUP_INTERFACE_HANDLE  hIsupInterface;
}
```

```

TBX_UINT16          un16Cic;
TBX_UINT8           ControlType;
TBX_UINT8           fContChkForOutgoing;
TBX_UINT8           SlotId;
TBX_UINT8           fDualSeizureCtrlMRate;
TBX_UINT8           fNonSs7Conn;
TBX_UINT8           aun8Padding0 [1];
TBX_UINT8           Options;
TBX_UINT16          un16FirstCic;
TBX_UINT16          un16NbCircuitInGroup;
TBX_UINT8           aun8Padding1 [6];
TBX_CHAR            szOutTrkGrpNb [TB640_SS7_ISUP_MAX_TRK_GRP_DIGIT];
TBX_CHAR            szCirIdentName [TB640_SS7_ISUP_MAX_CIRCUIT_IDENTIFICATION_LEN];
TBX_UINT8           aun8Padding2 [1];
TBX_CHAR            szLocationId [TB640_SS7_ISUP_MAX_LOCATION_LEN];
TBX_UINT8           aun8Padding3 [1];
TBX_UINT32          un32T3Timer;
TBX_UINT32          un32T12Timer;
TBX_UINT32          un32T13Timer;
TBX_UINT32          un32T14Timer;
TBX_UINT32          un32T15Timer;
TBX_UINT32          un32T16Timer;
TBX_UINT32          un32T17Timer;
TBX_UINT32          un32ValTimer;
} TB640_SS7_ISUP_CIRCUIT_CFG, *PTB640_SS7_ISUP_CIRCUIT_CFG;

```

The following enumeration lists the different configuration parameters and their description. Unless specifically noted, all parameters are not reconfigurable:

- The circuit ID parameter (*un32CircuitId*) is a 32bits value, assigned by the host application, that is used as an opaque handle by the ISUP layer to refer to a specific circuit part of a specific interface which is in turn part of a specific userpart. The only two requirements are for this value to be unique for the ISUP overall stack and not to be equal to zero.

 *un32CircuitId* cannot be zero

- The ISUP interface parameter (*hIsupInterface*) is the handle provided by the TB640_MSG_SS7_ISUP_OP_INTERFACE_ALLOC message. It is used to specify within which interface this/these circuit instance(s) is/are to be created.
- The CIC parameter (*un16Cic*) is the circuit identification code that will be used for incoming and outgoing calls. Allowable values are protocol signaling standard-specific. For example, ITU-T allows 12bits CIC values and ANSI allows 14bits CIC values. Both standards have spare bits to extend this range to 16 bits but, according to specifications, can do so only with a common agreement/understanding over the SS7 network.
- The control type parameter (*ControlType*) is used to determine which side is controlling call. This is required when handling special cases such as call collisions or dual seizure. This configuration parameter needs to be understood (i.e. properly configured) by both ends of the circuits. Allowed values are:

Table 47 - ISUP circuit control types

Control type	Description
TB640_SS7_ISUP_CALL_CONTROL_INCOMING	Circuit is always controlled by remote end for incoming calls. Only incoming calls are accepted on the circuit. <u>Outgoing</u> call attempts will be refused.
TB640_SS7_ISUP_CALL_CONTROL_OUTGOING	Circuit always controls the remote end for outgoing calls. Only outgoing calls are accepted on the circuit. Incoming call attempts will be refused.
TB640_SS7_ISUP_CALL_CONTROL_BOTHWAY	If ((OPC<DPC) && (CIC is odd)) the circuit controls the call during collisions else if ((OPC >DPC) && (CIC is even)) the circuit is controlled by remote end during collisions. Both incoming and outgoing calls are allowed on the circuit.
TB640_SS7_ISUP_CALL_CONTROL_CONTROLLED	Circuit is always controlled by remote end for all calls during a collision. Both incoming and outgoing calls are allowed on the circuit.
TB640_SS7_ISUP_CALL_CONTROL_CONTROLLING	Circuit is always controlled by the local stack during a collision. Both incoming and outgoing calls are allowed on the circuit.

- The outgoing continuity check parameter (*fContChkForOutgoing*) is used to instruct the ISUP layer to issue a continuity check request for outgoing calls on that circuit.
- The slot identification parameter (*SlotId*) contains the slot ID within a trunk group. This field is read while filling the CAM parameter for sending GRS from the ISUP layer. Moreover, the ISUP layer uses it to validate a CAM parameter with it receives a message with a CAM present. It is only used in ANSI95, ITU97 and ETSI variants. The format of the slot ID is shown below:

Table 48 - ISUP slot ID format

7	6	5	4	3	2	1	0
Spare	MRC supported ?	MRC skipped ?	Timeslot ID				

- Bits 0-4 Represents the 0-31 (E1) or 0-23 (T1) timeslot within the trunk group. Should configure the Slot ID of the first position circuit in a trunk group as 0.
- Bit 5 Tells whether or not the circuit used for contiguous multi-rate calls or is skipped. Can only be used when the checking of starting circuit is done in accordance with Q.763 Table 3 is not required.
- Bit 6 Tells whether or not the circuit supporting contiguous multi-rate calls.

- The dual seizure control for multi-rate parameter (*fDualSeizureCtrlMRate*) is used to indicate the controlling side for CIC for the entire trunk group, where at least one of the calls is a non-single rate connection. This field is used only for ANSI95, ITU97 and ETSI variants.
- The non-SS7 connection parameter (*fNonSs7Conn*) is used to indicate if this circuit is connected to a non-SS7 network. If set, the outgoing trunk group address provided by the parameter *szOutTrkGrpNb* is used. This is also used to generate the EXIT message for outgoing calls. The timer tEx (*un32ExTimer* parameter from the ISUP userpart

configuration) is started when the IAM is sent. Upon timer expiry, an EXIT message is sent. This field is used only for ANSI88, ANSI92, ANSI95 and Telcordia variants.

- The options parameter (*Options*) is used to active per-circuit options. Allowed values are:


Table 49 - ISUP circuit options

Circuit options	Description
TB640_SS7_ISUP_CIRCUIT_OPTIONS_ANSI_INTERNATIONAL	Used to support international calls in addition to national (default). Used for ANSI network only.
TB640_SS7_ISUP_CIRCUIT_OPTIONS_USE_CONFUSION_MSG	Activates the use of the CONFUSION message.
TB640_SS7_ISUP_CIRCUIT_OPTIONS_CIRCUIT_TYPE_UNKNOWN	Indicates circuit group carrier information.
TB640_SS7_ISUP_CIRCUIT_OPTIONS_CIRCUIT_TYPE_ANALOG	
TB640_SS7_ISUP_CIRCUIT_OPTIONS_CIRCUIT_TYPE_DIGITAL	
TB640_SS7_ISUP_CIRCUIT_OPTIONS_CIRCUIT_TYPE_ANALOG_DIGITAL	
TB640_SS7_ISUP_CIRCUIT_OPTIONS_ALARM_CARRIER_UNKNOWN	Indicates the information of alarm carrier.
TB640_SS7_ISUP_CIRCUIT_OPTIONS_ALARM_CARRIER_SOFTWARE	
TB640_SS7_ISUP_CIRCUIT_OPTIONS_ALARM_CARRIER_HARDWARE	
TB640_SS7_ISUP_CIRCUIT_OPTIONS_CONTINUITY_CHECK_UNKNOWN	Indicates the continuity check requirements. These options can be reconfigured.
TB640_SS7_ISUP_CIRCUIT_OPTIONS_CONTINUITY_CHECK_NONE	
TB640_SS7_ISUP_CIRCUIT_OPTIONS_CONTINUITY_CHECK_STATISTICAL	
TB640_SS7_ISUP_CIRCUIT_OPTIONS_CONTINUITY_CHECK_PERCALL	

- The first CIC parameter (*un16FirstCic*) indicates the first circuit identification code in the circuit group. This field is used only for ANSI88, ANSI92, ANSI95 and Telcordia. When using one of those variants and receiving (or sending) a group command with a range of 0 (within the “range and status” IE), the ISUP layer uses *un16FirstCic* and *un16NbCircuitInGroup* to know which CICs are affected by the command. Refer to section 6.3.3.1 for more information.
- The number of circuit in group parameter (*un16NbCircuitInGroup*) indicates total number of CIC part of the circuit group. This field is used only for ANSI88, ANSI92, ANSI95 and Telcordia. When using one of those variants and receiving (or sending) a group command with a range of 0 (within the “range and status” IE), the ISUP layer uses *un16FirstCic* and *un16NbCircuitInGroup* to know which CICs are affected by the command. Refer to section 6.3.3.1 for more information.
- The outgoing trunk group nb parameter (*szOutTrkGrpNb*) is used to store the outgoing trunk group number to be used in the EXIT messages for ANSI88, ANSI92, ANSI95 and Telcordia variants for non-SS7 circuits. For SS7 circuits, this field is used to store the circuit ID name for the Telcordia variant.
- The circuit identification name parameter (*szCirIdentName*) is used to store the circuit identification name expected in the CVR message. This field is only used for ANSI88, ANSI92, ANSI95 and Telcordia to validate the circuit identification name upon validation of incoming CRV messages. Refer to the ANSI specification for the format of this field.
- The location ID parameter (*szLocationId*) is contains the common language name to identify the switching office by town, state and building subdivision. This field is only used for ANSI88, ANSI92 and Telcordia variants.

- The timer configuration parameters. All timers are reconfigurable. All timer values must be expressed in milliseconds. The timers have the following definitions:

un32T3Timer:	OVERLOAD message received. Terminated normally after 2 minutes.
un32T12Timer:	BLOCKING message sent. Terminated normally when BLOCKING ACKNOWLEDGEMENT is received. Typical values are 15 to 60 seconds.
un32T13Timer:	INITIAL BLOCKING message sent. Terminated normally when BLOCKING ACKNOWLEDGMENT is received. Typical values are 5 to 15 minutes.
un32T14Timer:	UNBLOCKING message sent. Terminated normally when UNBLOCKING ACKNOWLEDGMENT is received. Typical values are 15 to 60 seconds.
un32T15Timer:	INITIAL UNBLOCKING message sent. Terminated normally when UNBLOCKING ACKNOWLEDGMENT is received. Typical values are 5 to 15 minutes.
un32T16Timer:	RESET message sent. Terminated normally when RELEASE COMPLETE is received. Typical values are 15 to 60 seconds.
un32T17Timer:	INITIAL RESET message sent. Terminated normally when RELEASE COMPLETE is received. Typical values are 5 to 15 minutes.
un32ValTimer:	Circuit validation timer for ANSI88, ANSI92, ANSI95 and Telcordia. Typical value is 10 seconds.

 UK: The following timers are not required for the UK: T3.

The **response** part of the message TB640_MSG_SS7_ISUP_OP_CIRCUIT_ALLOC contains the field:

```

...
TBX_UINT32                un32NbCircuitAllocated;
TBX_UINT32                un32NbCircuitNotAllocated;
...

```

The following enumeration lists the different response parameters and their description:

- The number of allocated circuit response parameter (*un32NbCircuitAllocated*) contains the number of circuits that were successfully allocated. If everything went well, this number should be equal to *un32NbCircuitAlloc* from the request part of the message (see above).
- The number of non-allocated circuit response parameter (*un32NbCircuitNotAllocated*) contains the number of unsuccessful allocation attempt. If this number is non-zero, the host application must consider that *un32NbCircuitAllocated* were allocated. Since circuits are allocated sequentially in case of multiple allocations, the host application can assume that the first *un32NbCircuitAllocated* were allocated.

$$\text{✍ } un32NbCircuitAllocated + un32NbCircuitNotAllocated = un32NbCircuitAlloc$$

✍ As specified in section 6.1.3, an ISUP circuit can only be part of a single ISUP interface. On the other hand, multiple circuits can be attached to the same interface instance. Those circuits represent the voice channels available between the local SS7 node and the remote SS7 node.

6.2.3 Compatibility

6.2.3.1 Variants

See section 4.2.2.1 Variants.

6.3 ISUP Signaling

6.3.1 Host application requirements

The TB640 signaling architecture is built for high performance throughput in terms of “calls per seconds”. To achieve these goals, the SS7 stack located on the TB640 processes the protocols layers MTP1-3/ISUP and provides alarm and event notifications to the host application for multiple calls in parallel. For ISUP, each call is attached to a specific circuit ID handle (which is in turn related to a specific network/OPC/DPC/variant/CIC). This ID (provided by the application upon circuit allocation) is included in every call-related event to allow the user application to know to which call it refers to. Thus, the application will receive thousands of events, each referring to different calls. Since a complete ISUP calls is achieve by processing multiple SS7 messages (IAM, ACM, ANM, REL, RLC, etc), multiple messages are required for a call to reach the active state and then to be released. These message flows, explained in section 6.3.4, need to be controlled by the host application. Only the application possesses the information to accept/route calls according to call information (numbers, characteristics, billing, etc). Therefore, the application plays an important role in the performance of the stack. If this application is not optimized or poorly designed (e.g. synchronous), decent performance will not be achieved.

6.3.1.1 General guidelines for designing an efficient call control application using ISUP

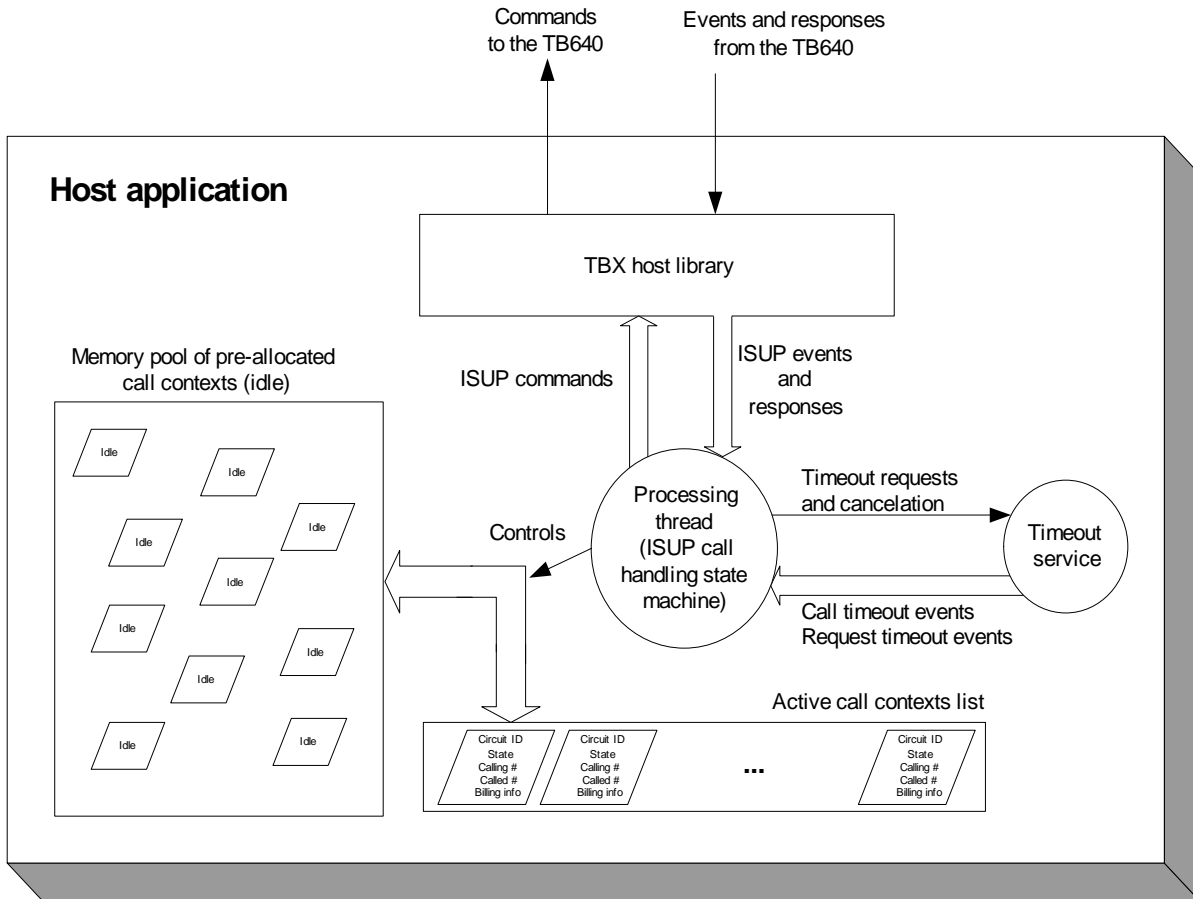


Figure 13 - Host application typical architecture

As mentioned before, a call control application must be designed as a state machine that can be applied to every individual call. Although multi-threading is preferable, it is not required as long as there is no synchronous operation during the processing of a call event. The easiest way to achieve this is to create dynamic structure (call context) containing a call’s useful information and state. Upon the reception of an event associated to a call (by the “circuit ID” handle), the application retrieves the call context, processes the new event, sends appropriate requests toward the TB640 blade, updates the information/state of the call context and waits for another event from the board. Usually, a call control state machine also has to deal with timers (i.e. timeout events) to cover for call failure cases (e.g. end-user not responding, retries, etc). The main state-machine thread can offload this task to a timeout service (e.g. another thread) responsible to generate timeout events to the main thread in case a specific timer, associated with a specific call, expires. Thus, timeout events are processed as any other events coming from the TB640. The state machine should be implemented to react as specified by the call flows shown in section 6.3.4. A more detailed description of the SS7 ISUP state machine implementation can be found along in the SS7 sample directory of the TB640 package.

1. One thread should be responsible to process all ISUP and timeout events for every calls associated with the controlled TB640.

2. This thread should configure a TBX host library filter to receive only the ISUP events of the targeted TB640.
3. Once an event is received, the process should retrieve the call context associated with the circuit ID (or create it in the case of a new call) and process the events
4. If, during the processing of the event, a synchronous operation is required to continue (i.e. reading information from a remote database), post the request to the database, make the call to enter a 'waiting for DB' state and return the call context into the waiting queue. In this case, database events should be able to wakeup the thread as well as TB640 events or timeouts. This way, the main thread will be processing other calls while waiting for information from the database. When the answer from the database will arrive, the thread will re-fetch the call context and continue the processing where it had left off. The developer must remember that an SS7 call can live from a couple of seconds to hours depending on numerous factors. Thus, the latency of SS7 events related the same calls is not as important as the total capacity of event processing of the thread to achieve high performance system.

Another important aspect of an efficient system is about the flow control. The host application must have the capacity to slow down its rate of calls when the ISUP layer informs it of a potential congestion of the network as described in section 6.3.3.3. This is usually done by doing back-pressure on the input of the system. Some applications totally control the input into the system because they are the call generators. But usually, the input comes from an external source (such as another system calling into this one). In this last case, the rate of input calls can be reduced by refusing calls with the proper cause value (i.e. "call rejected", "destination out-of-order", "No circuit/channel available", "network out-of-order", "switching equipment congestion", etc) as defined in specification Q.850.

6.3.1.2 General guidelines for designing highly redundant application (blade failure)

1. SS7 protocol protects the overall network from link (trunk) failure by automatically re-routing traffic to other links. This is done automatically by the MTP3 layer of the stack providing the proper physical setup (multiple links and optionally links toward multiple STPs)
2. In case of TB640 blade failure (software, hardware or chassis), the TB640 SS7 architecture provides redundant stack on other blade providing the proper setup and configuration. This way, the call control application only has to access the stand-by board and continue processing as if nothing had happened. **[This option is not available to end-user yet].**

6.3.1.3 General guidelines for designing a high-performance system

High traffic (high number of circuits) and performance system (low call processing response time) usually requires multiple TB640 blades and multiple host machines. These systems are almost

always required to be scalable (down or up) to be adaptable to multiple different customer application (i.e. a switch platform). Thus, those huge systems must have a very clean (read 'simple') architecture to be easily maintainable and accommodate new features sets.

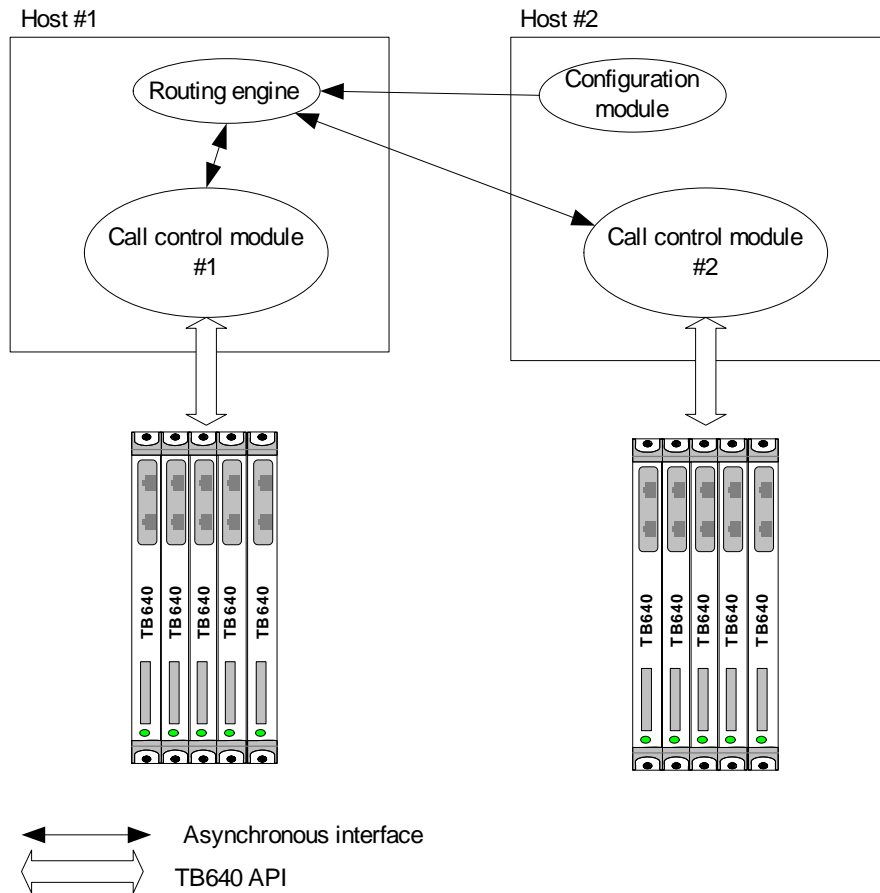


Figure 14 - Multi-host high-performance system

1. It is easier to architect such a system when every individual host machine can control a set of resources independently of other host machine. For example (refer to Figure 14), in a system where ten TB640 would be required (i.e. more than 300,000 circuits with a capacity of more than 8000 calls per second), a machine could process the call control of five TB640 and their associated circuits. This approach usually also makes the system easily scalable by adding more 'node' of TB640/host machines without having to change the architecture.
2. Other functions of the system (e.g. routing, configuration, etc) should also be divided in functional modules and allow interaction with other module through asynchronous APIs (i.e. message based APIs between modules). This will force all modules to deal with asynchronous events and thus parallelize work (as mentioned before).
3. Once every interface is made asynchronous, functional modules may be dispersed amongst different machines to increase the overall throughput of the system by using the CPU (and memory) resources of multiple machines.

4. All previously mentioned guidelines must be respected (i.e. no synchronous operations) in every modules to avoid creating a bottleneck because of a slow module in the data flow. In case of extensive operation required by a single module, think either to re-divide (parallelize) the functions of that module or to isolate the module onto a very powerful machine dedicated to that functional task.

6.3.1.4 General guidelines to have highly redundant high-performance system (host failure)

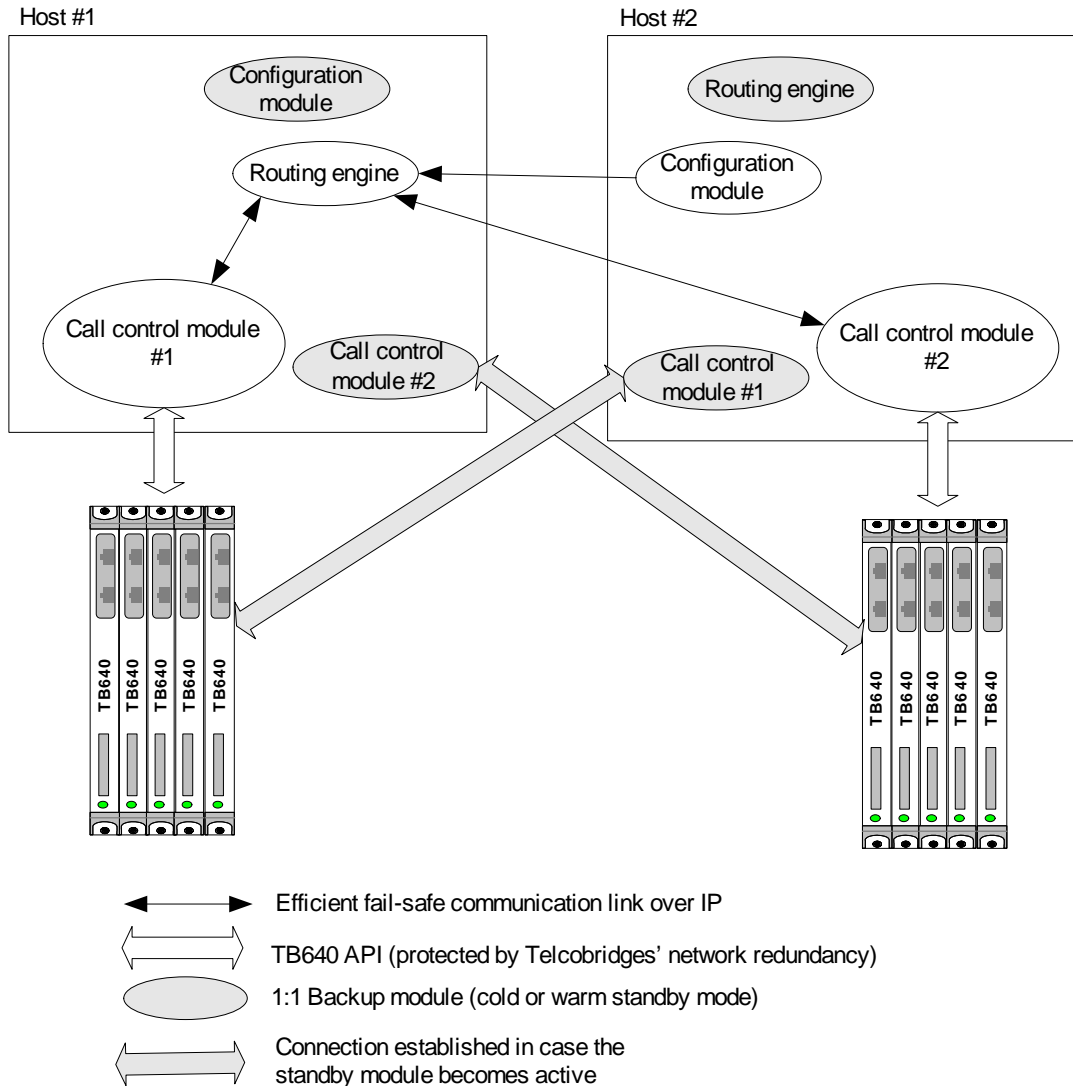


Figure 15 - HA introduced in large system

1. Extending the previous guidelines, every module should have a backup module on another machine (refer to Figure 15) in case an unexpected failure occurs (i.e. power failure on a host machine). Standby module should monitor the active module and decide to take over in case of failure.

2. Depending on the system requirements, the architect may want to use host machine capable to handle twice the normal load if no performance hit is required in case of failures.
3. Inter-module communication through IP must be made fail-safe as well to avoid dealing with link failure or message re-ordering within the functional modules.
4. Call control module redundancy is mostly implemented by the TB640 API architecture since it is always possible to recover the configuration and/or state of all resources and calls presents on a TB640 blade at any moment. Thus, the standby call control application only has to query the different TB640 blades and reconstruct its internal data structure based on this information.
5. Communication links between hosts and the TB640 blades are protected by the Telcobridges' network redundancy (i.e. link disconnection, switch failures, bad cables, etc.) and thus only require the proper system configuration (i.e. having two Ethernet switches and two Ethernet line card on every host machine).

6.3.2 Information element usage

6.3.2.1 IE Usage

Within every notifications or request messages (other than configuration messages), the host application will have to read or insert data structure called "information elements" (or IEs). These data structure contains mandatory or optional information required by the ISUP layer and/or the remote ISUP layer to properly process the required operation (such as issuing a voice call). The SS7 ISUP specifications defines approximately one hundreds different IEs that are used differently according to variant or type of message. As a common example, the "called party number" IE contains the phone number of the party the host application wants to connect to. Therefore, this IE is inserted in the IAM message to inform the remote ISUP layer about the targeted end-user. This section will cover the usage of IEs with the TB640 ISUP API and will provide basic guidelines on how to use them. Even if this document gives the basics instruction for information elements, the end-user should always refer to the proper specifications to get a more detailed description and usage information. Some local or national variants may also be in use in different SS7 networks all over the world and, thus, the customer should always refer to the proper authority when interoperability is wanted. In summary, the ISUP layer does part of the job when controlling calls over the SS7 network and the host application also needs to feed the proper information to the ISUP layer for the complete solution to work properly.



ITU-T information elements are described more in details in specification document Q.763.



IE encoding, although similar between specifications (i.e. ANSI and ITU), may still differ slightly and requires the user to get access to the specification documents he plans to use. Since those are copyright-protected documents, Telcobridges cannot make those accessible to customer directly.

6.3.2.2 IE inclusion in different messages

Depending on which messages is sent or receive to/from the SS7 network, the host application should expect specific IEs to be inserted or processed. Table 50 and Table 51 shows in which messages IEs can be used or are to be expected. Some IEs are mandatory which means that a request toward the stack will fail if they are not present. On the other hand, it does mean that optional IEs are not required but it might depends on the variant that is being used or on the local network implementation. This is why the host application designer needs to verify with national or international authority for the proper IEs.

Table 50 - ISUP mandatory/optional IEs (ITU)

	IAM	SAM	IN	IN	COT	ACM	CO	CP	ANM	FOT	RE	RL	CC RS LPA	BLO UBL OL	BLA UBA UCI	SU RE	CF	CG CG	CGB CGU	CO GR GR	CQ	FAA FAR	FR	FAC	ID	IR	NR	SG	PAM	US	UPA UP			
Access delivery information						O	O	O	O		O																							
Access transport	O					O	O	O	O		O																O		O			O		
Automatic congestion level											O																							
Backward call indicators						M	M	O	O																									
Call diversion information						O		O																										
Call history information							O		O																									
Call reference	O		O	O		O	O	O	O	O							O						O											
Called party number	M																																	
Calling party number	O			O																							O							
Calling party's category	M			O																														
Cause indicators						O		O			M	O						M						M										
Circuit group supervision message type ind.																		M	M															
Circuit state indicator																							M											
CUG interlock code	O																																	
Connected number							O		O																									
Connection request	O			O																				O										
Continuity indicators					M																													
Echo control information						O	O		O																					O				
Event information	O	O	O	O		O	O	O	O	O	O	O					O	O					O	O	O	O	O	O	O	O	O	O	O	
Facility indicator									M																									
Forward call indicator	M																																	
Generic digit	O																																O	
Generic notification ind.	O					O	O	O	O																								O	
Generic number	O						O		O																		O		O					
Generic reference	O																																	
Information indicators					M																													
Information request indicators			M																															
Location number	O																																	
Message compatibility information																									O	O	O	O	O					
MCID response indicator																												O						
MCID request indicators																										O								
MLPP precedence	O																																	
Nature of connection indicators	M																																	
Network specific facility	O		O	O		O	O	O	O	O	O																							
Optional backward call indicators						O	O	O	O																									
Optional forward call indicators	O																																	
Original called number	O																																	
Originating ISC point code	O																																	
Parameter compatibility information	O		O	O		O	O	O	O	O	O													O		O	O	O	O				O	
Propagation delay counter	O																																	
Range and status																		M	M	M	M													
Redirection information	O										O																							
Redirecting number	O																																	
Redirection number						O	O	O	O	O	O																							
Redirection number restricted						O	O	O	O	O	O																							
Remote operations	O					O	O	O	O																	O								
Service activation	O					O	O	O	O																	O								
Signaling point code											O																							
Subsequent number		M																																
Suspend/resume indicators																	M																	
Transit network selection	O																																	
Transmission medium requirement	M																																	
Transmission medium requirement prime	O																																	
Transmission medium used						O	O	O	O																									
User service information	O																																	
User service information prime	O																																	
User-to-User indicators	O					O	O	O	O	O	O													O	O									
User-to-user information	O					O	O	O	O	O	O																				O		M	

Table 51 - ISUP mandatory/optional IEs (ANSI)

	IAM	INR	INF	CRA	CRM	COT	ACM	EXM	ANM	CPG	FOT	REL	CFN	CVR	CVT	RSC	BLO	BLA	SUS	CGB	CGBA	GRS	CQR	FAC
Access transport	O		O				O	O	O		O													
Automatic congestion level												O												
Backward call indicators							M	O	O															
Business group	O		O				O	O	O															
Call reference	O	O	O				O	O	O	O	O								O					
Called party number	M																							
Changed number	O		O									O												
Calling party number	O		O																					
Calling party's category	M		O																					
Carrier identification	O																							
Carrier selection information	O																							
Cause indicators							O		O		M													
Circuit group assignment map	O																					O		
Circuit group characteristic indicators														M										
Circuit group supervision message type ind.																				M	M			
Circuit identification name														O										
Circuit state indicator																							M	
Circuit validation response ind.														M										
CLLI code														O										
Connection request	O	O	O				O	O																
Continuity indicators						M																		
Egress service	O																							
Event information										M														
Forward call indicator	M																							
Generic address	O											O												
Generic digits	O																							
Generic name	O																							
Hop counter	O																							
Information indicators			M				O	O	O															
Information request indicators	O		M																					
Justification indicators	O																							
Nature of connection indicators	M				M																			
Network transport	O	O					O	O	O															
Notification indicator							O		O															
Original called number	O																							
Operator services information	O																							
Optional backward call indicators							O	O	O															
Originating line information	O		O																					
Outgoing trunk group number							O																	
Precedence	O																							
Range and status																				M	M	M	M	
Redirection information	O		O			O																		
Redirecting number	O									O														
Remote operations	O						O	O	O															O
Service activation	O						O	O	O		O													O
Service code	O																							
Special processing request	O																							
Suspend/resume indicators																				M				
Transaction request	O																							
Transit network selection	O																							
Transmission medium used							O	O	O															
User service information																								
User service information prime	O																							
User-to-User indicators							O	O	O															
User-to-user information	O		O				O	O	O		O													

6.3.2.3 Reading and writing IEs

Information elements are data structures that carry data onto the SS7 network in a pre-defined format. Since most IEs and even internal fields of IEs are optional, the TB640 ISUP API uses a mechanism of opaque byte-buffer to transfer data between the blade and the host application.

Every ISUP API message (except configuration message) contains fields named *un16InfoElemSize* and *aun8InfoElem* (see below). The field *aun8InfoElem* is a buffer of byte that contains the actual IE data and *un16InfoElemSize* is the number of bytes contained in the buffer.

```
typedef struct _TB640_REQ_SS7_ISUP_OP_CONN_REQUEST
{
    TBX_MSG_HEADER                Header;
    TBX_UINT32                    un32MsgVersion;
    TB640_SS7_ISUP_HANDLE         hIsup;
    TB640_SS7_ISUP_CONNECTION_CFG Cfg;

    TBX_UINT16                    un16InfoElemSize;
    TBX_UINT8                     aun8Padding [2];
    TBX_UINT8                     aun8InfoElem [4];
} TB640_REQ_SS7_ISUP_OP_CONN_REQUEST, *PTB640_REQ_SS7_ISUP_OP_CONN_REQUEST;
```

Since the byte-buffer can be considerably larger than what is actually declared in the typedef (actually 4 bytes), it is important that the host application allocates message buffer size according to the real data it is going to insert in it as shown in the sample code below:

```
PTB640_MSG_SS7_ISUP_OP_CONN_REQUEST    pMsg = NULL;
TBX_MSG_HANDLE                          hMsg = NULL;
TBX_UINT32                              un32MsgMaxSize = 0;

/* Calculate max message size with max size of IEs used */
un32MsgMaxSize = sizeof(*pMsg) +
    (sizeof(TB640_SS7_ISUP_IE_CALLED_PARTY_NUMBER) +
    sizeof(TB640_SS7_ISUP_IE_CALLING_PARTY_NUMBER) +
    sizeof(TB640_SS7_ISUP_IE_CALLING_PARTY_CATEGORY) +
    sizeof(TB640_SS7_ISUP_IE_FORWARD_CALL_INDICATORS) +
    sizeof(TB640_SS7_ISUP_IE_NATURE_OF_CONNECTION_INDICATORS));

/* Add padding if required to align the message on 64 bits boundary */
TBX_MSG_ADD_PADDING_TO_ALIGN_64 (un32MsgMaxSize);

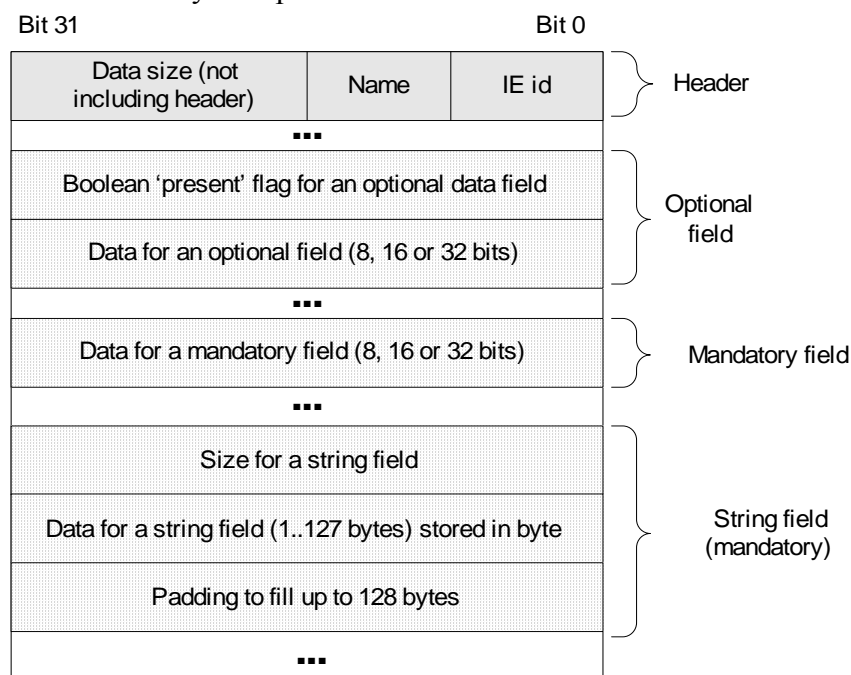
/* Get message buffer */
TBXGetMsg (hLib, un32MsgMaxSize, &hMsg);

/* Fill the generic header */
TBX_FORMAT_MSG_HEADER (
    hMsg,
    TB640_MSG_ID_SS7_ISUP_OP_CONN_REQUEST,
    TBX_MSG_TYPE_REQUEST,
    un32MsgMaxSize,
    hAdapter,
    0,
    0);

/* Convert into proper message structure */
pMsg = (PTB640_MSG_SS7_ISUP_OP_CONN_REQUEST)TBX_MSG_PAYLOAD_POINTER (hMsg);

/* Fill the content of the message */
pMsg->Request.un16InfoElemSize = 0;
pMsg->Request.hIsup = hIsup;
...
```

The actual format of this encoding is shown in Figure 16 as a reference only (the host application does not need to know this format). This ensures an optimal encoding of IE into TB640 messaging system and helps with overall system performance.



Note: Data is stored in big-endian format

Figure 16 - IE encoding

However, it wouldn't be very user-friendly to let the user application to encode itself the IE into the TB640 message one byte at the time. Therefore, the TB640 ISUP API includes a set of macros to access the IEs (read, write and search) into a message/notification buffer. Every IE has its own read and write macros with the proper parameters. The application designer will find the macros for every supported IE in the API file **tb640_ss7_isupie.h**. Below is a sample of code writing IEs into an allocated message buffer. Note that the *un16InfoElemSize* is updated automatically by the macros and, thus, only needs to be initialized to 0 before using the macros.

```

/* Fill the message */
pMsg = (PTB640_MSG_SS7_ISUP_OP_CONN_REQUEST)TBX_MSG_PAYLOAD_POINTER(hMsg);
pReq = &(pMsg->Request);
pReq->un32MsgVersion = 1;
pReq->hIsup = hIsup;
pReq->Cfg.un32StructVersion = 1;
pReq->Cfg.un32CircuitId = un32CircuitId;
pReq->Cfg.ExchangeType = TB640_SS7_ISUP_CONNECTION_EXCHANGE_TYPE_A;

/* Fill the IEs */
pReq->un16InfoElemSize = 0;

/* Insert called party number IE */
TB640_SS7_ISUP_WRITE_CALLED_PARTY_NUMBER (
    pReq->aun8InfoElem,
    &(pReq->un16InfoElemSize),
    0,
    un8NatOfAddressInd,
    (TBX_UINT8)fCalledAddressSizeOdd, /* Odd or even -> 0 = even (refer

```



```

    * to Q.763) */
    1, /* Numbering plan -> 1 = ISDN
    * numbering plan (refer to Q.763)
    */
    0, /* Internal network number
    * indicator -> 0 = routing to
    * internal number allowed (refer
    * to Q.763) */

    un8CalledAddressSize,
    aun8CalledAddress);

/* Insert calling party number IE (optional) */
TB640_SS7_ISUP_WRITE_CALLING_PARTY_NUMBER (
    pReq->aun8InfoElem,
    &(pReq->un16InfoElemSize),
    0,
    un8NatOfAddressInd,
    (TBX_UINT8)fCallingAddressSizeOdd, /* Odd or even -> 0 = even (refer
    * to Q.763) */
    1, /* Screening indicator -> 1 = user
    * provided, verified and passed
    * (refer to Q.763) */
    0, /* Address presentation restricted
    * indicator -> 0 = Allowed (refer
    * to Q.763) */
    1, /* Numbering plan -> 1 = ISDN
    * numbering plan (refer to Q.763)
    */
    0, /* Internal network number
    * indicator -> 0 = routing to
    * internal number allowed (refer
    * to Q.763) */

    in_pCallContext->un8CallingAddressSize,
    in_pCallContext->aun8CallingAddress);

/* Insert calling party category IE */
TB640_SS7_ISUP_WRITE_CALLING_PARTY_CATEGORY (
    pReq->aun8InfoElem,
    &(pReq->un16InfoElemSize),
    0,
    10); /* Calling party's category -> 10 =
    * ordinary calling subscriber
    * (refer to Q.763) */

```

When all IE elements are inserted inside buffer, you can adjust the message size to the real used space by using these steps.

```

/* Verify and add padding of the actual message size */
un32MsgMaxSize = sizeof(*pMsg) + pReq->un16InfoElemSize;
TBX_MSG_ADD_PADDING_TO_ALIGN_64 (un32MsgMaxSize);

/* Shrink message with the new message size */
TBX_MSG_PAYLOAD_LENGTH_SET (hMsg, un32MsgMaxSize);

/* Send message buffer */
TBXSendMsg (hLib, hMsg, NULL);

```

The reading of an IE buffer, received in a notification, is quite similar. Using a set of macros, the host application can parse through the buffer to search for a specific IE within the buffer. Once the IE is found (if it is), the application can read its content into local variable. Below is a sample of

code reading IEs from a received notification. Note that, in the sample code, the local variable *unl6Offset* is used by the macros to keep track of where the search pointer is within the IE buffer.

```


pEvt = (PTB640_EVT_SS7_ISUP_NOTIF_CONN_INDICATION)
        TBX_MSG_PAYLOAD_POINTER(in_hMsg);

/* Search for called number IE */
unl6Offset = 0;
unl6SearchResult = TB640_SS7_ISUP_SEARCH_IE (
    pEvt->aun8InfoElem,
    &unl6Offset,
    pEvt->unl6InfoElemSize,
    TB640_SS7_ISUP_IE_ID_CALLED_PARTY_NUMBER,
    0);
if (unl6SearchResult == TB640_SS7_ISUP_IE_NOT_FOUND )
{
    TBX_EXIT_ERROR(TBX_RESULT_FAIL, 0, "Didn't found mandatory IE");
}
TB640_SS7_ISUP_READ_CALLED_PARTY_NUMBER (
    pEvt->aun8InfoElem,
    &unl6Offset,
    &un8Unused, &un8Unused,
    &fCalledAddressSizeOdd,
    &un8Unused, &un8Unused,
    &un8CalledAddressSize,
    aun8CalledAddress);


/* Search for calling number */
unl6Offset = 0;
unl6SearchResult = TB640_SS7_ISUP_SEARCH_IE (
    pEvt->aun8InfoElem,
    &unl6Offset,
    pEvt->unl6InfoElemSize,
    TB640_SS7_ISUP_IE_ID_CALLING_PARTY_NUMBER,
    0);
if (unl6SearchResult == TB640_SS7_ISUP_IE_NOT_FOUND)
{
    TBX_EXIT_ERROR(TBX_RESULT_FAIL, 0, "Didn't found mandatory IE");
}
TB640_SS7_ISUP_READ_CALLING_PARTY_NUMBER (
    pEvt->aun8InfoElem,
    &unl6Offset,
    &un8Unused, &un8Unused,
    &fCallingAddressSizeOdd,
    &un8Unused, &un8Unused, &un8Unused, &un8Unused,
    &un8CallingAddressSize,
    aun8CallingAddress);

```

You may have noticed that all IE ‘write’ macros their third parameter always set to 0. This third parameter is called the ‘IE name’ and mustn’t be confused with the IE type. The IE type is an identifier representing which specific IE type is desired. Examples of those are TB640_SS7_ISUP_IE_ID_CALLED_PARTY_NUMBER, TB640_SS7_ISUP_IE_ID_CALLING_PARTY_CATEGORY, TB640_SS7_ISUP_IE_ID_GENERIC_DIGITS and are corresponding to the IE enumeration found in specifications (i.e. Q.763 for ITU).

 IE name ≠ IE type.

The IE name is an integer digits used to differentiate between two IEs that have the same IE type (i.e. multiple instances of a same IE type). For example, in a connection notification, you may receive two IE of the type TB640_SS7_ISUP_IE_ID_CALLED_PARTY_NUMBER. One (with name 0) will represent the ‘called party number’ and the other (with name 1) will contain the ‘redirection number). Thus, the same IE type may represent different information through a different IE name. To know which IE name/type to use, please refer to the SS7 ISUP API reference guide in the section “*IE usage information according to SS7 message types*” to get a detailed description.

 A single IE type can be found multiple times in a single notification or request message. Those multiple instances are differentiated with the IE name value.

6.3.2.4 Most common IEs ITU format definitions

This section contains the most commonly used IE from the ITU-T specifications. These IEs are the ones used in the ss7 sample application contained in the TB640 official packages. Those should be sufficient to establish and release basic ISUP calls. To use more advanced features of the SS7 networks, the application designer will need to get access to the proper ITU-T, ANSI and/or Telcordia specifications. As mentioned before, the user application does not need to format the actual IE since it has to use the appropriate macros. But the application designer must understand the signification of every fields of an IE to produce the proper behavior onto the SS7 network.

6.3.2.4.1 Called party number IE

Refer to "Called party number" information element as described Q.763 section 3.9.

Table 52 - Called party number IE format

	8	7	6	5	4	3	2	1
1	Odd/ even	Nature of address indicator						
2	INN Ind.	Numbering plan Ind.			Spare			
3	2nd address signal				1st address signal			
.								
.								
n	Filler (if necessary)				nth address signal			

The “nature of address indicator” contains the type of number being inserted in the address signals (i.e. 1 means a subscriber number in a national context). The “odd/even” bit specifies the number of address signals is odd (1) or even (0). The “numbering plan indicator” specifies the format of the signals (i.e. 1 means ISDN numbering as per recommendation E.164). The “INN ind.” indicates if routing into the internal network is allowed (0) or not allowed (1). The signals are BDC data corresponding to the number the party wants to reach (i.e. 0 means digit 0, 1 means digit 1, and so on).

6.3.2.4.2 Calling party number IE

Refer to "Calling party number" information element as described Q.763 section 3.10.

Table 53 - Calling party number IE format

	8	7	6	5	4	3	2	1
1	Odd/ even	Nature of address indicator						
2	NI	Numbering plan Ind.			Address presentation restricted indicator		Screening	
3	2nd address signal				1st address signal			
.								
.								
n	Filler (if necessary)				nth address signal			

The “nature of address indicator”, “odd/even”, “numbering plan ind.” and signals have the same usage as in section 6.3.2.4.1. The “screening” field is used to identify the source of the screening (i.e. 1 means that the user has already validated the information). The “address presentation restricted indicator” indicates if the number is private (1) or public (0). The “NI” field indicates whether or not the number is complete (0) or incomplete (1).

6.3.2.4.3 Calling party’s category IE

Refer to "Calling party’s category" information element as described Q.763 section 3.11.

Table 54 - Calling party's category IE format

8	7	6	5	4	3	2	1
Calling party's category							

The “calling party’s category” is used to identify the type of subscriber making the call (i.e. 10 means an ordinary calling subscriber).

6.3.2.4.4 Forward call indicator IE

Refer to "Forward call indicator" information element as described Q.763 section 3.23.

Table 55 - Forward call indicator IE format

	8	7	6	5	4	3	2	1
1	H	G	F	E	D	C	B	A
2	P	O	N	M	L	K	J	I

The field A indicates if the call is a national (0) or international (1) call. The fields BC indicates if an end-to-end communication method is available (i.e. 1 means the pass-along method is available). The field D indicates if internetworking with other protocols (other than SS7) has been encountered (1) or not (0). The field E indicates if end-to-end information is available (1) or not (0). The field F indicates if the ISUP is used throughout the path (1) or not (0). The fields GH indicates the preferred communication way throughout the path (i.e. 0 means ISUP is preferred all the way). The field I indicates if the originating end is ISDN (1) or non-ISDN (0). Other fields should be 0.

6.3.2.4.5 Nature of connection indicators IE

Refer to "Nature of connection indicators" information element as described Q.763 section 3.35.

Table 56 - Nature of connection indicators IE format

8	7	6	5	4	3	2	1
H	G	F	E	D	C	B	A

The fields AB indicates if satellite(s) are included in the path (i.e. 0 means no satellite connection in the circuit connection). The field CD indicates if continuity check is required (1) or not (0). The field E indicates the presence (1) or not (0) of an outgoing echo canceller device on the circuit.

6.3.2.4.6 Backward call indicators IE

Refer to "Backward call indicators" information element as described Q.763 section 3.5.

Table 57 - Backward call indicators IE format

	8	7	6	5	4	3	2	1
1	H	G	F	E	D	C	B	A
2	P	O	N	M	L	K	J	I

The fields AB are the charge indicator (i.e. 2 means “charge”). The fields CD are the called party’s status indicator (i.e. 1 means “subscriber free”). The fields EF are the called party’s category indicators (i.e. 1 means “ordinary subscriber”). The fields GH indicate the type of end-to-end mechanism is available (i.e. 1 means the pass-along method is available). The field I indicates if internetworking with other protocols (other than SS7) has been encountered (1) or not (0). The field J indicates if end-to-end information is available (1) or not (0). The field K indicates if the ISUP is used throughout the path (1) or not (0). The field L indicates if a hold operation has been requested (1) or not (0). The field M indicates if the call is terminating access for ISDN (1) or non-ISDN (0). The field N indicates the presence (1) or not (0) of an incoming echo canceller.

6.3.2.4.7 Subsequent number IE

Refer to "Subsequent number" information element as described Q.763 section 3.51.

Table 58 - Subsequent number IE format

	8	7	6	5	4	3	2	1
1	Odd/ even	Spare						
2	2nd address signal				1st address signal			
.								
.								
n	Filler (if necessary)				nth address signal			

The field formats are identical to the ones in section 6.3.2.4.1

6.3.2.4.8 Cause indicators IE

Refer to "Subsequent number" information element as described Q.763 section 3.12.

Table 59 - Cause indicators IE format

	8	7	6	5	4	3	2	1
1	ext.	Coding standard		Spare	Location			
2	ext.	Cause value						
3	Diagnostic(s) (if any)							
.								
.								
n								

NOTE – Octet 3 to 3n may be omitted or repeated, e.g. 3' to 3'n.

The coding and subfields are identical as those of the ISDN signaling protocol. Refer to the TB640 User's guide and to Q.850 for more information.

6.3.2.4.9 Suspend/resume indicators IE

Refer to "Suspend/resume indicators" information element as described Q.763 section 3.52.

Table 60 - Suspend/resume indicators IE format

8	7	6	5	4	3	2	1
H	G	F	E	D	C	B	A

The field A indicates if the request is user-initiated (0) or network-initiated (1).

6.3.2.4.10 Range and status IE

Refer to "Range and status" information element as described Q.763 section 3.43.

Table 61 - Range and status IE format

	8	7	6	5	4	3	2	1
1	Range							
2	Status							
.								
.								
n								

The range value specifies the total number of circuits affected by the command. Actually, (range+1) is the number of affected circuits. The range value of zero is invalid for RESET, BLOCK and UNBLOCK commands according to ITU specifications (but is valid for ANSI). Receiving a zero indicates to the ISUP layer to affect the whole circuit group according to its configuration parameters (i.e. using *un16FirstCic* and *un16NbCircuitInGroup* from the circuit configuration parameters described in section 6.2.2.5). The maximum number of affected circuits must be 32 or less (which makes range value to be up to 31 only). The status field is a bitfield that contains the indication of which specific circuits are affected by the command. Although the bitfield covers a maximum of 256 circuits, a maximum of 32 bits can be set to 1 at the same time (max of 32 circuits affected). When doing calculation, the starting circuit is always the circuit on which the MSU was sent or received. Note that the status field is optional and, thus, is not present

if the range value is zero and for group RESET indication message (group RESET acknowledge has the status bit field).

6.3.3 ISUP functionalities and behaviors

6.3.3.1 Blocking and unblocking

ISUP gives the opportunity to the local application to decide whether or not a specific circuit can receive or transmit normal call. This is achieved through the blocking/unblocking state of the circuit. An ISUP layer always remembers the state of the local circuit as well as the state of the remote ISUP peer regarding the same circuit. It is important to remember that a local stack cannot change the state of the remote stack regarding the block state of the circuit. This behavior prevents an SS7 node to re-activate a circuit that is flagged as out-of-service by the remote end which would be hazardous because it does not know the reason why it was blocked. Note that the 'blocked' state does not prevent test calls to be made on a circuit.

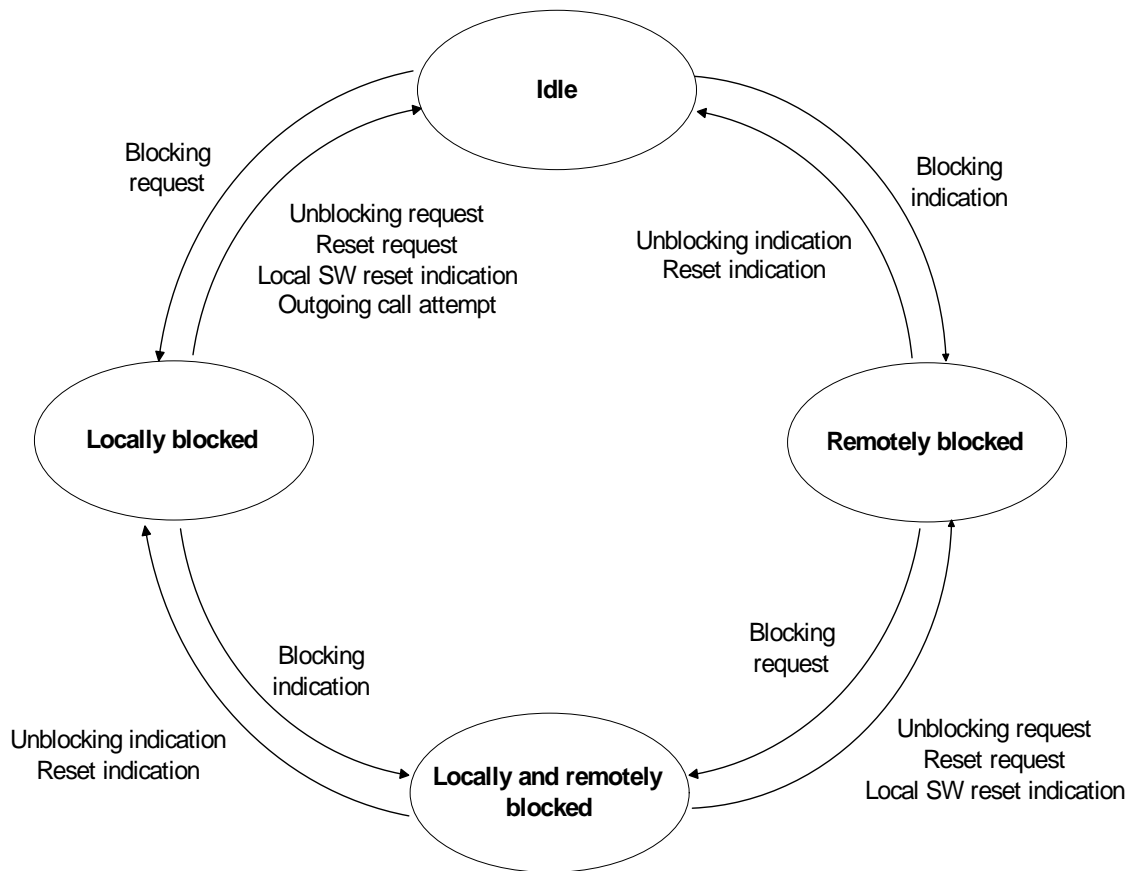



Figure 17 - Circuit states

As shown in Figure 17, a specific circuit can be in four different states namely idle, locally blocked, remotely blocked and “locally and remotely blocked”. A call normally cannot be made if the circuit is not in the ‘idle’ state. There is one exception to that as the TB640 ISUP layer will interpret an outgoing call as an ‘unblocking’ request from its host application if the circuit state was ‘locally blocked’.

 An SS7 application can only change the local state of a circuit. The remote state is always controlled by the remote node.

6.3.3.2 Circuit group functions

SS7 protocol offers some circuit group actions which affect more than one circuit at the same time. Amongst those actions are RESET, BLOCK and UNBLOCK. To understand how those work, the end-user must first know that a group is defined by a mutual agreement between two peer ISUP nodes about which circuits are part of which group. It also means that there is no ‘group ID’ or a similar concept that is carried over SS7 MSUs. When the ISUP layer receives a group command, it has to calculate which circuits are to be affected according to the configuration it has and three parameters from the received command (current CIC, range value and the status bit field). In every group event, the “range and status” IE indicates which circuits are affected by the command. Since the command is also received on a specific circuit, the layer calculates which circuits are affected starting from that circuit. Refer to the IE definition in section 6.3.2.4.10. Note that certain other SS7 equipment requires group functions to be sent on the CIC of the first circuit in a circuit group and will just ignore group commands not addressed that way.

- ✍ For better interoperability with other equipment, always send group commands on the first circuit of a circuit group and fill the range (and/or status bit field) accordingly.

6.3.3.3 Flow control

The SS7 network robustness relies on redundant links, fail-over switching and flow control. Every protocol layer (MTP2, MTP3 and ISUP) has the capability to tell its service user to slow down its rate of input to recover from an abnormal situation on the network. The ISUP layer sends this information to the host application with congestion events (shown in the table below). Although these events are sent on a particular circuit, they affect the whole interface. The host application needs to keep track of those events and to react accordingly by slowing, stopping or restarting the flow of calls toward the ISUP layer. Failure to do so will probably force the stack to drop call request or to react much slower than expect for currently active calls. Thus, the host application **MUST** respect these warnings.

Table 62 - ISUP flow control events

Flow control events	Description
TB640 SS7 ISUP STATUS EVENT CONGESTION LEVEL 0	Very low congestion on the interface
TB640 SS7 ISUP STATUS EVENT CONGESTION LEVEL 1	Low congestion on the interface
TB640 SS7 ISUP STATUS EVENT CONGESTION LEVEL 2	High congestion on the interface
TB640 SS7 ISUP STATUS EVENT CONGESTION LEVEL 3	Heavy congestion on the interface
TB640 SS7 ISUP STATUS EVENT STOP CONGESTION	No more congestion on the interface
TB640 SS7 ISUP STATUS EVENT PAUSE INDICATION	Interface (i.e. remote DPC) is no longer accessible
TB640 SS7 ISUP STATUS EVENT RESUME INDICATION	Interface (i.e. remote DPC) is now accessible again

Causes of interface congestion may vary greatly and may come from multiple sources. For example, if an intermediate SS7 node in the path toward a specific destination point code becomes congested (i.e. lack of CPU power), the local interface will be also notified of the congestion to apply proper routing through an alternate route (if available). Thus, the congestion is not always caused by the local SS7 nodes. This enforces the fact that the host application needs to be “nice” otherwise the local stack will apply its own flow control and start refusing calls. The interface can also be flagged as ‘paused’ which means that there is no more a route to reach the destination point code assigned to the circuit. In this case, the ISUP layer will do the configured actions on the

opened and transient calls as specified by the *PauseAction* parameter from the interface configuration structure (refer to section 6.2.2.4).

6.3.3.4 Circuit continuity testing

As the SS7 signaling is out-of-band compared to the circuits (voice/data channels), there is sometimes the need to validate the integrity of a circuit before using it to establish connections. This is especially true for connections having to go through a satellite path. There are two methods of initiating a continuity testing: within a call or outside a call). Those two methods are described respectively in sections 6.3.3.4.1 and 6.3.3.4.2.

During setup time, a circuit is configured with options. These options are used by the ISUP layer when a remote SS7 node queries information about a circuit and don't affect the behavior of the layer. Three of those options are `CONTINUITY_CHECK_NONE`, `CONTINUITY_CHECK_STATISTICAL` and `CONTINUITY_CHECK_PERCALL`. When the first option (NONE) is used, the host application should not issue any continuity check on the specific circuit. There is no side effect to issue a continuity check on such a circuit other than not respecting the local node configuration as seen by other SS7 nodes. When the last option is used (PER_CALL), the host application should issue a continuity check for every outgoing call. The remaining option (STATISTICAL) let the host application decide on a per-call basis if continuity testing should be done.

Since the circuit options don't influence the behavior of the ISUP layer, the three ways to issue continuity checks are to 1) set the '*fContChkForOutgoing*' field in the circuit configuration structure upon allocation or 2) set the '*continuity test required*' field of the '*nature of connection indicators*' IE into an outgoing IAM message or 3) send a CCR (continuity check request) primitive upon an idle circuit. The '*fContChkForOutgoing*' should be set to `TBX_TRUE` when the circuit option `CONTINUITY_CHECK_PERCALL` is used to be consistent with the configuration. This will instruct the ISUP layer to force the proper IE configuration for every outgoing call.

- ✍ When using configuration parameter '*fContChkForOutgoing*', the TB640 will enforce the '*continuity check required*' field in the '*nature of connection indicators*' IE for every outgoing call. To be consistent, the host application should set the circuit option to `CONTINUITY_CHECK_PERCALL`.

6.3.3.4.1 Continuity testing within a call

This circuit testing occurs when the forward side (originating side) sets the '*continuity test required*' field of the '*nature of connection indicators*' IE into the IAM message. This indicates to the backward end to activate a physical loopback (see note below) on the circuit associated with the call. Then, the forward side can proceed with the circuit testing, usually by sending a test tone and analyzing the returned data. The forward side indicates to the backward side about the result of the test by sending a COT (continuity test report) which will then remove the loopback from the circuit. In case of success, the call can proceed as usual (see call flow 6.3.4.9). If the test fails, the forward side then starts a '*continuity re-check*' sequence by sending a CCR (continuity check request) (see call flow 0). Because the first test fails, it is not recommended to keep the pending call opened and, therefore, the host application should consider it as closed. Upon reception of the CCR, the backward side re-activates the loopback on the circuit and waits for the test report (COT). Again, after analyzing the test data, the forward end sends a report (COT) to the backward end. In

case of success, the circuit is returned to the idle state. In case of failure, it is recommended that the host applications (both forward and backward) block the circuit to prevent subsequent calls on the circuit (see call flow 6.3.4.12).

- ✍ If the backward-end circuit is located on a TB640 blade, there is an easy way to do a trunk resource physical loopback. The application only needs to allocate a full-duplex trunk resource (which should already been done anyway if it plans to receive calls on this specific trunk) and connect it on itself with a half-duplex connection (source connected on destination) using the connection API. To achieve this, use the same hTrunkRes as the source and destination in the TB640_MSG_ID_CONN_OP_CREATE message and set the connection type to half-duplex. No other resource is required.

6.3.3.4.2 *Continuity testing outside a call*

This circuit testing occurs when one of the nodes wants to validate a currently idle circuit. It is recommended that the switch first blocks the circuit to prevent calls to use the circuit during the testing. The testing pattern is similar as for the continuity testing within a call except that the initial testing is indicated by a 'CCR' sent by the testing side (instead of being embedded in an IAM primitive). Refer to call flows 6.3.4.14, 6.3.4.15 and 6.3.4.16.

6.3.4 Call flow and scenarios

The following sections describe different call scenarios between two host applications exchanging calls through SS7 ISUP layer. The scenarios show the TB640 API requests and notifications that are sent and received to obtain the specific call flow. They also show the correspondent SS7 message that will be sent on the signaling link(s) at every step of the call. Below is a brief description of each of those SS7 messages (also referred to as MSUs).

IAM (Initial Address Message)

This message is the very first message sent by the originating side to try establishing a call. It contains the routing information (i.e. the called number) and other data (i.e. optionally the calling number, etc) for the peer to accept or refused the call. The called number does not need to be complete. In this case, the remote side could request for other information or decide to release the call.

ACM (Address Complete Message)

This message is the acknowledgment by the receiving side that the routing info (i.e. the called number) is sufficient to proceed with the call. Usually, this indicates to the originating switch to start the audible ringing. At this point, all the circuits are reserved and maybe also connected (except for the two ends).

ANM (Answer Message)

This message is sent when the receiving user picks up the phone. At this point, the two end switches connect their respective circuit to the end-users phone and the conversation can occur.

REL (Release)

This message is sent by either end to specify the termination of the connection. It also indicates to all switches within the path to release the circuits used during the call.

RLC (Release Complete)

This message is the confirmation that the circuits have been properly released or reset.

CPG (Call Progress)

This message is sent to report progress information (i.e. status) during the call establishment procedure.

SAM (Subsequent Address Message)

This message is used during “overlap” signaling by the originating end to send more digit information to the receiving end. One or more digits can be sent within a single message depending on the IE content.

CCR (Continuity check requests)

This message is sent to request the peer to put a loopback on a specific circuit to perform a line testing. The peer actually does not have to do anything for the test apart from creating a loop on the circuit. The originator of CCR however will issue a test tone onto the circuit and will analyze the returned data.

COT (Continuity)

This message is sent to report the success or failure of a continuity check.

LPA (Loopback acknowledge)

This message is sent to confirm that a loopback has been activated on a circuit.

RSC (Reset circuit)

This message is sent to reset the state of a single circuit.

GRS (Circuit group reset)

This message is sent to reset a group of circuits.

GRA (Circuit group reset acknowledge)

This message is sent to answer (positive or negative acknowledge) of a previous GRS command.

BLO (Circuit block)

This message is sent to indicate that the local ISUP layer considers a specific circuit as locally blocked. This command does not affect the remote blocking state of a circuit.

BLA (Circuit blocking acknowledge)

This message is sent to acknowledge a previous BLO command.

UBL (Circuit unblock)

This message is sent to indicate that the local ISUP layer considers a specific circuit as locally unblocked. This command does not affect the remote blocking state of a circuit.

UBA (Circuit unblocking acknowledge)

This message is sent to acknowledge a previous UBL command.

CGB (Circuit group block)

This message is sent to indicate that the local ISUP layer considers a group of circuit as locally blocked. This command does not affect the remote blocking state of circuits.

CGBA (Circuit group block acknowledge)

This message is sent to acknowledge a previous CGB command.

CGU (Circuit group unblock)

This message is sent to indicate that the local ISUP layer considers a group of circuit as locally unblocked. This command does not affect the remote blocking state of circuits.

CGUA (Circuit group unblock acknowledge)

This message is sent to acknowledge a previous CGU command.

To get more details about specific SS7 primitive, the application designer must refer to the appropriate SS7 specifications. With this basic view of the messages, the following sections will demonstrate basic call flows for simple SS7 ISUP calls in “enbloc” and “overlap” mode:

6.3.4.1 Successful basic call setup (no ACM)

Table 63 - ISUP successful basic call setup (no ACM)

Outgoing side	Network	Incoming side
OP_CONN_REQUEST →		
	IAM →	
		NOTIF_CONN_INDICATION →
		<resource is available and user answers>
		← OP_CONN_RESPONSE
	← ANM	
← NOTIF_CONN_CONFIRMED		

6.3.4.2 Successful basic call setup (with ACM)

Table 64 - ISUP Successful basic call setup (with ACM)

Outgoing side	Network	Incoming side
OP_CONN_REQUEST →		
	IAM →	
		NOTIF_CONN_INDICATION →
		<resource is available>
		← OP_CONN_STATUS_REQUEST(Address Complete)
	← ACM	

← NOTIF_CONN_STATUS_INDICATION (Address complete)		<user answers>
		← OP_CONN_RESPONSE
	← ANM	
← NOTIF_CONN_CONFIRMED		

6.3.4.3 Successful call setup (with CPG)

Table 65 - ISUP successful call setup (with CPG)

Outgoing side	Network	Incoming side
OP_CONN_REQUEST →		
	IAM →	
		NOTIF_CONN_INDICATION →
		<resource is available>
		← OP_CONN_STATUS_REQUEST (Address Complete)
	← ACM	
← NOTIF_CONN_STATUS_INDICATION (Address complete)		<delay in the call establishment such as bridging to another network>
		← OP_CONN_STATUS_REQUEST (Call progress)
	← CPG	
← NOTIF_STATUS_INDICATION (Call progress)		<user answers>
		← OP_CONN_RESPONSE
	← ANM	
← NOTIF_CONN_CONFIRMED		

6.3.4.4 Successful call setup (overlap)

Table 66 - ISUP successful call setup (overlap)

Outgoing side	Network	Incoming side
OP_CONN_REQUEST →		
	IAM →	
		NOTIF_CONN_INDICATION →
		<Number is incomplete>
OP_CONN_STATUS_REQUEST (Subsequent addr.) →		
	SAM →	
		NOTIF_CONN_STATUS_INDICATION (Subsequent addr.) →
OP_CONN_STATUS_REQUEST (Subsequent addr.) →		
	SAM →	
		NOTIF_CONN_STATUS_INDICATION (Subsequent addr.) →
		<Number is complete enough and resource is available>
		← OP_CONN_STATUS_REQUEST (Address Complete)
	← ACM	
← NOTIF_CONN_STATUS_INDICATION (Address complete)		<user answers>
		← OP_CONN_RESPONSE
	← ANM	
← NOTIF_CONN_CONFIRMED		

6.3.4.5 Successful call release

Table 67 - ISUP successful call release

Releasing side	Network	Other side
Call is already established		
OP_RELEASE_REQUEST	REL →	
		NOTIF_RELEASE_INDICATION →
		<resource are now freed>
		← OP_RELEASE_RESPONSE
	← RLC	
← NOTIF_RELEASE_CONFIRMED		

6.3.4.6 Outgoing call refused by incoming side

Table 68 - ISUP outgoing call refused by incoming side

Outgoing side	Network	Incoming side
OP_CONN_REQUEST →		
	IAM →	
		NOTIF_CONN_INDICATION →
		<resource is not available >
		← OP_RELEASE_REQUEST
	← REL	
← NOTIF_RELEASE_INDICATION		
OP_RELEASE_REQUEST →		
	RLC →	
		NOTIF_RELEASE_CONFIRMED →

6.3.4.7 Call collision because racing condition host ↔ TB640

This scenario assumes the host application has sent a connection request while a connection indication for the same circuit was already in the communication pipe between the TB640 and the host.

Table 69 - ISUP call collision (race condition)

Host application	TB640	Network
		<-IAM (Circuit = 1)
	← NOTIF_STATUS_INDICATION (Circuit = 1)	
OP_CONN_REQUEST (Circuit = 1) →		
← NOTIF_CONN_INDICATION (Circuit = 1)		
← NOTIF_STATUS_INDICATION (Reattempt)		

6.3.4.8 Call collision between two SS7 nodes

This scenario assumes the right hand side to be the controlling switch for circuit 1 per Q.764, Section 2.9.1.4 [higher point switch controls even numbered CICs].

Table 70 - ISUP call collision between two SS7 nodes

TB640	Network	TB640
OP_CONN_REQUEST (Circuit = 1) →		← OP_CONN_REQUEST (Circuit = 1)
	← IAM →	
← NOTIF_STATUS_INDICATION (Reattempt)		
← NOTIF_CONN_INDICATION		

6.3.4.9 Call with successful continuity testing

Table 71 - ISUP call with successful continuity testing

Outgoing side	Network	Incoming side
OP_CONN_REQUEST → (The IE 'nature of connection indicators' must have the 'continuity check required' field set to 0x1)		
	IAM →	
<Connects a tone generator and detector. Start playing a test tones on the targeted circuit>		NOTIF_CONN_INDICATION → <activates a loopback on the targeted circuit>
		← OP_STATUS_REQUEST(LOOPBACK_ACTIVATED)
	← LPA	This is optional for some variants
	...	
<Tone is detected back correctly> <Disconnect tone resources>		
OP_CONN_STATUS_REQUEST(CONTINUITY_REPORT) with success →		
	COT →	
		NOTIF_STATUS_INDICATION (CONTINUITY_REPORT) → <deactivates a loopback on the targeted circuit>
		← OP_CONN_STATUS_REQUEST(Address Complete)
	← ACM	
← NOTIF_CONN_STATUS_INDICATION (Address complete)		<user answers>
		← OP_CONN_RESPONSE
	← ANM	
← NOTIF_CONN_CONFIRMED		

6.3.4.10 Call with failed continuity testing and successful re-check

Table 72 - ISUP call with failed continuity testing and successful re-check

Outgoing side	Network	Incoming side
OP_CONN_REQUEST → (The IE 'nature of connection indicators' must have the 'continuity check required' field set to 0x1)		
	IAM →	
<Connects a tone generator and detector. Start playing a test tones on the targeted circuit>		NOTIF_CONN_INDICATION → <activates a loopback on the targeted circuit>
		← OP_STATUS_REQUEST(LOOPBACK_ACTIVATED)
	← LPA	This is optional for some variants
	...	
<Tone has NOT been detected correctly>		
OP_CONN_STATUS_REQUEST(CONTINUITY_REPORT) with failure →		
	COT →	
		NOTIF_STATUS_INDICATION (CONTINUITY_REPORT) with failure result → <deactivates a loopback on the targeted circuit>
< Call should be reattempted on another circuit >		
OP_CONN_STATUS_REQUEST(CONTINUITY_REQUEST) →	CCR ->	
		NOTIF_STATUS_INDICATION (CONTINUITY_REQUEST) → <activates a loopback on the targeted circuit>
		← OP_STATUS_REQUEST(LOOPBACK_ACTIVATED)
	← LPA	This is optional for some variants
← NOTIF_STATUS_INDICATION (LOOPBACK_ACK)		
	...	
<Tone is detected back correctly> <Disconnect tone resources>		
OP_CONN_STATUS_REQUEST(CONTINUITY_REPORT) with success → * This is optionnal	COT →	
		NOTIF_STATUS_INDICATION (CONTINUITY_REPORT) with success result → * This is optionnal
OP_RELEASE_REQUEST	REL →	
		NOTIF_RELEASE_INDICATION → <deactivates a loopback on the targeted circuit> <resource are now freed>
		← OP_RELEASE_RESPONSE
	← RLC	
← NOTIF_RELEASE_CONFIRMED		

6.3.4.11 Call with failed continuity testing and no re-check

Table 73 - ISUP call with failed continuity testing and no re-check

Outgoing side	Network	Incoming side
OP_CONN_REQUEST → (The IE 'nature of connection indicators' must have the 'continuity check required' field set to 0x1)		
	IAM →	
<Connects a tone generator and detector. Start playing a test tones on the targeted circuit>		NOTIF_CONN_INDICATION → <activates a loopback on the targeted circuit>
		← OP_STATUS_REQUEST(LOOPBACK_ACTIVATED)
	← LPA	This is optional for some variants
	...	
		<Timer T8 expires>
← NOTIF_RELEASE_INDICATION	← REL	NOTIF_RELEASE_INDICATION →
<Disconnect tone resources>		<deactivates a loopback on the targeted circuit>
OP_RELEASE_RESPONSE →	RLC →	← OP_RELEASE_RESPONSE

6.3.4.12 Call with failed continuity testing and failed re-check

Table 74 - ISUP call with failed continuity testing and failed re-check

Outgoing side	Network	Incoming side
OP_CONN_REQUEST → (The IE 'nature of connection indicators' must have the 'continuity check required' field set to 0x1)		
	IAM →	
<Connects a tone generator and detector. Start playing a test tones on the targeted circuit>		NOTIF_CONN_INDICATION → <activates a loopback on the targeted circuit>
		← OP_STATUS_REQUEST(LOOPBACK_ACTIVATED)
	← LPA	This is optional for some variants
	...	
<Tone has NOT been detected correctly>		
OP_CONN_STATUS_REQUEST(CONTINUITY_REPORT) with failure →		
	COT →	
		NOTIF_STATUS_INDICATION (CONTINUITY_REPORT) with failure result →
		<deactivates a loopback on the targeted circuit>
< Call should be reattempted on another circuit >		
OP_CONN_STATUS_REQUEST(CONTINUITY_REQUEST) →	CCR ->	
		NOTIF_STATUS_INDICATION (CONTINUITY_REQUEST) →
		<activates a loopback on the targeted circuit>
		← OP_STATUS_REQUEST(LOOPBACK_ACTIVATED)
	← LPA	This is optional for some variants
← NOTIF_STATUS_INDICATION (LOOPBACK_ACK)		
	...	
<Tone has NOT been detected correctly> <Disconnect tone resources>		
OP_CONN_STATUS_REQUEST(CONTINUITY_REPORT) with failure →		
	COT →	
		NOTIF_STATUS_INDICATION (CONTINUITY_REPORT) with failure result →
		<deactivates a loopback on the targeted circuit>
	See Note	

NOTE: CCR, LPA, COT sequence continues until COT with succeed which is followed by REL/RLC sequence.

6.3.4.13 Call with failed continuity testing and failed re-check (timeout)

Table 75 - ISUP call with failed continuity testing and failed re-check (timeout)

Outgoing side	Network	Incoming side
OP_CONN_REQUEST → (The IE 'nature of connection indicators' must have the 'continuity check required' field set to 0x1)		
	IAM →	
<Connects a tone generator and detector. Start playing a test tones on the targeted circuit>		NOTIF_CONN_INDICATION → <activates a loopback on the targeted circuit>
		← OP_STATUS_REQUEST(LOOPBACK_ACTIVATED)
	← LPA	This is optional for some variants
	...	
<Tone has NOT been detected correctly>		
OP_CONN_STATUS_REQUEST(CONTINUITY_REPORT) with failure →		
	COT →	
		NOTIF_STATUS_INDICATION (CONTINUITY_REPORT) with failure result →
		<deactivates a loopback on the targeted circuit>
< Call should be reattempted on another circuit >		
OP_CONN_STATUS_REQUEST(CONTINUITY_REQUEST) →	CCR ->	
		NOTIF_STATUS_INDICATION (CONTINUITY_REQUEST) →
		<activates a loopback on the targeted circuit>
		←
		OP_STATUS_REQUEST(LOOPBACK_ACTIVATED)
	← LPA	This is optional for some variants
← NOTIF_STATUS_INDICATION (LOOPBACK_ACK)		
	...	
		<Timer T34 expires>
	← RSC	NOTIF_STATUS_INDICATION (LOCAL_RESET) →
		<deactivates a loopback on the targeted circuit>
← NOTIF_STATUS_INDICATION (LOCAL_RESET)		
<Disconnect tone resources>		
OP_CONN_STATUS_REQUEST(RESET_RESPONSE) →		
<The application should locally block the circuit>		NOTIF_STATUS_INDICATION (RESET_CONFIRM) →
		<The application should locally block the circuit>

6.3.4.14 Successful continuity testing out-of-call

Table 76 - ISUP successful continuity testing out-of-call

Outgoing side	Network	Incoming side
OP_CONN_STATUS_REQUEST(CONTINUITY_REQUEST) →	CCR ->	
		NOTIF_STATUS_INDICATION (CONTINUITY_REQUEST) →
		<activates a loopback on the targeted circuit>
		← OP_STATUS_REQUEST(LOOPBACK_ACTIVATED)
	← LPA	This is optional for some variants
← NOTIF_STATUS_INDICATION (LOOPBACK_ACK)		
	...	
<Tone is detected back correctly>		
<Disconnect tone resources>		
OP_CONN_STATUS_REQUEST(CONTINUITY_REPORT) with success →	COT →	
		NOTIF_STATUS_INDICATION (CONTINUITY_REPORT) with success result →
		<deactivates a loopback on the targeted circuit>
← NOTIF_RELEASE_INDICATION	← REL	NOTIF_RELEASE_INDICATION →
<Disconnect tone resources>		<deactivates a loopback on the targeted circuit>
OP_RELEASE_RESPONSE →	RLC →	← OP_RELEASE_RESPONSE

6.3.4.15 Failed continuity testing out-of-call with successful re-check

Table 77 - ISUP failed continuity testing out-of-call with successful re-check

Outgoing side	Network	Incoming side
OP_CONN_STATUS_REQUEST(CONTINUITY_REQUEST) →	CCR ->	
		NOTIF_STATUS_INDICATION (CONTINUITY_REQUEST) →
		<activates a loopback on the targeted circuit>
		←
		OP_STATUS_REQUEST(LOOPBACK_ACTIVATED)
	← LPA	This is optional for some variants
← NOTIF_STATUS_INDICATION (LOOPBACK_ACK)		
	...	
<Tone has NOT been detected correctly>		
OP_CONN_STATUS_REQUEST(CONTINUITY_REPORT) with failure →	COT →	
		NOTIF_STATUS_INDICATION (CONTINUITY_REPORT) with failure result →
		<deactivates a loopback on the targeted circuit>
OP_CONN_STATUS_REQUEST(CONTINUITY_REQUEST) →	CCR ->	
		NOTIF_STATUS_INDICATION (CONTINUITY_REQUEST) →
		<activates a loopback on the targeted circuit>
		←
		OP_STATUS_REQUEST(LOOPBACK_ACTIVATED)
	← LPA	This is optional for some variants
← NOTIF_STATUS_INDICATION (LOOPBACK_ACK)		
	...	
<Tone is detected back correctly> <Disconnect tone resources>		
OP_CONN_STATUS_REQUEST(CONTINUITY_REPORT) with success →	COT →	
		NOTIF_STATUS_INDICATION (CONTINUITY_REPORT) with success result →
← NOTIF_RELEASE_INDICATION	← REL	NOTIF_RELEASE_INDICATION →
<Disconnect tone resources>		<deactivates a loopback on the targeted circuit>
OP_RELEASE_RESPONSE →	RLC →	← OP_RELEASE_RESPONSE

6.3.4.16 Failed continuity testing out-of-call with failed re-check

Table 78 - ISUP failed continuity testing out-of-call with failed re-check

Outgoing side	Network	Incoming side
OP_CONN_STATUS_REQUEST(CONTINUITY_REQUEST) →	CCR ->	
		NOTIF_STATUS_INDICATION (CONTINUITY_REQUEST) →
		<activates a loopback on the targeted circuit>
		←
		OP_STATUS_REQUEST(LOOPBACK_ACTIVATED)
	← LPA	This is optional for some variants
← NOTIF_STATUS_INDICATION (LOOPBACK_ACK)		
	...	
<Tone has NOT been detected correctly>		
OP_CONN_STATUS_REQUEST(CONTINUITY_REPORT) with failure →	COT →	
		NOTIF_STATUS_INDICATION (CONTINUITY_REPORT) with failure result →
		<deactivates a loopback on the targeted circuit>
	See Note	

NOTE: CCR, LPA, COT sequence continues until COT with succeed which is followed by REL/RLC sequence.

Circuit reset

Table 79 - ISUP circuit reset

Side A (incoming / outgoing)	Network	Side B (outgoing / incoming)
...assuming a call is already in progress (regardless of its state)...		
OP_CONN_STATUS_REQUEST(RESET_REQUEST) →	RSC ->	
		NOTIF_STATUS_INDICATION (RESET_REQUEST) →
		<circuit must be considered as idle>
		← OP_CONN_STATUS_REQUEST(RESET_RESPONSE)
	← RLC	
← NOTIF_STATUS_INDICATION (RESET_CONFIRM)		
<circuit must be considered as idle>		

6.3.4.17 Circuit group reset

Table 80 - ISUP circuit group reset

Side A (incoming / outgoing)	Network	Side B (outgoing / incoming)
...assuming calls are already in progress (regardless of their state)...		
OP_CONN_STATUS_REQUEST(GRP_RESET_REQUEST)*→	GRS ->	
		NOTIF_STATUS_INDICATION (GRP_RESET_REQUEST) →
		<All circuits indicated in the 'range and status' IE must be considered as idle>
		← OP_CONN_STATUS_REQUEST(GRP_RESET_RESPONSE)**
	← GRA	
← NOTIF_STATUS_INDICATION (GRP_RESET_CONFIRM)		
<circuit must be considered as idle>		

* The group reset request MUST be sent on a circuit that is part of the circuit group the user wants to reset (partially or not).

** The group reset response MUST be sent on the same circuit on which the group reset request was received.

6.3.4.18 Circuit blocking

Table 81 - ISUP circuit blocking

Side A (incoming / outgoing)	Network	Side B (outgoing / incoming)
...assuming a call is already in progress (regardless of its state)...		
OP_CONN_STATUS_REQUEST(BLOCKING_REQUEST) →	BLO ->	
		NOTIF_STATUS_INDICATION (BLOCKING_REQUEST) →
		← OP_CONN_STATUS_REQUEST(BLOCKING_RESPONSE)
	← BLA	
← NOTIF_STATUS_INDICATION (BLOCKING_CONFIRM)		
... there is no effect on a call that is currently in progress on the circuit. Once the call is over, side B must consider the circuit as remotely blocked and side A must consider the circuit as locally blocked. Only test calls are allowed on a blocked circuit.		

6.3.4.19 Circuit group blocking

Table 82 - ISUP circuit group blocking

Side A (incoming / outgoing)	Network	Side B (outgoing / incoming)
...assuming calls are already in progress (regardless of their state)...		
OP_CONN_STATUS_REQUEST(GRP_BLOCKING_REQUEST)*→	CGB ->	
		NOTIF_STATUS_INDICATION (GRP_BLOCKING_REQUEST) →
		← OP_CONN_STATUS_REQUEST(GRP_BLOCKING_RESPONSE)**
	← CGBA	

← NOTIF_STATUS_INDICATION (GRP_BLOCKING_CONFIRM)		
... there is no effect on any call that is currently in progress on circuits. Once a call is over, side B must consider the circuit as remotely blocked and side A must consider the circuit as locally blocked. Only test calls are allowed on a blocked circuit.		

- * The group blocking request MUST be sent on a circuit that is part of the circuit group the user wants to block (partially or not).
- ** The group blocking response MUST be sent on the same circuit on which the group blocking request was received.

6.3.4.20 Circuit unblocking

Table 83 - ISUP circuit unblocking

Side A (incoming / outgoing)	Network	Side B (outgoing / incoming)
...assuming a test call is already in progress (regardless of its state)...		
OP_CONN_STATUS_REQUEST(UNBLOCKING_REQUEST) →	UBL ->	
		NOTIF_STATUS_INDICATION (UNBLOCKING_REQUEST) →
		← OP_CONN_STATUS_REQUEST(UNBLOCKING_RESPONSE)
	← UBA	
← NOTIF_STATUS_INDICATION(UNBLOCKING_CONFIRM)		
... there is no effect on a test call that is currently in progress on the circuit. Once the test call is over, side B must consider the circuit as remotely unblocked and side A must consider the circuit as locally unblocked. Only test calls are allowed on a blocked circuit.		

6.3.4.21 Circuit group unblocking

Table 84 - ISUP circuit group unblocking

Side A (incoming / outgoing)	Network	Side B (outgoing / incoming)
...assuming test calls are already in progress (regardless of their state)...		
OP_CONN_STATUS_REQUEST(GRP_UNBLOCKING_REQUEST)* →	CGU ->	
		NOTIF_STATUS_INDICATION (GRP_UNBLOCKING_REQUEST) →
		← OP_CONN_STATUS_REQUEST(GRP_UNBLOCKING_RESPONSE)**
	← CGUA	
← NOTIF_STATUS_INDICATION (GRP_UNBLOCKING_CONFIRM)		
... there is no effect on any test call that is currently in progress on circuits. Once a test call is over, side B must consider the circuit as remotely unblocked and side A must consider the circuit as locally unblocked. Only test calls are allowed on a blocked circuit.		

- * The group unblocking request MUST be sent on a circuit that is part of the circuit group the user wants to unblock (partially or not).
- ** The group unblocking response MUST be sent on the same circuit on which the group unblocking request was received.

7 SCCP

7.1 Overview

This section gives a description of the TB640 Signaling Connection Control Part (SCCP) layer architecture and usage. This layer is referred to as SCCP in the rest of the section.

7.1.1 Summary

The SCCP layer provides connectionless network services and an address translation mechanism, called global title translation (GTT), capability above MTP3. A global title can translate a string of digits (e.g., a dialled 800 number, calling card number, or mobile subscriber identification number) into a destination point code and/or subsystem number. A subsystem number uniquely identifies an application at the destination signaling point. SCCP provides the transport layer for TCAP-based services within the OSI model.

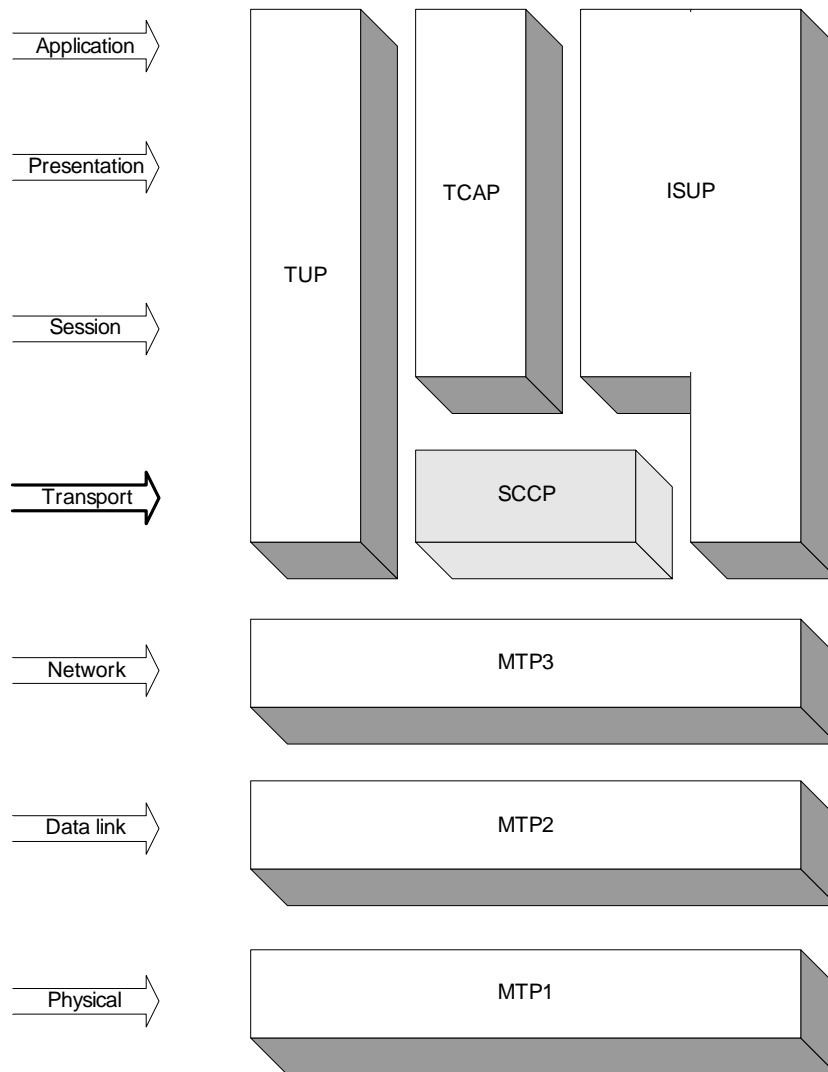


Figure 18 - SCCP OSI model

7.1.2 Features

SCCP supports the following features:

- Routing and translation functions.
- Management functions.
- Transfer of data without logical signaling connections.

The SCCP software supports two classes of service, in connectionless mode:

- Class 0: Basic connectionless class.
- Class 1: Sequenced connectionless class.

7.1.3 Architecture

The SCCP software supports these functions:

- Connectionless service.
- Management.
- Routing.
- Interworking.
- Traffic limitation and congestion control mechanisms (ITU-96).

SCCP provides the functions and procedures to transfer data across an SS7 network. SCCP is a service provider to TCAP and a service user of MTP Level 3 (as shown on Figure 19).

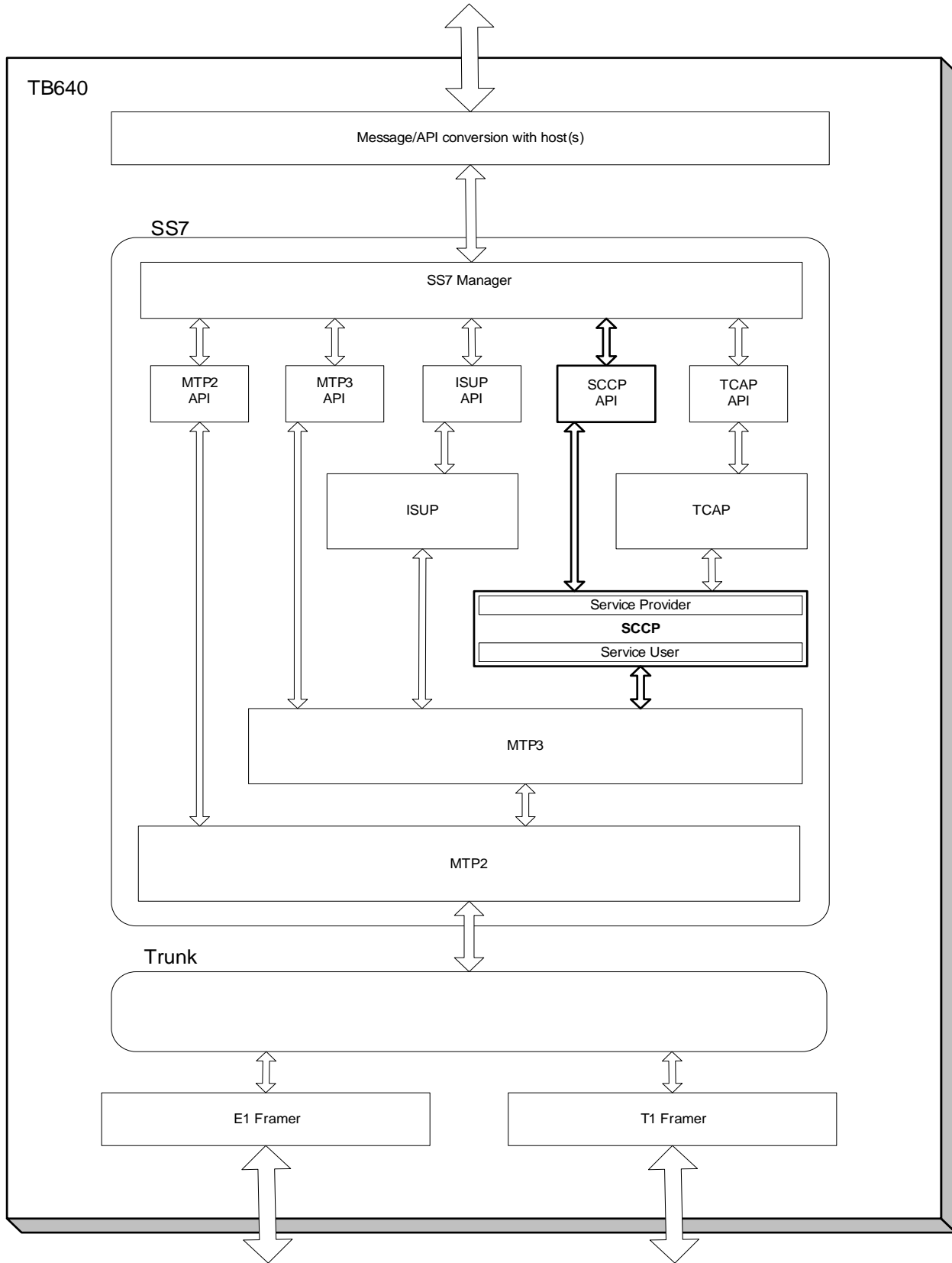


Figure 19 - SCCP layer organization within TB640

A SCCP layer has a Network section which represents a network identifier for a specific protocol variant. For a Network you must define:

- LSAP(s)
- Routes (for a specific or a range of dpc)
- Userpart

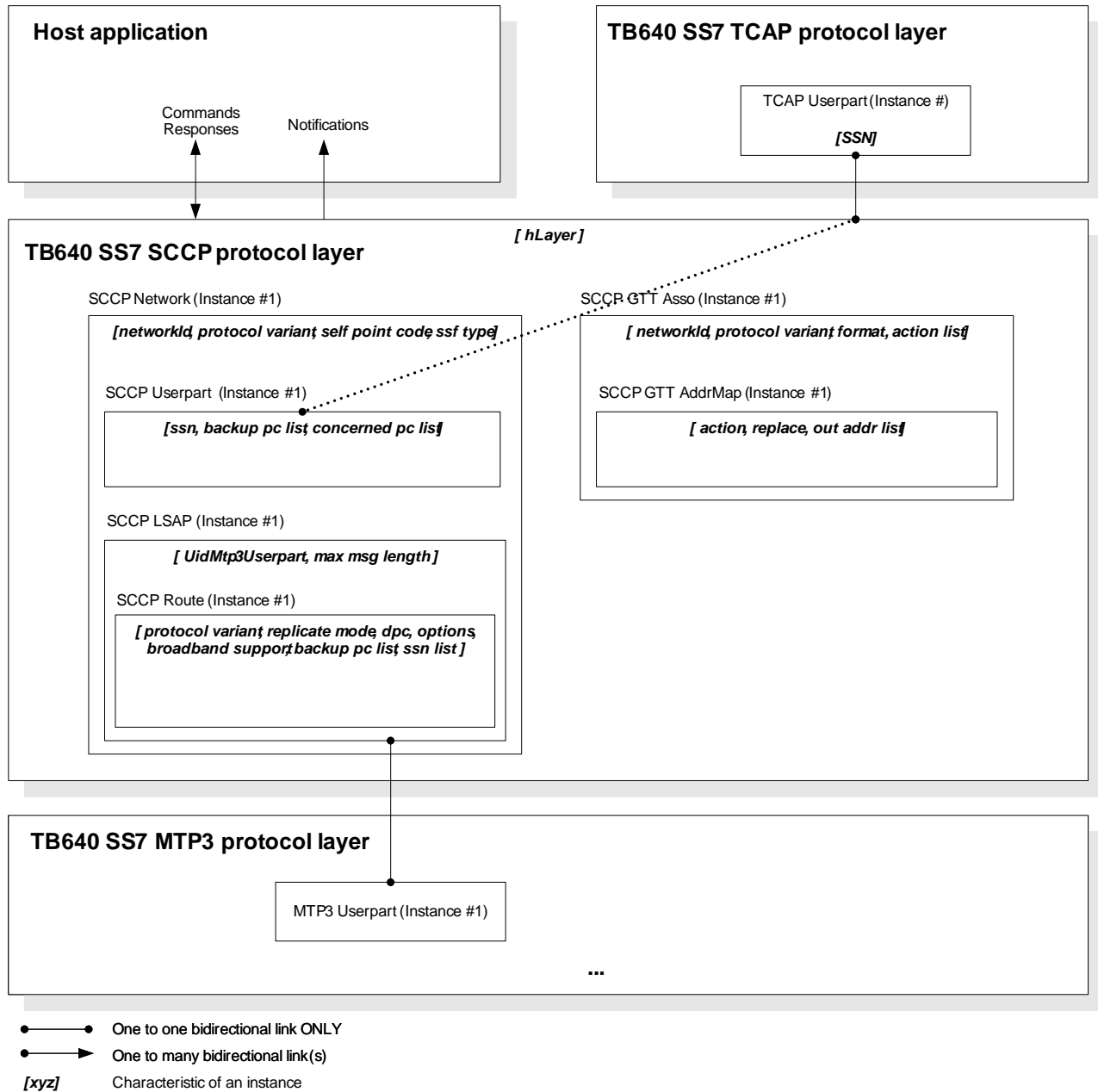


Figure 20 - SCCP layer hierarchy

SCCP Lsap must be associated with one MTP3 Userpart. It's a 1 to 1 association.

A SCCP Route indicates a destination point code (DPC) that is an accessible from this SCCP configuration node and their SSN.

SCCP Userpart can be used in conjunction with an above local TCAP layer and can be used as a standalone and communicates with remote TCAP⁸ layer (host application). See the connection mode table in the SCCP Userpart Configuration section.

A SCCP layer has a Userpart section which represents a subsystem number.

7.1.3.1 Connectionless Service

These functions are provided for connectionless service:

- Transport user data.
- Map the network address to signalling relations.
- Sequence service classification.
- Provide segmentation and reassembly of user data.

7.1.3.2 Management

These functions are provided for management:

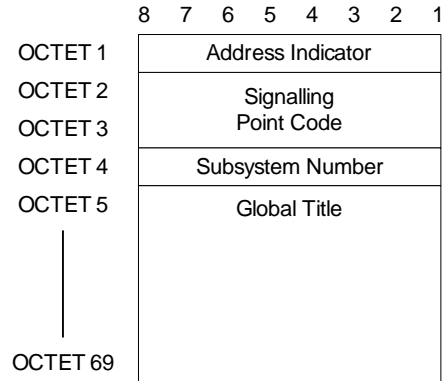
- SCCP subsystem status check.
- Notification of changes in the status of an SCCP subsystem to other SCCP subsystems.
- Service withdrawing of an SCCP subsystem.

7.1.3.3 Addressing

The SCCP software supports the numbering plan contained in the 1996 ITU Q.713 - Q.714 and ANSI T1.112 recommendations.

The SCCP software places no constraints on the numbering scheme except those outlined in the 1996 ITU recommendations.

Figure 21 - SCCP Addressing (ITU) illustrates the ITU addressing architecture for SCCP. The SCCP software supports from 1 to 128 digits (64 octets) within the address field.



Address Indicator:

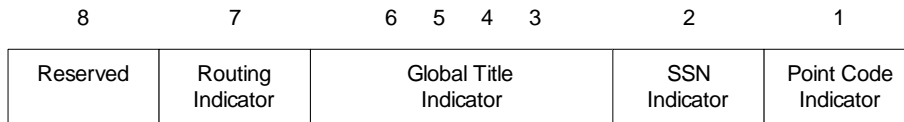
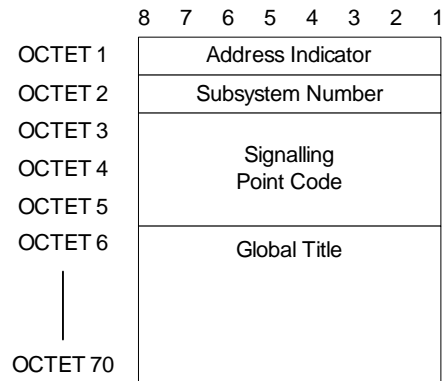


Figure 21 - SSCP Addressing (ITU)

Figure 22 - SSCP Addressing (ANSI) illustrates the ANSI addressing architecture in SSCP:



Address Indicator:

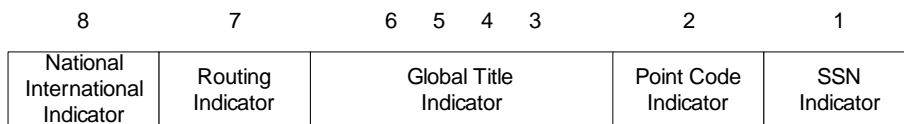


Figure 22 - SSCP Addressing (ANSI)

The address information can take the following forms:

- DPC + (SSN or global title or both)
- Global title

- Global title + SSN


7.1.3.3.1 Global Title Translation Association

A GTT association defines a particular global title format and associated fields. It is assumed that all the global titles associated with this association have specific translation needs and therefore can be served by a single translation function set.

Each entry in **Table 85** defines a GTT association for ITU.

Table 85 - SCCP GTT association for ITU

Format	Nature Of Address Indicator	Numbering Plan	Encoding Scheme	Translation Type
FORMAT_1	International number	Not required	Not required	Not required
FORMAT_2	Not required	Not required	Not required	01
FORMAT_3	Not required	Telephony numbering plan	BCD odd	02
FORMAT_4	Don't care	ISDN numbering	Don't care	03

 *Don't care* values indicate that all values are allowed for this field in this GTT association. This allows for interoperability in the field scenarios where existing implementations may not be filling some fields of the global title format.

GTT associations are provided as run-time configurable options with TB640_MSG_ID_SS7_SCCP_OP_GTT ASSO_ALLOC message. The customers are required to configure one entry per association for which they require global title translation to be performed in the system. Any incoming GT that requires translation must match at least one configured association.

The GTT associations configuration from customers should ensure that associations do not overlap. For configuring the associations as defined in **Table 85**, the structure is filled as follows:

Association Format 1

```
ProtocolVariant = TB640_SS7_SCCP_PROTOCOL_VARIANT_ITU;
GtFormat = TB640_SS7_SCCP_GT_FORMAT_TYPE_1;
Format1.OddEven = TB640_SS7_SCCP_GT_ODD_EVEN_TYPE_NOT_USED;
Format1.NatureOfAddress = TB640_SS7_SCCP_GT_NATURE_OF_ADDR_IND_INTERNATIONAL;
```

Association Format 2

```
ProtocolVariant = TB640_SS7_SCCP_PROTOCOL_VARIANT_ITU;
GtFormat = TB640_SS7_SCCP_GT_FORMAT_TYPE_2;
Format2.fUseTypeValue = TBX_TRUE;
Format2.un32TypeValue = 1;
```


Association Format 3

```
ProtocolVariant = TB640_SS7_SCCP_PROTOCOL_VARIANT_ITU;  
GtFormat = TB640_SS7_SCCP_GT_FORMAT_TYPE_3;  
Format3.fUseTypeValue = TBX_TRUE;  
Format3.un32TypeValue = 2;  
Format3.NumberingPlan = TB640_SS7_SCCP_GT_NUMB_PLAN_TYPE_TELEPHONY;  
Format3.EncodingScheme = TB640_SS7_SCCP_GT_ENCODING_TYPE_BCD_ODD;
```

Association Format 4

```
ProtocolVariant = TB640_SS7_SCCP_PROTOCOL_VARIANT_ITU;  
GtFormat = TB640_SS7_SCCP_GT_FORMAT_TYPE_4;  
Format4.fUseTypeValue = TBX_TRUE;  
Format4.un32TypeValue = 1;  
Format4.NumberingPlan = TB640_SS7_SCCP_GT_NUMB_PLAN_TYPE_ISDN;  
Format4.EncodingScheme = TB640_SS7_SCCP_GT_ENCODING_TYPE_NOT_USED;  
Format4.NatureOfAddress = TB640_SS7_SCCP_GT_NATURE_OF_ADDR_IND_NOT_USED;
```

The action list defines the list of searching operations to be performed by the GTT function for a particular GTT association. The GTT function performs the search in the same sequence as given by the list. For example, the first action specified in the list is performed first. The action list serves as input to the GTT function for performing the global title translation.

 An actions list **must be** configured for a GTT Association instance.

7.1.3.4 Routing

The SCCP software routes data based on:

- DPC + SSN
- DPC + global title
- DPC + global title + SSN
- Global title
- Global title + SSN
- ISNI + Global title (Valid only for ANSI 96 and GR-246-CORE Issue 5).
- INS + Global title (Valid only for GR-246-CORE Issue 5).

When a message is received from SCCP user (upper layer), the point code determines whether the message is processed further locally or sent to a remote node.

When a message is received from MTP Level 3 (lower layer), the point code is ignored, and the routing indicator determines whether global title translation is required or the message is passed to an SCCP user. Global title translation can result in a remote point code.

SCCP can perform routing based on the ISNI parameter as specified in T1.112.4 (ANSI 1996, GR246-CORE Issue 5, December 2000-Telcordia). SCCP can perform routing based on INS parameter as specified in T1.112.4 (GR246-CORE Issue 5, December 2000 -Telcordia).

7.1.3.5 Traffic Limitation and Congestion Control

SCCP traffic limitation and congestion control procedures defined in ITU 1996 can be enabled. When enabled, SCCP performs traffic limitation by discarding a portion of the user traffic depending on the congestion reported by MTP3 towards the destination and the nodal congestion reported by SCCP at the destination.

7.1.3.6 Interworking

SCCP supports interworking between various ITU-T variants (1988, 1992, and 1996) as described in the Q.715 ITU-T Specification.

From the point of view of the SCCP user, there is only one SCCP. It is therefore the responsibility of the SCCP to take all the necessary measures to select the correct message types to be sent out and to segment messages if necessary. Hence, at certain interworking nodes, a capability needs to be provided that allows conversion of the new message types (LUDT⁷, LUDTS⁷) to XUDT, XUDTS message types provided in the *1993 edition*, therefore allowing the segmenting of a LUDT message into multiple XUDT messages (or possibly multiple LUDT messages if the destination understands them), or truncating a long LUDTS to fit into a single XUDTS message, if necessary.

7.1.3.7 Messages

These messages can be received or transmitted by SCCP across the MTP Level 3 SAP:

- Extended Unit Data (XUDT)
- Extended Unit Data Service (XUDTS)
- Long Unit Data (LUDT⁷) (ITU-T 1996, GR246-CORE Issue 5)
- Long Unit Data Service (LUDTS⁷) (ITU-T 1996, GR246-CORE Issue 5)
- Subsystem-Allowed (SSA)
- Subsystem-Congested (SSC) (ITU-T 1996)
- Subsystem-Out-of-Service-Grant (SOG)
- Subsystem-Out-of-Service-Request (SOR)
- Subsystem-Prohibited (SSP)
- Subsystem-Status-Test (SST)
- Unit Data (UDT)
- Unit Data Service (UDTS)

7.1.4 Specification

The SCCP software conforms to the following standards:

- Q.711 - Functional Description of the Signalling Connection Control Part, ITU-T.
- Q.712 - Definition and Function of Signalling Connection Control Part Messages, ITU-T.
- Q.713 - Signalling Connection Control Part Format and Codes, ITUT.
- Q.714 - Signalling Connection Control Part Procedures, ITU-T.
- Q.715 - Signalling Connection Control Part User Guide, ITU-T.
- Q.716 - Signalling Connection Control Part Performance, ITU-T.

⁷ LUDT and LUDTS are not supported for now.

- Q.752 - Monitoring and Measurements for Signalling System No. 7 Networks, ITU-T.
- Q.786 - SCCP Test Specification, ITU-T.
- T1.112.1 - Functional Description of the Signalling Connection Control Part, ANSI.
- T1.112.2 - Definition and Functions of Signalling Connection Control Part Messages, ANSI.
- T1.112.3 - Signalling Connection Control Part Formats and Codes, ANSI.
- T1.112.4 - Signalling Connection Control Part Procedures, ANSI.
- T1.112.5 - Signalling Connection Control Part Performance, ANSI.
- T1.112.1 - Functional Description of the Signalling Connection Control Part, Telcordia.
- T1.112.2 - Definition and Functions of Signalling Connection Control Part Messages, Telcordia.
- T1.112.3 - Signalling Connection Control Part Formats and Codes, Telcordia.
- T1.112.4 - Signalling Connection Control Part Procedures, Telcordia.
- T1.112.5 - Signalling Connection Control Part Performance, Telcordia.
- China (1994) SCCP Specification.
- JT-Q711 Summary: Functional Description of the Signalling Connection Control Part.
- JT-Q712 Summary: Definition and Functions of Signalling Connection Control Part
- JT-Q713 Summary: Signalling Connection Control Part Formats and Codes.
- JT-Q714 Summary: Signalling Connection Control Part Procedures.
- GSM 08.06 Release 99 Version 8.0.0 - Signalling Transport Mechanism Specification for the Base Station System - Mobile-Service Switching Centre (BSS - MSC) Interface.

SCCP can support different protocol variants:

- ITU (used for ITU88, ITU92, and ITU96)
- ANSI (used for ANSI88, ANSI92, and ANSI96)
- TELCORDIA (GR246-CORE Issue 5 December 2000)
- CHINA 1994
- ETSI GSM 08.06 Release 99 Version 8.0.0
- JAPAN, JT-Q711...JT-Q714 (*not available yet*)

7.2 SCCP Configuration

7.2.1 Configuration of layer







General guidelines for configuration of SCCP for a proper operation include the following:

1. The SCCP general allocation⁸ (TB640_MSG_ID_SS7_SCCP_OP_ALLOC) must precede all other messages (other configuration SCCP alloc, get, states and stats). The response of this message is a SCCP handle.
2. The SCCP Network allocation (TB640_MSG_ID_SS7_SCCP_OP_NETWORK_ALLOC) must be made (with the SCCP handle from **step 1**) for a particular network identifier with a particular variant, self point code and subservice field type. The response of this message is a SCCP Network handle.
3. The SCCP Lsap allocation (TB640_MSG_ID_SS7_SCCP_OP_LSAP_ALLOC) must be made (with the SCCP handle from **step 1**, network handle from **step 2** and the unique identifier of the MTP3 userpart) to connect with the MTP3 userpart. The response of this message is a SCCP Lsap handle.
4. The SCCP Route allocation (TB640_MSG_ID_SS7_SCCP_OP_ROUTE_ALLOC) must be made (with the SCCP handle from **step 1** and the Lsap handle from **step 3**) for a particular variant, replicated mode and DPC. The response of this message is a SCCP Route handle.
5. The SCCP Userpart allocation (TB640_MSG_ID_SS7_SCCP_OP_USERPART_ALLOC) must be made (with the SCCP handle from **step 1** and the network handle from **step 2**) for a particular subsystem number. The response of this message is a SCCP Userpart handle.
6. The SCCP GTT Association allocation (TB640_MSG_ID_SS7_SCCP_OP_GTT ASSO_ALLOC) must be made (with the SCCP handle from **step 1**) for a particular global title association. The response of this message is a SCCP Gtt asso handle.
7. The SCCP GTT Addrmap allocation (TB640_MSG_ID_SS7_SCCP_OP_GTT_ADDRMAP_ALLOC) must be made (with the SCCP handle from **step 1** and the Gtt asso handle from **step 6**) for a particular global title address map. The response of this message is a SCCP Gtt addrmap handle.



For any reconfiguration with a SCCP (TB640_MSG_ID_SS7_SCCP_OP_xyz_SET_PARAMS) message, all reconfigurable parameters must be filled appropriately even if the intention is to modify a single parameter.

⁸ All fields of a configuration message alloc must be filled unless explicitly optional or not defined for certain variants.

-  The **step 2** can be done 1 to maximum of Network.
-  The **step 3** can be done 1 to maximum of Lsap.
-  The **step 4** can be done 1 to maximum of Route.
-  The **step 5** must be done 1 to maximum of Userpart.
-  The **step 6** must be done 1 to maximum of GTT Asso.
-  The **step 7** must be done 1 to maximum of GTT Addrmap.

7.2.1.1 General Configuration

The TB640_MSG_ID_SS7_SCCP_OP_ALLOC (request/response) message is used to initialize the general parameters of the SCCP layer.

The **request** part of the message TB640_MSG_ID_SS7_SCCP_OP_ALLOC contains the field:

```
...
TB640_SS7_HANDLE      hLayer; /* Contains the layer handle from system manager module */
TB640_SS7_SCCP_CFG    Cfg;    /* Contains the configuration of the SCCP layer */
...
```

Structure contains the general configuration parameters for SCCP layer:

```
typedef struct _TB640_SS7_SCCP_CFG
{
    TBX_UINT32                un32StructVersion;
    TBX_UINT32                un32DefStatusEnquiry;
    TBX_BOOL                  fUseTrafficLimit;
    TBX_BOOL                  fUseMsgImportance;
    TB640_SS7_SCCP_TRAFFIC_LIMITATION a3TrafficLim
        [TB640_SS7_SCCP_MAX_RL]
        [TB640_SS7_SCCP_MAX_RSL]
        [TB640_SS7_SCCP_MAX_RCL];


    TB640_SS7_SCCP_MSG_IMPORTANCE    MsgImportance;
    TBX_UINT32                un32NbISNIEntries;
    TBX_UINT32                un32NbINSEntries;
    TB640_SS7_SCCP_ISNI_MAP_CFG    aIsniNidMap
        [TB640_SS7_SCCP_MAX_NID_MAP];

    TB640_SS7_SCCP_INS_MAP_CFG    aInsNidMap
        [TB640_SS7_SCCP_MAX_NID_MAP];
} TB640_SS7_SCCP_CFG, *PTB640_SS7_SCCP_CFG;
```

The following enumeration lists the different configuration parameters and their description:

- The default status enquiry timer parameter (*un32DefStatusEnquiry* field in structure TB640_SS7_SCCP_CFG), specifies the time to send the periodic status request to MTP

Level 3 to enquire of the status of a point code. This parameter is reconfigurable. Typical value is 1000 milliseconds (1 second).

 The value 0 is to disable the timer.

- The traffic limitation flag parameter (*fUseTrafficLimit*). If this flag is set to TBX_TRUE, then the default traffic limitation is replaced by the new traffic limitation values defined in the array *a3TrafficLim[]*. This parameter is reconfigurable.

 Can be set to TBX_TRUE only when using **ITU96**.


- The message importance flag parameter (*fUseMsgImportance*). If this flag is set to TBX_TRUE, then the default message importance is replaced by the new message importance values defined in the structure *MsgImportance*. This parameter is reconfigurable.

 Can be set to TBX_TRUE only when using **ITU96**.


- The array of traffic limitation parameter (*a3TrafficLimitation*) specifies the values of the traffic limitation mechanism. As part of traffic limitation mechanism, on the event of remote SCCP congestion and/or underlying network congestion, SCCP discards a portion of traffic on the basis of importance of message and value of restriction level (RL) / restriction sub-level (RSL) for a destination. Each element of the table contains un8RL, un8RSL and un8RIL. RIL is restricted importance level, which is broadcasted to SCCP user. SCCP user fills the importance of message on the basis of received RIL. This parameter is reconfigurable. Allowable values:


Table 86 - SCCP Traffic Limitation

Traffic limitation fields	Description
un8RL	Restriction Level. It can assume value in the range 0 to 7.
un8RSL	Restriction Sub-Level. It can assume value in the range 0 to 3.
un8RIL	Restricted Importance Level. It can assume value in the range 0 to 7. RIL is passed to user at SPT interface so as user can fill proper value for parameter importance in subsequent messages to avoid message being discarded due to traffic limitation mechanism.

 Allowed for **ITU96** only.

- The message importance parameter (*MsgImportance*) specifies the values of the message importance. These values of message importance reconfiguration are used in traffic limitation mechanism. This parameter is reconfigurable.

 Message importance configuration is optional. If the message importance configuration is not done, then SCCP assumes the default values of message importance as per Q714, ITU-T 1996. If message importance is configured then the default values are overwritten with those of configured values.

 Allowed for **ITU96** only.

- The number of ISNI entries parameter (*un32NbISNIEntries*), specifies the number of ISNI (Intermediate Signalling Network Identification) entries in the array *aIsniNidMap[]*. This parameter is reconfigurable.
 - ✍ Allowed for **ANSI96** and **TELCORDIA** only.
- The number of INS entries parameter (*un32NbINSEntries*), specifies the number of INS (Intermediate Network Selections) entries in the array *aInsNidMap[]*. This parameter is reconfigurable.
 - ✍ Allowed for **TELCORDIA** only.
- The array of ISNI mapping entries parameter (*aIsniNidMap*) specifies the values of the SS7 format NID to PC configuration. NID to PC configuration serves as a lookup table to select a destination point code corresponding to a network ID. This parameter is reconfigurable.
- The array of INS mapping entries parameter (*aInsNidMap*) specifies the values of the Network specific NID to SS7-format NID mapping configuration. This configuration allows to use a NID in network specific format in INS parameter and serves as a lookup table to select an SS7 format NID corresponding to a network specific NID present in INS parameter. The SS7 format NID selected from this database is used to find a destination point code from the table containing SS7-NID to PC mapping. This parameter is reconfigurable.

7.2.1.2 Network Configuration

The TB640_MSG_ID_SS7_SCCP_OP_NETWORK_ALLOC (request/response) message is used to initialize the network parameters of the SCCP layer.

The **request** part of the message TB640_MSG_ID_SS7_SCCP_OP_NETWORK_ALLOC contains the field:

```

...
TB640_SS7_SCCP_HANDLE          hSccp; /* Handle to the SCCP general layer */
TB640_SS7_SCCP_NETWORK_CFG    Cfg;    /* Contains the configuration of the SCCP
                                         * Network instance */
...

```

Structure contains the network configuration parameters for SCCP layer:

```

typedef struct _TB640_SS7_SCCP_NETWORK_CFG
{
    TBX_UINT32                un32StructVersion;
    TBX_UINT32                un32NetworkId;
    TB640_SS7_SCCP_PROTOCOL_VARIANT ProtocolVariant;
    TB640_SS7_POINT_CODE      SelfPc;
    TB640_SS7_SUBSERVICE_FIELD_TYPE SsfType;
    TBX_UINT32                un32DefaultHopCount;
    TB640_SS7_NETWORK_INDICATOR_BIT_TYPE NetworkBit;
    TBX_BOOL                  fUseMappingOfSioPrioImp;
    TBX_UINT8                 aun8SioPrioImp [4];
    TBX_UINT32                un32SSTTimer;
    TBX_UINT32                un32SRTTimer;
}

```

```

TBX_UINT32          un32IgnoreSSTimer;
TBX_UINT32          un32CRDTimer;
TBX_UINT32          un32AsmbTimer;
TBX_UINT32          un32AttackTimer;
TBX_UINT32          un32DecayTimer;
TBX_UINT32          un32CongestionTimer;

} TB640_SS7_SCCP_NETWORK_CFG, *PTB640_SS7_SCCP_NETWORK_CFG;

```

The following enumeration lists the different configuration parameters and their description:

- The network identifier parameter (*un32NetworkId*) specifies the ID of the network. This parameter is not reconfigurable.
- The protocol variant parameter (*ProtocolVariant*) specifies the SCCP protocol variant required to provide service to the userpart connected through this upper SAP (Service Access Point). This parameter is not reconfigurable. Allowable values:

Table 87 - SCCP protocol variant

Protocol Variant	Description
TB640_SS7_SCCP_PROTOCOL_VARIANT_ANSI	Used for ANSI.
TB640_SS7_SCCP_PROTOCOL_VARIANT_ITU	Used for ITU.
TB640_SS7_SCCP_PROTOCOL_VARIANT_CHINA	Used for CHINA.

- The self point code parameter (*SelfPc*) contains the point code (address) of the local SS7 node to which the SCCP network is bounded to. This parameter is not reconfigurable. Proper point code length is: see **Table 3 - DPC Length**.
- The sub-service field (*SsfType*) specifies the type of network. This parameter is not reconfigurable. Allowable values: see **Table 21 - SSF values**.
- The default hop count parameter (*un32DefaultHopCount*) specifies the default hop count value. This parameter is reconfigurable. Allowable values are between 1 to 15.
- The network bit parameter (*NetworkBit*) specifies the network indicator bit. Used in the SCCP address (example when sending management messages). This parameter is not reconfigurable. Allowable values:

Table 88 - SCCP network indicator bit

Network Indicator Bit	Description
TB640_SS7_NETWORK_INDICATOR_BIT_TYPE_INTERNATIONAL	International indication.
TB640_SS7_NETWORK_INDICATOR_BIT_TYPE_NATIONAL	National indication.

- The use mapping of SIO priority importance flag parameter (*fUseMappingOfSioPrioImp*). Flag indicating presence of SIO priority to importance mapping. SIO priority to importance mapping should be configured only if SIO priority is carrying message importance as a national option. This parameter is reconfigurable. Allowable values: **TBX_TRUE** or **TBX_FALSE**.
- The array of SIO priority importance parameter (*aun8SioPrioImp*) specifies the mapping of SIO priority to importance of message value. This parameter is reconfigurable.

- The network timers configuration parameters. All timers are reconfigurable.

The timers have the following definitions:

<i>un32SSTTimer:</i>	Default subsystem status test timer. Time between sending SST messages to unavailable remote subsystem. This is the same as timer T(stat.info) in T1.112.4 and Q.714. Typical value is 30000 milliseconds (30 seconds) for ANSI and 5000 milliseconds (5 seconds) to 1200000 milliseconds (20 minutes) for ITU.
<i>un32SRRTimer:</i>	Default subsystem routing test timer. Time between sending SRT messages to remote subsystem (ANSI and TELCORDIA). This is the same as timer T(rtg.stat.info) in T1.112.4. Typical value is 30000 milliseconds (30 seconds).
<i>un32IgnoreSSTTimer:</i>	Default ignore SST timer. Time to ignore SST messages after receiving a subsystem Out-of-Service Grant (SOG) message. This is the same as timer T(ignore.SST) in T1.112.4 and Q.714. Typical value is 30000 milliseconds (30 seconds) for ANSI and selected by management for ITU.
<i>un32CRDTimer:</i>	Default coordinated state change timer. Time to wait for subsystem SOG message. This is the same as timer T(coord.chg) in T1.112.4 and Q.714. Typical value is 30000 milliseconds (30 seconds) for ANSI and 60000 milliseconds (1 minute) to 120000 milliseconds (2 minutes) for ITU.
<i>un32AsmbTimer:</i>	Default extended unit data reassembly timer. This is the same as timer T(reassembly) in Q.714. Typical value is 10000 milliseconds (10 seconds) to 20000 milliseconds (20 seconds).
<i>un32AttackTimer:</i>	Default attack timer (ITU96). This is the same as timer Ta in Q.714. Typical value is 100 milliseconds to 600 milliseconds.
<i>un32DecayTimer:</i>	Default decay timer (ITU96). This is the same as timer Td in Q.714. Typical value is 1000 milliseconds (1 second) to 10000 milliseconds (10 seconds).
<i>un32CongestionTimer:</i>	Default SCCP congestion timer (ITU96). This is the same as timer Tcon in Q.714. Typical value is 1000 milliseconds (1 second) to 10000 milliseconds (10 seconds).

The **response** part of the message TB640_MSG_ID_SS7_SCCP_OP_NETWORK_ALLOC contains the field:

```
...
TB640_SS7_SCCP_NETWORK_HANDLE          hSccpNetwork;          /* Handle of the instance of
* the SCCP network. */
...
```

7.2.1.3 Lsap Configuration

The TB640_MSG_ID_SS7_SCCP_OP_LSAP_ALLOC (request/response) message is used to initialize an instance of an SCCP Lsap (Lower service access point). Creating such entity tells the SCCP protocol layer about a collection of SS7 nodes that are accessible through MTP3.

The **request** part of the message TB640_MSG_SS7_SCCP_OP_LSAP_ALLOC contains the field:

```
...
TB640_SS7_SCCP_HANDLE                   hSccp; /* Handle of the SCCP layer */
TB640_SS7_SCCP_LSAP_CFG                  Cfg; /* Contains the configuration of a SCCP
Lsap instance */
...
```

This structure contains the configuration parameters for SCCP Lsap instance:

```
typedef struct _TB640_SS7_SCCP_LSAP_CFG
{
    TBX_UINT32                un32StructVersion;
    TB640_SS7_SCCP_NETWORK_HANDLE  hSccpNetwork;
    TB640_SS7_UID            UidSccpLsap;
    TB640_SS7_UID            UidMtp3Userpart;
    TBX_UINT32                un32MaxMsgLength;
    TBX_UINT8                 aun8Padding0[4];
} TB640_SS7_SCCP_LSAP_CFG, *PTB640_SS7_SCCP_LSAP_CFG;
```

The following enumeration lists the different configuration parameters and their description:

- The handle SCCP network parameter (*hSccpNetwork*) specifies the handle of the SCCP network. This parameter is not reconfigurable.
- The unique identifier SCCP Lsap parameter (*UidSccpLsap*) specifies the unique ID of the SCCP Lsap for a system SS7. This parameter is not reconfigurable.
- The unique identifier MTP3 userpart parameter (*UidMtp3Userpart*) specifies the unique ID of the MTP3 userpart to which this Lsap is connected. This parameter is not reconfigurable.
- The maximum message length parameter (*un32MaxMsgLength*) specifies the maximum length of the message delivered to SCCP from this Lsap. This parameter is reconfigurable.

The **response** part of the message TB640_MSG_SS7_SCCP_OP_LSAP_ALLOC contains the field:

```
...
TB640_SS7_SCCP_LSAP_HANDLE      hSccpLsap;    /* Handle of the instance of the
                                           SCCP Lsap */
...
```

7.2.1.4 Route Configuration

The TB640_MSG_ID_SS7_SCCP_OP_ROUTE_ALLOC (request/response) message is used to create a route to a specific DPC.

The **request** part of the message TB640_MSG_ID_SS7_SCCP_OP_ROUTE_ALLOC contains the field:

```
...
TB640_SS7_SCCP_HANDLE          hSccp;    /* Handle to the SCCP layer */
TB640_SS7_SCCP_ROUTE_CFG      Cfg;      /* Contains the configuration of the SCCP route
                                           instance */
...
```

This structure contains the configuration parameters for SCCP Route instance:

```
typedef struct _TB640_SS7_SCCP_ROUTE_CFG
{
```

```

TBX_UINT32                un32StructVersion;
TB640_SS7_SCCP_LSAP_HANDLE hSccpLsap;
TB640_SS7_SCCP_ROUTE_PROTOCOL_VARIANT ProtocolVariant;
TB640_SS7_SCCP_ROUTE_REPLICATED_MODE   ReplicatedMode;

TB640_SS7_POINT_CODE      Dpc;

TBX_UINT32                Options;
TBX_BOOL                   fBroadbandSupport;

TB640_SS7_SCCP_ROUTE_SLS_MASK SlsMask;
TBX_UINT32                un32NbBpc;
TB640_SS7_POINT_CODE      aBpc [TB640_SS7_SCCP_MAX_BPC];
TB640_SS7_SCCP_MSG_PRIORITY aPrioBpc [TB640_SS7_SCCP_MAX_BPC];

TBX_UINT8                 aun8Padding2[4];
TBX_UINT32                un32NbSSN;
TB640_SS7_SCCP_SSN_CFG    aSsn [TB640_SS7_SCCP_MAX_SSN_PER_DPC];

} TB640_SS7_SCCP_ROUTE_CFG, *PTB640_SS7_SCCP_ROUTE_CFG;

```

The following enumeration lists the different configuration parameters and their description:

- The handle SCCP Lsap parameter (*hSccpLsap*) specifies the handle of the SCCP Lsap. This parameter is not reconfigurable.
- The protocol variant parameter (*ProtocolVariant*) specifies the SCCP route protocol variant for this destination point code. This parameter is not reconfigurable. Allowable values:

Table 89 - SCCP route protocol variant

Route Protocol Variant	Description
TB640_SS7_SCCP_ROUTE_PROTOCOL_VARIANT_ANSI88	Used for ANSI88.
TB640_SS7_SCCP_ROUTE_PROTOCOL_VARIANT_ANSI92	Used for ANSI92.
TB640_SS7_SCCP_ROUTE_PROTOCOL_VARIANT_ANSI96	Used for ANSI96.
TB640_SS7_SCCP_ROUTE_PROTOCOL_VARIANT_TELCORDIA	Used for TELCORDIA.
TB640_SS7_SCCP_ROUTE_PROTOCOL_VARIANT_ITU88	Used for ITU88.
TB640_SS7_SCCP_ROUTE_PROTOCOL_VARIANT_ITU92	Used for ITU92.
TB640_SS7_SCCP_ROUTE_PROTOCOL_VARIANT_ITU96	Used for ITU96.
TB640_SS7_SCCP_ROUTE_PROTOCOL_VARIANT_GSM0806	Valid in both ITU and ANSI networks.

- The replicate mode parameter (*ReplicateMode*) specifies the mode of replicated nodes/subsystems. This field is reconfigurable. Allowable values:


Table 90 - SCCP replicate mode

Replicate Mode	Description
TB640_SS7_SCCP_ROUTE_REPLICATED_MODE_DOMINANT	In this mode, each backup node/subsystem is assigned a unique priority. The node/subsystem on this route is the primary node/subsystem and is selected as the preferred node/subsystem when accessible. On failure of the preferred node/subsystem the next higher priority available backup node/subsystem is selected as the

	preferred node/subsystem.
TB640_SS7_SCCP_ROUTE_REPLICATED_MODE_LOADSHARE	In this mode, each backup node/subsystem together with the node/subsystem configured on the route is assigned equal priority and load is distributed among nodes/subsystems on the basis of: 1) SLS for class 1 traffic, and 2) in round-robin fashion for class 0 traffic. Loadshare mode is applicable in TELCORDIA only.
TB640_SS7_SCCP_ROUTE_REPLICATED_MODE_DOMINANT_ALTERNATE	This mode is significant only for class 0 traffic. In this mode, class 0 traffic is prevented from reaching an accessible but congested node. For class 1 traffic, this mode is same as DOMINANT mode. In this mode, each backup node/subsystem is assigned a unique priority. The node/subsystem configured on the route is selected as the primary node/subsystem and is the preferred node/subsystem when accessible and not congested. On failure or congestion of the preferred node/subsystem the next higher priority available and uncongested backup node/subsystem is selected as the preferred node/subsystem. This mode of operation is applicable only in ANSI96 and TELCORDIA.
TB640_SS7_SCCP_ROUTE_REPLICATED_MODE_LOADSHARE_ALTERNATE	This mode is significant only for class 0 traffic. In this mode, class 0 traffic is prevented from reaching an accessible, but congested node. For class 1 traffic, this mode is same as LOADSHARE mode. In this mode, each backup node/subsystem together with the node/subsystem configured on the route is assigned equal priority and load is distributed among nodes/subsystems on the basis of: 1) SLS for class 1 traffic, and 2) in round-robin fashion for class 0 traffic. For class 0 traffic, a node/subsystem is selected when it is accessible and uncongested. On failure or congestion of the node, next available and uncongested node/subsystem is selected. This mode of operation is applicable only in TELCORDIA.

 Only used with ANSI96 and TLECORDIA.


- The destination point code parameter (*Dpc*) specifies the point code of the node where the route is terminating. This field is not reconfigurable. Proper point code length is: see **Table 3 - DPC Length**.

 **Default routing:** Note that destination point code 0.0.0 is reserved for default routing. When no explicit route exists for a destination point code and default route 0.0.0 is present, the message is passed to MTP 3 for delivery in place of being dropped. If the MTP 3 layer is unable to deliver the message for any reason, the message is dropped and no notification is given to the application that originated the message.


- The options parameter (*Options*) specifies the route options. This field is reconfigurable. Allowable values:

Table 91 - SCCP route options

Route Options	Description
TB640_SS7_SCCP_ROUTE_OPTIONS_NONE	None (no options).
TB640_SS7_SCCP_ROUTE_OPTIONS_ONLINE_BY_DEFAULT	Node is online by default.
TB640_SS7_SCCP_ROUTE_OPTIONS_TRANSLATOR_NODE	Translator node.
TB640_SS7_SCCP_ROUTE_OPTIONS_ADJACENT_NODE	Adjacent node.
TB640_SS7_SCCP_ROUTE_OPTIONS_INS_CAPABLE_NODE	INS capable. Only in TELCORDIA.

 These values can be ORed together.


- The broadband support flag parameter (*fBroadbandSupport*). Flag to indicate whether the peer SCCP on the route has broadband support. This parameter is reconfigurable. Allowable values: **TBX_TRUE** or **TBX_FALSE**.

 For ITU88, ITU92, ANSI88, ANSI92 and ANSI96 network this flag should always be **TBX_FALSE**.


- The signaling link selector mask parameter (*SlsMask*) specifies, in ANSI and TELCORDIA networks, if the SLS can be either 5 bits or 8 bits. This field is reconfigurable. Allowable values:

Table 92 - SCCP route SLS mask

Route SLS Mask	Description
TB640_SS7_SCCP_ROUTE_SLS_MASK_5BITS	Networks SLS is 5 bits.
TB640_SS7_SCCP_ROUTE_SLS_MASK_8BITS	Networks SLS is 8 bits.

 For ANSI and TELCORDIA only.

- The number backup point code parameter (*un32NbBpc*) specifies the number of backup point code entries in the arrays *aBpc[]* and *aPrioBpc[]*. This field is reconfigurable.
- The array of backup point code entries parameter (*aBpc*) specifies the list of backup point code for this route. This parameter is reconfigurable.
- The array of priority of backup point code entries parameter (*aPrioBpc*) specifies the list of priority of each backup point code for this route. This parameter is reconfigurable. Allowable values are from **0** to **255**.

 Zero being the lowest priority.

- The number subsystem number parameter (*un32NbSSN*) specifies the number of subsystem number entries in the array *aSsn[J]*. This field is reconfigurable.
- The array of subsystem number entries parameter (*aSsn*) specifies the list of subsystem number for this route. This parameter is reconfigurable.

Structure contains the SSN specific configuration parameters for route configuration:

```
typedef struct _TB640_SS7_SCCP_SSN_CFG
{
    TBX_UINT32                un32StructVersion;
    TBX_UINT8                 un8SSN;
    TBX_UINT8                 aun8Padding0[3];
    TBX_UINT32                SsnOptions;
    TB640_SS7_SCCP_ROUTE_REPLICATED_MODE    ReplicatedMode;

    TBX_UINT8                 aun8Padding1[4];
    TBX_UINT32                un32NbBpc;
    TB640_SS7_POINT_CODE     aBpc [TB640_SS7_SCCP_MAX_BPC];
    TBX_UINT32                aPrioBpc [TB640_SS7_SCCP_MAX_BPC];

    TBX_UINT8                 aun8Padding2[4];
    TBX_UINT32                un32NbConPc;
    TB640_SS7_POINT_CODE     aConPc
    [TB640_SS7_SCCP_MAX_CONCERNED_PC];
} TB640_SS7_SCCP_SSN_CFG, *PTB640_SS7_SCCP_SSN_CFG;
```

General explanation of the parameters of SSN specific configuration:

- The subsystem number parameter (*un8SSN*) specifies a subsystem number for this route. This field is reconfigurable. Allowable values are from **1** to **255**.

Table 93 - SCCP SSN value to identify SCCP user


SSN value to identify SCCP user	Description
TB640_SS7_SCCP_SSN_VALUE_UNKNOWN	SSN not known or not used.
TB640_SS7_SCCP_SSN_VALUE_SCCPMNGT	SCCP management.
TB640_SS7_SCCP_SSN_VALUE_ISUP	ISDN user part.
TB640_SS7_SCCP_SSN_VALUE_OMAP	Operation, Maintenance and Administration Part.
TB640_SS7_SCCP_SSN_VALUE_MAP	Mobile Application Part.
TB640_SS7_SCCP_SSN_VALUE_HLR	Home Location Register.
TB640_SS7_SCCP_SSN_VALUE_VLR	Visitor Location Register.
TB640_SS7_SCCP_SSN_VALUE_MSC	Mobile Switching Center.
TB640_SS7_SCCP_SSN_VALUE_EIR	Equipment Identifier Register.
TB640_SS7_SCCP_SSN_VALUE_AC	Authentication Center.
TB640_SS7_SCCP_SSN_VALUE_ISDNSUPP	ISDN Supplementary services.
TB640_SS7_SCCP_SSN_VALUE_BISDNE2EAPP	Broadband ISDN Edge-to-Edge Application.
TB640_SS7_SCCP_SSN_VALUE_TCTR	TC Test Responder.
TB640_SS7_SCCP_SSN_VALUE_RANAP	Radio Access Network Application Part.
TB640_SS7_SCCP_SSN_VALUE_RNSAP	Radio Network Subsystem Application Part.
TB640_SS7_SCCP_SSN_VALUE_GMLC	Gateway Mobile Location Centre (MAP).
TB640_SS7_SCCP_SSN_VALUE_CAP	CAMEL application part.
TB640_SS7_SCCP_SSN_VALUE_GSMSCF	Global System for Mobile Service Control Function (MAP).
TB640_SS7_SCCP_SSN_VALUE_SIWF	Shared Interworking Functions (MAP).

TB640_SS7_SCCP_SSN_VALUE_SGSN	Serving GPRS Support Node (MAP).
TB640_SS7_SCCP_SSN_VALUE_GGSN	Gateway GPRS Support Node (MAP).
TB640_SS7_SCCP_SSN_VALUE_BSC_PCAP	Positioning Calculation Application Part.
TB640_SS7_SCCP_SSN_VALUE_BSC_BSSAP	BSC Base Station System Application Part (BSSAP-LE).
TB640_SS7_SCCP_SSN_VALUE_MSC_BSSAP	MSC Base Station System Application Part (BSSAP-LE).
TB640_SS7_SCCP_SSN_VALUE_SMLC	Serving Mobile Location Center (BSSAP-LE).
TB640_SS7_SCCP_SSN_VALUE_BSSOM	Base Station System Operation & Maintenance (A-interface).
TB640_SS7_SCCP_SSN_VALUE_BSSAP	Base Station System Application Part (A-interface).


- The subsystem number options parameter (*SsnOptions*) specifies the subsystem number options. This field is reconfigurable. Allowable values:

Table 94 - SCCP SSN options

Route Options	Description
TB640_SS7_SCCP_SSN_OPTIONS_ACCESSIBLE_BY_DEFAULT	Subsystem is accessible by default.
TB640_SS7_SCCP_SSN_OPTIONS_NORMAL_ROUTED	Subsystem normal routed - valid in ANSI and TELCORDIA network only.

 These values can be ORed together.

- The replicate mode parameter (*ReplicateMode*) specifies the mode of replicated nodes/subsystems. This field is reconfigurable. Allowable values: see **Table 90 - SCCP replicate mode**.
- The number backup point code parameter (*un32NbBpc*) specifies the number of backup point code entries in the arrays *aBpc[]* and *aPrioBpc[]*. This field is reconfigurable.
- The array of backup point code entries parameter (*aBpc*) specifies the list of backup point code for this SSN. This parameter is reconfigurable.
- The array of priority of backup point code entries parameter (*aPrioBpc*) specifies the list of priority of each backup point code for this SSN. This parameter is reconfigurable. Allowable values are from **0** to **255**.

 Zero being the lowest priority.

- The number concerned point code parameter (*un32NbConPc*) specifies the number of concerned point code entries in the array *aConPc[]*. This field is reconfigurable.
- The array of concerned point code entries parameter (*aConPc*) specifies the list of concerned point code for this SSN. This parameter is reconfigurable.

The **response** part of the message TB640_MSG_ID_SS7_SCCP_OP_ROUTE_ALLOC contains the field:

```

...
TB640_SS7_SCCP_ROUTE_HANDLE          hSccpRoute;      /* The handle of the instance of
...                                     the SCCP route. */

```

7.2.1.5 Userpart Configuration

The TB640_MSG_ID_SS7_SCCP_OP_USERPART_ALLOC (request/response) message is used to initialize the userpart parameters of the SCCP layer.

The **request** part of the message TB640_MSG_ID_SS7_SCCP_OP_USERPART_ALLOC contains the field:

```

...
TB640_SS7_SCCP_HANDLE      hSccp; /* Handle to the SCCP layer */
TB640_SS7_SCCP_USERPART_CFG Cfg; /* Contains the configuration of the SCCP
                                   * Userpart instance */
...

```

This structure contains the configuration parameters for SCCP Userpart instance:

```

typedef struct _TB640_SS7_SCCP_USERPART_CFG
{
    TBX_UINT32          un32StructVersion;
    TB640_SS7_SCCP_NETWORK_HANDLE hSccpNetwork;
    TB640_SS7_UID      UidSccpUserpart;
    TB640_SS7_SCCP_MSG_PRIORITY  MsgPriority;

    TB640_SS7_SCCP_CONNECTION_MODE ConnectionMode;
    TBX_UINT32          un32SSN;

    TBX_UINT8          aun8Padding0[4];
    TBX_UINT32          un32NbBpc;
    TB640_SS7_POINT_CODE aBpc [TB640_SS7_SCCP_MAX_BPC];
    TBX_UINT32          aPrioBpc [TB640_SS7_SCCP_MAX_BPC];

    TBX_UINT8          aun8Padding1[4];
    TBX_UINT32          un32NbConPC;
    TB640_SS7_POINT_CODE aConPc [TB640_SS7_SCCP_MAX_CONCERNED_PC];
} TB640_SS7_SCCP_USERPART_CFG, *PTB640_SS7_SCCP_USERPART_CFG;

```

The following enumeration lists the different configuration parameters and their description:

- The handle SCCP network parameter (*hSccpNetwork*) specifies the handle of the SCCP network. This parameter is not reconfigurable.
- The unique identifier SCCP Userpart parameter (*UidSccpUserpart*) specifies the unique ID of the SCCP Userpart for a system SS7. This parameter is not reconfigurable.
- The message priority parameter (*MsgPriority*) specifies the priority of messages This field is not reconfigurable. Allowable values:


Table 95 - SCCP message priority

Message Priority	Description
TB640_SS7_SCCP_MSG_PRIORITY_HIGHEST	
TB640_SS7_SCCP_MSG_PRIORITY_0	Same as highest
TB640_SS7_SCCP_MSG_PRIORITY_1	
TB640_SS7_SCCP_MSG_PRIORITY_2	
TB640_SS7_SCCP_MSG_PRIORITY_3	Same as lowest
TB640_SS7_SCCP_MSG_PRIORITY_LOWEST	

- The connection mode parameter (*ConnectionMode*) specifies the different modes of connection for a SCCP userpart. This field is not reconfigurable. Allowable values:

Table 96 - SCCP connection mode

Connection Mode	Description
TB640_SS7_SCCP_CONNECTION_MODE_TCAP	SCCP userpart is used in conjunction with an above local TCAP layer and cannot be controlled by the host application but the host application will receive ALARM notifications.
TB640_SS7_SCCP_CONNECTION_MODE_HOST_TCAP	SCCP userpart is used as a standalone and communicates with remote TCAP layer (host application).

- The subsystem number parameter (*un32SSN*) specifies a subsystem number for this userpart. This field is not reconfigurable.. Allowable values are from 1 to 255. See **Table 93 - SCCP SSN value to identify SCCP user.**
- The number backup point code parameter (*un32NbBpc*) specifies the number of backup point code entries in the arrays *aBpc[]* and *aPrioBpc[]*. This field is reconfigurable.
- The array of backup point code entries parameter (*aBpc*) specifies the list of backup point code for this Userpart. This parameter is reconfigurable.
- The array of priority of backup point code entries parameter (*aPrioBpc*) specifies the list of priority of each backup point code for this Userpart. This parameter is reconfigurable. Allowable values are from **0** to **255**.
 -  Zero being the lowest priority.
- The number concerned point code parameter (*un32NbConPc*) specifies the number of concerned point code entries in the array *aConPc[]*. This field is reconfigurable.
- The array of concerned point code entries parameter (*aConPc*) specifies the list of concerned point code for this Userpart. This parameter is reconfigurable.

The **response** part of the message TB640_MSG_ID_SS7_SCCP_OP_USERPART_ALLOC contains the field:

```

...
TB640_SS7_SCCP_USERPART_HANDLE          hSccpUserpart;          /* The handle of the instance of
...                                     * the SCCP userpart. */

```

7.2.1.6 GTT Association Configuration

The TB640_MSG_ID_SS7_SCCP_OP_GTT ASSO_ALLOC (request/response) message is used to configure associations for Global Title Translation of the SCCP layer. An association specifies the translation procedure to be used for a particular type of global title. The user configures an association for each global title type in SCCP.

The **request** part of the message TB640_MSG_ID_SS7_SCCP_OP_GTT ASSO_ALLOC contains the field:

```

...
TB640_SS7_SCCP_HANDLE          hSccp; /* Handle to the SCCP layer */
TB640_SS7_SCCP_GTT ASSO_CFG    Cfg; /* Contains the configuration of the SCCP GTT
                                     association instance */
...

```

This structure contains the configuration parameters for SCCP GTT association instance:

```

typedef struct _TB640_SS7_SCCP_GTT ASSO_CFG
{
    TBX_UINT32                un32StructVersion;
    TBX_UINT32                un32NetworkId;
    TB640_SS7_SCCP_PROTOCOL_VARIANT ProtocolVariant;
    TBX_UINT8                 aun8Padding0[4];

    TB640_SS7_SCCP_GT_FORMAT_TYPE GtFormat;
    TBX_UINT32                un32NbActions;
    TB640_SS7_SCCP_ACTION      aActions [TB640_SS7_SCCP_MAX_GTT_ACTIONS];

    union
    {
        TB640_SS7_SCCP_GT_FORMAT1      Format1;
        TB640_SS7_SCCP_GT_FORMAT2      Format2;
        TB640_SS7_SCCP_GT_FORMAT3      Format3;
        TB640_SS7_SCCP_GT_FORMAT4      Format4;
    };
} TB640_SS7_SCCP_GTT ASSO_CFG, *PTB640_SS7_SCCP_GTT ASSO_CFG;

```

The following enumeration lists the different configuration parameters and their description:

- The network identifier parameter (*un32NetworkId*) specifies the ID of the network.
- The protocol variant parameter (*ProtocolVariant*) specifies the SCCP protocol variant. Allowable values: see **Table 87 - SCCP protocol variant**.
- The global title format parameter (*GtFormat*) specifies the SCCP GT format. Allowable values:

Table 97 - SCCP GT format type

Global Title Format Type	Description
TB640_SS7_SCCP_GT_FORMAT_TYPE_NOT_USED	Indicates that no global title is used.
TB640_SS7_SCCP_GT_FORMAT_TYPE_0	Indicates that no global title is used.
TB640_SS7_SCCP_GT_FORMAT_TYPE_1	Global title format 1.
TB640_SS7_SCCP_GT_FORMAT_TYPE_2	Global title format 2.
TB640_SS7_SCCP_GT_FORMAT_TYPE_3	Global title format 3.
TB640_SS7_SCCP_GT_FORMAT_TYPE_4	Global title format 4.

- The number of actions parameter (*un32NbActions*) specifies the number of actions configured for this association. Allowable values: **0 - 7**.
- The array of actions parameter (*aActions*) contains the actions for this association. Allowable values: see Actions structure definition below.

Structure contains the Action specific configuration parameters for GTT association configuration:

```
typedef struct _TB640_SS7_SCCP_ACTION
{
    TBX_UINT32                un32StructVersion;
    TB640_SS7_SCCP_ACTION_TYPE ActionType;
    TBX_UINT32                un32StartDigit;
    TBX_UINT32                un32EndDigit;
} TB640_SS7_SCCP_ACTION, *PTB640_SS7_SCCP_ACTION;
```



General explanation of the parameters of Action specific configuration:

- The action type parameter (*ActionType*) specifies the type of action for this association. Allowable values:

Table 98 - SCCP GT action type

SSN value to identify SCCP user	Description
TB640_SS7_SCCP_ACTION_TYPE_FIX	Use a fixed range of digits to perform GTT. For example, use first 3 digits.
TB640_SS7_SCCP_ACTION_TYPE_VARIABLE_ASCENDING	Use variable number of digits GTT in ascending order. For example, use digits 4 to 6 in a variable fashion to perform GTT. In this case, variable means match the fourth digit, and if no match, then match the fourth and fifth digits, and so on.
TB640_SS7_SCCP_ACTION_TYPE_VARIABLE_DESCENDING	Use variable number of digits GTT in descending order. For example, use digits 4 to 6 in a variable fashion to perform GTT. In this case, variable means match the sixth digit, and if no match, then match the fifth and sixth digits, and so on.
TB640_SS7_SCCP_ACTION_TYPE_GT_TO_PC	Use digits in incoming GT as the point code for routing purposes. This type of GTT is used in ISNI.
TB640_SS7_SCCP_ACTION_TYPE_CONST	Always translate the incoming GT to a fixed address.
TB640_SS7_SCCP_ACTION_TYPE_INSERT_PC	Insert DPC before incoming GTAI and change TT, ES and NAI, leaving GTAI unchanged. This action is for modifying calling party address at an international Gateway, when message is entering international network, originating from a national network and having calling address in national specific format with Generic Numbering Plan. DPC inserted is the point code of the international Gateway, having capability to translate back the international address format to national specific address in response message. The DPC inserted, need not be the self point code of the international

	Gateway modifying the calling party address.
TB640_SS7_SCCP_ACTION_TYPE_STRIP_PC	If first six digits are self point code in international format, strip off first six digits from the GTAI. Then do the translation and replace TT, ES, and NAI. Examine the DPC of the translated address, if it is self, do translation of the resulting address again else message will be forwarded to DPC for further translation without changing GTAI of the called address.

- The start digit parameter (*un32StartDigit*) specifies the starting digit of the range for this action.
 -  Used only with action of type:
 - TB640_SS7_SCCP_ACTION_TYPE_FIX
 - TB640_SS7_SCCP_ACTION_TYPE_VARIABLE_ASCENDING
 - TB640_SS7_SCCP_ACTION_TYPE_VARIABLE_DESCENDING actions
- The end digit parameter (*un32EndDigit*) specifies the ending digit of the range for this action.
 -  Used only with action of type:
 - TB640_SS7_SCCP_ACTION_TYPE_FIX
 - TB640_SS7_SCCP_ACTION_TYPE_VARIABLE_ASCENDING
 - TB640_SS7_SCCP_ACTION_TYPE_VARIABLE_DESCENDING actions

Structure contains the GT Format 1 specific configuration parameters for GTT association configuration:

```
typedef struct _TB640_SS7_SCCP_GT_FORMAT1
{
    TB640_SS7_SCCP_GT_ODD_EVEN_TYPE          OddEven;
    TB640_SS7_SCCP_GT_NATURE_OF_ADDR_IND    NatureOfAddress;
} TB640_SS7_SCCP_GT_FORMAT1, *PTB640_SS7_SCCP_GT_FORMAT1;
```

General explanation of the parameters of GT Format 1 specific information:

- The odd even parameter (*OddEven*) specifies the odd or even indicator. Used to encode the odd or even indicator bit in the address parameter of an SCCP message. Allowable values:

Table 99 - SCCP GT odd/even type

GT Odd/Even Type	Description
TB640_SS7_SCCP_GT_ODD_EVEN_TYPE_NOT_USED	Not used.
TB640_SS7_SCCP_GT_ODD_EVEN_TYPE_ODD	Odd number of address signals.
TB640_SS7_SCCP_GT_ODD_EVEN_TYPE_EVEN	Even number of address signals.

- The nature of address parameter (*NatureOfAddress*) specifies the nature of address indicator. Used to encode the 7-bit nature of address indicator in the address parameter of an SCCP message. Allowable values:

Table 100 - SCCP GT nature of address indicator

GT Nature of Address Indicator	Description
TB640_SS7_SCCP_GT_NATURE_OF_ADDR_IND_NOT_USED	Not used.
TB640_SS7_SCCP_GT_NATURE_OF_ADDR_IND_SUBSCRIBER	Subscriber number.
TB640_SS7_SCCP_GT_NATURE_OF_ADDR_IND_NATIONAL	National significant number.
TB640_SS7_SCCP_GT_NATURE_OF_ADDR_IND_INTERNATIONAL	International number.

Structure contains the GT Format 2 specific information parameters for GTT association configuration:

```
typedef struct _TB640_SS7_SCCP_GT_FORMAT2
{
    TBX_BOOLEAN                fUseTypeValue;
    TBX_UINT8                  aun8Padding0[3];
    TBX_UINT32                 un32TypeValue;
} TB640_SS7_SCCP_GT_FORMAT2, *PTB640_SS7_SCCP_GT_FORMAT2;
```

General explanation of the parameters of GT Format 2 specific information:

- The use type value parameter (*fUseTypeValue*) specifies whether or not to use the translation type value. Note that the translation type value is required for GTT address map configuration and also to send data using GT but it is optional for GTT association configuration. When setting the use translation type value parameter (*fUseTypeValue*) to TBX_FALSE then type value (*un32TypeValue*) is ignored (don't care) for GTT association operation.
- The type value parameter (*un32TypeValue*) specifies the translation type value. Used to encode the 8-bit translation type field in the address parameter of an SCCP message. Allowable values: **0** to **255**.

Structure contains the GT Format 3 specific information parameters for GTT association configuration:

```
typedef struct _TB640_SS7_SCCP_GT_FORMAT3
{
    TBX_BOOLEAN                fUseTypeValue;
    TBX_UINT8                  aun8Padding0[3];
    TBX_UINT32                 un32TypeValue;
    TB640_SS7_SCCP_GT_NUMB_PLAN_TYPE    NumberingPlan;
    TB640_SS7_SCCP_GT_ENCODING_TYPE    EncodingScheme;
} TB640_SS7_SCCP_GT_FORMAT3, *PTB640_SS7_SCCP_GT_FORMAT3;
```

General explanation of the parameters of GT Format 3 specific information:

- The use type value parameter (*fUseTypeValue*)... see GT Format 2 explanation.

- The type value parameter (*un32TypeValue*)... see GT Format 2 explanation.
- The numbering plan parameter (*NumberingPlan*) specifies the numbering plan. Used to encode the numbering plan nibble in the address parameter of an SCCP message. Allowable values:

Table 101 - SCCP GT numbering plan type

GT Numbering Plan Type	Description
TB640_SS7_SCCP_GT_NUMB_PLAN_TYPE_NOT_USED	Not used.
TB640_SS7_SCCP_GT_NUMB_PLAN_TYPE_UNKNOWN	Unknown.
TB640_SS7_SCCP_GT_NUMB_PLAN_TYPE_ISDN	ISDN/telephony numbering plan.
TB640_SS7_SCCP_GT_NUMB_PLAN_TYPE_TELEPHONY	Telephony numbering plan. (Not valid in ITU96, ANSI96 and TELCORDIA)
TB640_SS7_SCCP_GT_NUMB_PLAN_TYPE_GENERIC	Generic numbering plan. (For ITU96 only)
TB640_SS7_SCCP_GT_NUMB_PLAN_TYPE_DATA	Data numbering plan.
TB640_SS7_SCCP_GT_NUMB_PLAN_TYPE_TELEX	Telex numbering plan.
TB640_SS7_SCCP_GT_NUMB_PLAN_TYPE_MARITIME_MOBILE	Maritime Mobile numbering plan.
TB640_SS7_SCCP_GT_NUMB_PLAN_TYPE_LAND_MOBILE	Land Mobile numbering plan.
TB640_SS7_SCCP_GT_NUMB_PLAN_TYPE_ISDN_MOBILE	ISDN Mobile numbering plan.
TB640_SS7_SCCP_GT_NUMB_PLAN_TYPE_PRIVATE	Private network or network specific numbering plan. (For ITU96, ANSI96 and TELCORDIA)

- The encoding scheme parameter (*EncodingScheme*) specifies the encoding scheme. Used to encode the encoding scheme nibble in the address parameter of an SCCP message (supports only BCD encoding scheme). Allowable values:

Table 102 - SCCP GT encoding scheme type

GT Encoding Scheme Type	Description
TB640_SS7_SCCP_GT_ENCODING_TYPE_NOT_USED	Not used.
TB640_SS7_SCCP_GT_ENCODING_TYPE_UNKNOWN	Unknown.
TB640_SS7_SCCP_GT_ENCODING_TYPE_BCD_ODD	BCD Odd.
TB640_SS7_SCCP_GT_ENCODING_TYPE_BCD_EVEN	BCD Even.
TB640_SS7_SCCP_GT_ENCODING_TYPE_NATIONAL_ES	National specific ES. (Allowed only in ITU96)

Structure contains the GT Format 4 specific information parameters for GTT association configuration:

```
typedef struct _TB640_SS7_SCCP_GT_FORMAT4
{
    TBX_BOOLEAN                fUseTypeValue;
    TBX_UINT8                  un8Padding0[3];
    TBX_UINT32                 un32TypeValue;
    TB640_SS7_SCCP_GT_NUMB_PLAN_TYPE    NumberingPlan;
    TB640_SS7_SCCP_GT_ENCODING_TYPE    EncodingScheme;
    TB640_SS7_SCCP_GT_NATURE_OF_ADDR_IND    NatureOfAddress;
    TBX_UINT8                  un8Padding1[4];
} TB640_SS7_SCCP_GT_FORMAT4, *PTB640_SS7_SCCP_GT_FORMAT4;
```

General explanation of the parameters of GT Format 4 specific information:

- The use type value parameter (*fUseTypeValue*)... see GT Format 2 explanation.
- The type value parameter (*un32TypeValue*)... see GT Format 2 explanation.
- The numbering plan parameter (*NumberingPlan*)... see GT Format 3 explanation.
- The encoding scheme parameter (*EncodingScheme*) ... see GT Format 3 explanation.
- The nature of address parameter (*NatureOfAddress*) ... see GT Format 1 explanation.

The **response** part of the message TB640_MSG_ID_SS7_SCCP_OP_GTT ASSO_ALLOC contains the field:

```

...
TB640_SS7_SCCP_GTT ASSO_HANDLE      hSccpGttAsso;          /* The handle of the instance of
                               * the SCCP GTT association. */
...

```

7.2.1.7 GTT Address Map Configuration

The TB640_MSG_ID_SS7_SCCP_OP_GTT_ADDRMAP_ALLOC (request/response) message is used to configure an address map instance for a GTT association of the SCCP layer. These address map entries specify the results of translation. Following the association configuration, the user has to configure address map entries in SCCP.

The **request** part of the message TB640_MSG_ID_SS7_SCCP_OP_GTT_ADDRMAP_ALLOC contains the field:

```

...
TB640_SS7_SCCP_HANDLE      hSccp;
TB640_SS7_SCCP_GTT_ADDRMAP_CFG  Cfg; /* Contains the configuration of a SCCP
                               Address map instance */
...

```

This structure contains the configuration parameters for SCCP address map instance:

```

typedef struct _TB640_SS7_SCCP_GTT_ADDRMAP_CFG
{
    TBX_UINT32                un32StructVersion;
    TB640_SS7_SCCP_GTT ASSO_HANDLE  hSccpGttAsso;

    TB640_SS7_SCCP_ACTION      Action;

    TBX_BOOLEAN               fReplaceGt;
    TBX_BOOLEAN               fLoadShare;
    TBX_UINT8                 aun8Padding0[6];
    TB640_SS7_SCCP_GLOBAL_TITLE  Gt;

    TBX_UINT32                un32OutNetworkId;
    TBX_UINT8                 un8NbEntities;
    TBX_UINT8                 aun8Padding1[3];
    TB640_SS7_SCCP_ENTITY      aOutAddr [TB640_SS7_SCCP_MAX_ENTITY];
} TB640_SS7_SCCP_GTT_ADDRMAP_CFG, *PTB640_SS7_SCCP_GTT_ADDRMAP_CFG;

```

The following enumeration lists the different configuration parameters and their description:

- The handle SCCP gtt association parameter (*hSccpGttAsso*) specifies the handle of the SCCP global title translation association.
- The action parameter (*Action*) specifies the type of action for this address map. Allowable values: see **Table 98 - SCCP GT action type**.
 - ✍ The range parameters might not be the same as configured in the association. They can be a subset.
- The replace GT flag parameter (*fReplaceGt*) specifies whether to replace the GT in outgoing (post GTT) address with the incoming GT.
- The load share flag parameter (*fLoadShare*) specifies the mode of operation of SCCP entities. Mode can be either DOMINANT (if flag set to TBX_FALSE) or LOADSHARE (if flag set to TBX_TRUE). Loadsharing among SCCP entities will be achieved on the basis of SLS for class 1 traffic and on round-robin fashion for class 0 traffic.
 - ✍ This parameter is applicable **only in ITU96**.
- The global title parameter (*Gt*) specifies the global title. Apart from the global title address, the user needs to fill up fields used for finding the rule. See Global Title structure definition below.
- The out network identifier parameter (*un32OutNetworkId*) specifies the network id of outgoing addresses.
- The number of entities parameter (*un8NbEntities*) specifies the number of entities in the array *aOutAddr*. Allowable values: **1 - 2**.
 - ✍ In ITU96 it should be defined as 2 and in all other versions it should be defined as 1.
- The array of out address parameter (*aOutAddr*) specifies the SCCP entity outgoing address. See Entity structure definition below.

Structure contains the Global Title specific information parameters for the address map:

```
typedef struct _TB640_SS7_SCCP_GLOBAL_TITLE
{
    TB640_SS7_SCCP_GT_FORMAT_TYPE    GtFormat;
    TBX_UINT32                       un32AddressLength;
    TBX_UINT8                         aun8BCDAddress [TB640_SS7_SCCP_MAX_ADDRESS_LENGTH];

    union
    {
        TB640_SS7_SCCP_GT_FORMAT1    Format1;
        TB640_SS7_SCCP_GT_FORMAT2    Format2;
        TB640_SS7_SCCP_GT_FORMAT3    Format3;
        TB640_SS7_SCCP_GT_FORMAT4    Format4;
    };
};
```

```
} TB640_SS7_SCCP_GLOBAL_TITLE, *PTB640_SS7_SCCP_GLOBAL_TITLE;
```

General explanation of the parameters of Global Title specific information:

- The global title format parameter (*GtFormat*) specifies the type of the global title format. Allowable values: see **Table 97 - SCCP GT format type**.
- The address length parameter (*un32AddressLength*) specifies the number of bytes the BCD digits occupy.
- The BCD address parameter (*aun8BCDAddress*) specifies the BCD digits of SCCP address.
- The format 1, 2, 3 and 4 parameters (*Format1 Format2, Format3 and Format4*) ... see GT Format 1, 2, 3 and 4 explanation in section **7.2.1.6 GTT Association Configuration**.

Structure contains the Entity specific information parameters for the address map:

```
typedef struct _TB640_SS7_SCCP_ENTITY
{
    TBX_BOOLEAN                fPresent;
    TBX_BOOLEAN                fNoPointCodeInHeader;

    TBX_BOOLEAN                fSsfPresent;
    TBX_UINT8                  un8SsfType;

    TBX_UINT8                  aun8Padding0[1];
    TBX_BOOLEAN                fSsnPresent;
    TBX_BOOLEAN                fPointCodePresent;
    TBX_UINT8                  un8SSN;
    TB640_SS7_POINT_CODE       Pc;

    TB640_SS7_NETWORK_INDICATOR_BIT_TYPE NetworkBit;
    TB640_SS7_SCCP_ROUTING_TYPE RoutingIndicator;

    TB640_SS7_SCCP_GLOBAL_TITLE Gt;
} TB640_SS7_SCCP_ENTITY, *PTB640_SS7_SCCP_ENTITY;
```

General explanation of the parameters of Entity specific information:

- The present flag parameter (*fPresent*) specifies the presence of the address. Allowable values: **TBX_TRUE** or **TBX_FALSE**.
- The no point code in header flag parameter (*fNoPointCodeInHeader*) specifies if do not include the point code in the SCCP header. Allowable values: **TBX_TRUE** or **TBX_FALSE**.
- The sub-service field present flag parameter (*fSsfPresent*) indicates whether the sub-service field contains a valid value. Allowable values: **TBX_TRUE** or **TBX_FALSE**.

- The sub-service field (*un8SsfType*) specifies the type of network. Allowable values: see **Table 21 - SSF values**.
- The subsystem number present flag parameter (*fSsnPresent*) indicates the subsystem number indicator presence. Used to encode the subsystem number indicator in the address indicator octet of the address parameter in an SCCP message. Allowable values: **TBX_TRUE** or **TBX_FALSE**.
- The point code present flag parameter (*fPointCodePresent*) indicates the point code indicator presence. Used to encode the point code indicator in the address indicator octet of the address parameter in an SCCP message. Allowable values: **TBX_TRUE** or **TBX_FALSE**.
- The subsystem number field (*un8SSN*) specifies the subsystem number used to identify the SCCP user. Allowable values: see **Table 93 - SCCP SSN value to identify SCCP user**.
- The point code field (*Pc*) specifies the point code value.
- The network bit field (*NetworkBit*) specifies the network indicator bit. Allowable values: see **Table 88 - SCCP network indicator bit**.
- The routing indicator field (*RoutingIndicator*) specifies the routing indicator. Used to encode the routing indicator in the address indicator octet of the address parameter in a SCCP message. Allowable values:

Table 103 - SCCP routing type

Routing Type	Description
TB640_SS7_SCCP_ROUTING_TYPE_GLOBAL_TITLE	Routing should be based on the global title.
TB640_SS7_SCCP_ROUTING_TYPE_SSN	Routing should be based on the destination point code and the subsystem number.

- The GT parameter (*Gt*) specifies the global title. See Global Title explanation above.

The **response** part of the message TB640_MSG_ID_SS7_SCCP_OP_GTT_ADDRMAP_ALLOC contains the field:

```

...
TB640_SS7_SCCP_GTT_ADDRMAP_HANDLE    hSccpGttAddrmap; /* Handle of the instance of
...                                     the SCCP address map */

```

7.2.2 Compatibility

7.2.2.1 Variants

Has to be done.

7.3 SCCP Alarms

7.3.1 Alarm

The TB640_MSG_ID_SS7_SCCP_NOTIF_ALARM (event) notification message is received by the host application when a SCCP is reporting an error.

Structure contains the **event** notification alarm for a SCCP layer:

```
typedef struct _TB640_EVT_SS7_SCCP_NOTIF_ALARM
{
    TBX_MSG_HEADER                Header;
    TBX_UINT32                    un32MsgVersion;
    TB640_SS7_SCCP_HANDLE         hSccp;

    TB640_SS7_SCCP_ALARM_CATEGORY Category;
    TB640_SS7_SCCP_ALARM_EVENT    Event;
    TB640_SS7_SCCP_ALARM_CAUSE    Cause;
    TBX_UINT8                     aun8Padding0 [4];
} TB640_EVT_SS7_SCCP_NOTIF_ALARM, *PTB640_EVT_SS7_SCCP_NOTIF_ALARM;
```

General explanation of the field of event notification alarm:

- The SCCP handle field (*hSccp*) specifies the handle of the SCCP instance.
- The category field (*Category*) indicates the category of the alarm. Possible values:

Table 104 - SCCP alarm category

Alarm Category	Description
TB640_SS7_SCCP_ALARM_CATEGORY_PROTOCOL	Protocol related alarms.
TB640_SS7_SCCP_ALARM_CATEGORY_INTERFACE	Interface related alarms.

- The event field (*Event*) specifies the event that cause the alarm.
- The cause field (*Cause*) specifies the cause of the alarm.

Table 105 - SCCP event and cause alarm under CATEGORY_PROTOCOL

Event of Protocol Category	Cause	Description
TB640_SS7_SCCP_ALARM_EVENT_...	TB640_SS7_SCCP_ALARM_CAUSE_...	
PROTOCOL_INVALID_EVENT	PROTOCOL_DECODE_ERROR PROTOCOL_INV_NETWORK_MSG	Invalid network message. - Message decoding error. - Invalid message type.
ROUTING_ERROR	PROTOCOL_INVALID_ROUTE	Routing error.
HOP_VIOLATION	OUT_OF_RANGE	Message discarded due to hop counter violation.
UNEQUIPPED	PROTOCOL_UNEQUIPPED	MTP-STATUS with indication remote SCCP unequipped.

7.3.2 Lsap Alarm

The TB640_MSG_ID_SS7_SCCP_NOTIF_LSAP_ALARM (event) notification message is received by the host application when a SCCP Lsap is reporting an error.

Structure contains the **event** notification Lsap alarm for a SCCP layer:

```
typedef struct _TB640_EVT_SS7_SCCP_NOTIF_LSAP_ALARM
{
    TBX_MSG_HEADER                Header;
    TBX_UINT32                    un32MsgVersion;
    TB640_SS7_SCCP_HANDLE         hSccp;
    TB640_SS7_SCCP_LSAP_HANDLE    hSccpLsap;

    TB640_SS7_SCCP_ALARM_CATEGORY Category;
    TB640_SS7_SCCP_ALARM_EVENT    Event;
    TB640_SS7_SCCP_ALARM_CAUSE    Cause;
    TBX_UINT8                     aun8Padding0 [4];
} TB640_EVT_SS7_SCCP_NOTIF_LSAP_ALARM, *PTB640_EVT_SS7_SCCP_NOTIF_LSAP_ALARM;
```

General explanation of the field of event notification Lsap alarm:

- The SCCP handle field (*hSccp*) specifies the handle of the SCCP instance.
- The SCCP Lsap handle field (*hSccpLsap*) specifies the handle of the SCCP Lsap instance.
- The category field (*Category*) indicates the category of the Lsap alarm. Possible values: see **Table 104 - SCCP alarm category**.
- The event field (*Event*) specifies the event that cause the Lsap alarm.
- The cause field (*Cause*) specifies the cause of the Lsap alarm.

Table 106 - SCCP event and cause LSAP alarm under CATEGORY_PROTOCOL

Event of Protocol Category	Cause	Description
TB640_SS7_SCCP_ALARM_EVENT_...	TB640_SS7_SCCP_ALARM_CAUSE_...	
PROTOCOL_BIND_FAIL	UNKNOW	Bind with service provider failed.
PROTOCOL_SYNTAX_ERROR	PROTOCOL_INV_NETWORK_MSG	Syntax error in message received from the network.

Table 107 - SCCP event and cause LSAP alarm under CATEGORY_INTERFACE

Event of Interface Category	Cause	Description
TB640_SS7_SCCP_ALARM_EVENT_...	TB640_SS7_SCCP_ALARM_CAUSE_...	
INTERFACE_INVALID_LI_EVENT	OUT_OF_RANGE INTERFACE_INV_SU INTERFACE_PROTOCOL_NOT_ACTIVE	Invalid lower interface event because of: - Parameter value out of range. - Invalid suld. - Protocol layer not active.

7.3.3 Userpart Alarm

The TB640_MSG_ID_SS7_SCCP_NOTIF_USERPART_ALARM (event) notification message is received by the host application when a SCCP Userpart is reporting an error.

Structure contains the **event** notification Userpart alarm for a SCCP layer:

```
typedef struct _TB640_EVT_SS7_SCCP_NOTIF_USERPART_ALARM
{
    TBX_MSG_HEADER                Header;
    TBX_UINT32                    un32MsgVersion;
    TB640_SS7_SCCP_HANDLE         hSccp;
    TB640_SS7_SCCP_USERPART_HANDLE hSccpUserpart;

    TB640_SS7_SCCP_ALARM_CATEGORY Category;
    TB640_SS7_SCCP_ALARM_EVENT    Event;
    TB640_SS7_SCCP_ALARM_CAUSE    Cause;
    TBX_UINT8                     aun8Padding0 [4];
} TB640_EVT_SS7_SCCP_NOTIF_USERPART_ALARM, *PTB640_EVT_SS7_SCCP_NOTIF_USERPART_ALARM;
```

General explanation of the field of event notification Userpart alarm:

- The SCCP handle field (*hSccp*) specifies the handle of the SCCP instance.
- The SCCP Userpart handle field (*hSccpUserpart*) specifies the handle of the SCCP Userpart instance.
- The category field (*Category*) indicates the category of the Userpart alarm. Possible values: see **Table 104 - SCCP alarm category**.
- The event field (*Event*) specifies the event that cause the Userpart alarm.
- The cause field (*Cause*) specifies the cause of the Userpart alarm.

Table 108 - SCCP event and cause USERPART alarm under CATEGORY_PROTOCOL

Event of Protocol Category	Cause	Description
TB640_SS7_SCCP_ALARM_EVENT_...	TB640_SS7_SCCP_ALARM_CAUSE_...	
PROTOCOL_IN_SERVICE	PROTOCOL_USER_INITIATED	SCCP user came in-service (user initiated request).
PROTOCOL_OUT_OF_SERVICE	PROTOCOL_USER_INITIATED	SCCP user going OOS (user initiated request).

Table 109 - SCCP event and cause USERPART alarm under CATEGORY_INTERFACE

Event of Interface Category	Cause	Description
TB640_SS7_SCCP_ALARM_EVENT_...	TB640_SS7_SCCP_ALARM_CAUSE_...	
INTERFACE_INVALID_UI_EVENT	OUT_OF_RANGE INTERFACE_INV_SP INTERFACE_PROTOCOL_NOT_ACTIVE	Invalid upper interface event because of: - Parameter value out of range. - Invalid spId. - Protocol layer not active.

--	--	--

7.3.4 Error Performance Report

The TB640_MSG_ID_SS7_SCCP_NOTIF_ERROR_PERF_REPORT (event) notification message is received by the host application when a SCCP is reporting an error performance.

Structure contains the **event** notification error performance report for a SCCP layer:

```
typedef struct _TB640_EVT_SS7_SCCP_NOTIF_ERROR_PERF_REPORT
{
    TBX_MSG_HEADER                Header;
    TBX_UINT32                    un32MsgVersion;
    TB640_SS7_SCCP_HANDLE         hSccp;

    TB640_SS7_SCCP_REPORT_ERR_PERF_CAUSE Cause;
    TBX_UINT8                     aun8Padding0 [4];

    TB640_SS7_SCCP_REPORT         Report;
} TB640_EVT_SS7_SCCP_NOTIF_ERROR_PERF_REPORT, *PTB640_EVT_SS7_SCCP_NOTIF_ERROR_PERF_REPORT;
```

General explanation of the field of event notification error performance report:

- The SCCP handle field (*hSccp*) specifies the handle of the SCCP instance.
- The cause field (*Cause*) specifies the cause of the error performance report.

Table 110 - SCCP cause under error performance report

Cause	Used Report parameters	Description
TB640_SS7_SCCP_REPORT_ERR_PERF_CAUSE ...		
UNKNOW	None	Unknown.
RTF_NTBADADDR	- un32NetworkId - ProtocolVariant - Called Address - Calling Address	Routing failure - No translation for address of such nature.
RTF_NTSPECADDR	- un32NetworkId - ProtocolVariant - Called Address - Calling Address	Routing failure - No translation for this specific address.
RTF_NETWORK_FAIL	- un32NetworkId - ProtocolVariant - Calling Address	Routing failure - Network failure (point code not available).
RTF_NETWORK_CONGESTION	- un32NetworkId - ProtocolVariant - Dpc - Calling Address	Routing failure - Network congestion.
RTF_SUBSYSTEM_FAILURE	- un32NetworkId - ProtocolVariant	Routing failure - Subsystem failure (unavailable).
RTF_SUBSYSTEM_CONGESTION	- un32NetworkId - ProtocolVariant - Calling Address	Routing failure - Subsystem congestion.
RTF_UNEQUIPPED	- un32NetworkId - ProtocolVariant	Routing failure - Unequipped user (subsystem).
REASSEMBLY_OUTSEQ	- un32NetworkId - ProtocolVariant	Reassembly error - Segment received out of sequence.

	- Calling Address - un32CongestionLvl	
HOP_VIOLATION	- un32NetworkId - ProtocolVariant - Called Address - Calling Address	Hop counter violation (XUDT(S) / LUDT(S)).
SEGFAIL_TOO_LARGE	- un32NetworkId - ProtocolVariant	Message too large for segmentation.
SSC_RECV	- un32NetworkId - ProtocolVariant - Dpc - un32CongestionLvl	SCCP / Subsystem congested message received.
SSP_RECV	- un32NetworkId - ProtocolVariant - Dpc - un32SSN	Subsystem prohibited message received.

Structure contains the Report specific information parameters for error performance report:

```
typedef struct _TB640_SS7_SCCP_REPORT
{
    TBX_UINT32                un32StructVersion;
    TBX_UINT32                un32NetworkId;

    TB640_SS7_SCCP_ENTITY     CalledAddress;
    TB640_SS7_SCCP_ENTITY     CallingAddress;
    TB640_SS7_POINT_CODE     Dpc;

    TB640_SS7_SCCP_PROTOCOL_VARIANT ProtocolVariant;
    TBX_UINT32                un32SSN;
    TBX_UINT32                un32SegmentationRef;
    TBX_UINT32                un32CongestionLvl;
    TBX_UINT8                 aun8Padding0[4];
} TB640_SS7_SCCP_REPORT, *PTB640_SS7_SCCP_REPORT;
```

General explanation of the parameters of Report specific information:

- The network identifier parameter (*un32NetworkId*) specifies the ID of the network.
- The called address parameter (*CalledAddress*) specifies the called party address. See Entity explanation in section **7.2.1.7 GTT Address Map Configuration**.
- The calling address parameter (*CallingAddress*) specifies the calling party address. See Entity explanation in section **7.2.1.7 GTT Address Map Configuration**.
- The destination point code parameter (*Dpc*) specifies the point code of the node where the route is terminating.
- The protocol variant parameter (*ProtocolVariant*) specifies the SCCP protocol variant. See **Table 87 - SCCP protocol variant**.
- The subsystem number parameter (*un32SSN*) specifies a subsystem number. See **Table 93 - SCCP SSN value to identify SCCP user**.

- The segmentation reference parameter (*un32SegmentationRef*) specifies the SCCP segmentation local reference number.
- The congestion level parameter (*un32CongestionLvl*) specifies the remote SCCP congestion level (reception of SSC). Value is in range **0** to **7**.

7.4 SCCP States

States information, which can be gathered at any time by SCCP, indicates the current state of the node or route. This information can be used to determine the quality of service. The collection of status information does not affect any of the information examined.

7.4.1 General States Get

The TB640_MSG_ID_SS7_SCCP_STATES_GET (request/response) message is used to obtain states from a SCCP.

The **request** part of the message TB640_MSG_ID_SS7_SCCP_STATES_GET contains the field:

```

...
TB640_SS7_SCCP_HANDLE      hSccp;          /* The handle of the instance of the SCCP
                                     layer */
...

```

The **response** part of the message TB640_MSG_ID_SS7_SCCP_STATES_GET contains the field:

```

...
TBX_UINT8                  un8CongLevel;
TBX_UINT8                  un8SscThreshold;
...

```

General explanation of the states parameters:

- The congestion level field (*un8CongLevel*) indicates SCCP congestion level. Possible values: **0** to **7**.
 - ✍ If *un8CongLevel* is greater than zero, SCCP shall repeatedly generate the message SSC on receipt of every *un8SscThreshold* number of messages. If *un8CongLevel* is zero, SCCP shall stop generating message SSC.
- The SSC threshold field (*un8SscThreshold*) indicates threshold for repeating SSC message in case of SCCP congestion.
 - ✍ The *un8SscThreshold* field is useful when the congestion is greater than zero.

7.4.2 General States Set

The TB640_MSG_ID_SS7_SCCP_STATES_SET (request/response) message is used to request a states change for a SCCP.

The **request** part of the message TB640_MSG_ID_SS7_SCCP_STATES_SET contains the field:

```

...
TB640_SS7_SCCP_HANDLE      hSccp;          /* The handle of the instance of the SCCP
                                     layer */
TBX_UINT8                  un8CongLevel;
TBX_UINT8                  un8SscThreshold;
...

```

...

General explanation of the states parameters:

- The congestion level parameter (*un8CongLevel*) specifies the SCCP congestion level. Allowable values: **0** to **7**.
 - ✍ If *un8CongLevel* is greater than zero, SCCP shall repeatedly generate the message SSC on receipt of every *un8SscThreshold* number of messages. If *un8CongLevel* is zero, SCCP shall stop generating message SSC.
- The SSC threshold parameter (*un8SscThreshold*) specifies threshold for repeating SSC message in case of SCCP congestion. Allowable values: **0** to **255**.
 - ✍ The *un8SscThreshold* field is useful when the congestion is greater than zero.

The **response** part of the message TB640_MSG_ID_SS7_SCCP_STATES_SET contains the field:

```
...
TBX_RESULT                               Result;
...
```

General explanation of the states response parameters:

- Result code of the request operation.

7.4.3 Route States Get

The TB640_MSG_ID_SS7_SCCP_STATES_ROUTE_GET (request/response) message is used to obtain states from a SCCP route.

The **request** part of the message TB640_MSG_ID_SS7_SCCP_STATES_ROUTE_GET contains the field:

```
...
TB640_SS7_SCCP_ROUTE_HANDLE  hSccpRoute;    /* The handle of the instance of the SCCP
...                               route */
```

The **response** part of the message TB640_MSG_ID_SS7_SCCP_STATES_ROUTE_GET contains the field:

```
...
TB640_SS7_SCCP_POINT_CODE_STATES  PointCodeStates;
TBX_UINT32                          un32SubsystemNb;
TB640_SS7_SCCP_SUBSYSTEM_STATES    SubsystemStates[TB640_SS7_SCCP_MAX_SSN_PER_DPC];
...
```

General explanation of the states route parameters:

- The subsystem number field (*un32SubsystemNb*) indicates the number of subsystem states in the array *SubsystemStates*.

Structure contains the point code states parameters in the message:

```
typedef struct _TB640_SS7_SCCP_POINT_CODE_STATES
{
    TBX_UINT32                un32StructVersion;
    TB640_SS7_SCCP_POINT_CODE_STATE    PCState;
    TB640_SS7_SCCP_REMOTE_SCCP_STATE    RemoteState;
    TB640_SS7_SCCP_SMI_STATE            SmiState;
    TBX_UINT8                    un8Rl;
    TBX_UINT8                    un8Rs1;
    TBX_UINT8                    un8Rcl;
    TBX_UINT8                    un8Ril;
    TBX_UINT8                    aun8Padding0[4];
} TB640_SS7_SCCP_POINT_CODE_STATES, *PTB640_SS7_SCCP_POINT_CODE_STATES;
```

General explanation of the point code states parameters:

- The point code state field (*PCState*) indicates the point code status. Possible values:

Table 111 – SCCP point code status

Point Code Status	Description
TB640_SS7_SCCP_POINT_CODE_STATE_ACCESSIBLE	Signalling Point Accessible.
TB640_SS7_SCCP_POINT_CODE_STATE_INACCESSIBLE	Signalling Point Inaccessible.
TB640_SS7_SCCP_POINT_CODE_STATE_CONGESTED	Signalling Point Congested.

- The remote state field (*RemoteState*) indicates the remote SCCP status. Possible values:

Table 112 – SCCP remote status

Remote Status	Description
TB640_SS7_SCCP_REMOTE_SCCP_STATE_AVAILABLE	Remote SCCP available.
TB640_SS7_SCCP_REMOTE_SCCP_STATE_UNAVAILABLE	Remote SCCP unavailable, reason unknown.
TB640_SS7_SCCP_REMOTE_SCCP_STATE_UNEQUIPPED	Remote SCCP unequipped.
TB640_SS7_SCCP_REMOTE_SCCP_STATE_INACCESSIBLE	Remote SCCP inaccessible.
TB640_SS7_SCCP_REMOTE_SCCP_STATE_CONGESTED	Remote SCCP congested.

- The subsystem multiplicity indicator state field (*SmiState*) indicates the state of a subsystem multiplicity indicator can have. Possible values:

Table 113 – SCCP SMI status

SMI Status	Description
TB640_SS7_SCCP_SMI_STATE_SOLITARY	Subsystem is not replicated.
TB640_SS7_SCCP_SMI_STATE_DUPLICATED	Subsystem is replicated.
TB640_SS7_SCCP_SMI_STATE_UNKNOWN	Unknown.

- The restriction level field (*un8Rl*) indicates the restriction level for that destination point code. Possible values: **0** to **7**.
- The restriction sub-level field (*un8Rs1*) indicates the restriction sub-level for that destination point code. Possible values: **0** to **3**.
- The remote congestion level field (*un8Rcl*) indicates the congestion level for the remote SCCP. Possible values: **0** to **7**.

- The restricted importance level field (*un8Ril*) indicates the restricted importance level for that destination point code. This level is passed to user so that user can fill the importance of the subsequent messages to that destination point code. Possible values: **0** to **7**.

Structure contains the subsystem states parameters in the message:

```
typedef struct _TB640_SS7_SCCP_SUBSYSTEM_STATES
{
    TBX_UINT32                un32StructVersion;
    TBX_UINT8                 un8SSN;
    TBX_UINT8                 aun8Padding0[3];
    TB640_SS7_SCCP_SSN_STATE  SsnState;
    TB640_SS7_SCCP_SMI_STATE  SmiState;
} TB640_SS7_SCCP_SUBSYSTEM_STATES, *PTB640_SS7_SCCP_SUBSYSTEM_STATES;
```

General explanation of the subsystem states parameters:

- The subsystem number field (*un8SSN*) indicates the subsystem number. Possible values: **1** to **255**. See **Table 93 - SCCP SSN value to identify SCCP user**.
- The subsystem state field (*SsnState*) indicates the state of a subsystem can have. Possible values:

Table 114 – SCCP SSN status

SSN Status	Description
TB640_SS7_SCCP_SSN_STATE_USER_OOS	Subsystem user out of service.
TB640_SS7_SCCP_SSN_STATE_USER_IS	Subsystem user in service.

- The subsystem multiplicity indicator state field (*SmiState*) indicates the state of a subsystem multiplicity indicator can have. Possible values: see **Table 113 – SCCP SMI status**.

7.5 SCCP Statistics

The SCCP can gather statistics information at any time to measure the performance of the SCCP software. This information can be used to determine the distribution of traffic loads and Quality of Service (QoS) parameters, and to assist in SCCP software debugging. The collection of statistics information may or may not result in the counters being reset.

7.5.1 General Statistics

The TB640_MSG_ID_SS7_SCCP_STATS_GET (request/response) message is used to obtain statistics from a SCCP layer.

The **request** part of the message TB640_MSG_ID_SS7_SCCP_STATS_GET contains the field:

```
...
TB640_SS7_MTP3_HANDLE    hMtp3;        /* Handle of the MTP3 layer */
TBX_BOOL                 fResetStats;    /* Reset stats if required */
...
```

The **response** part of the message TB640_MSG_ID_SS7_SCCP_STATS_GET contains the field:


```

...
TB640_SS7_SCCP_STATS          Stats;
...

```

Structure contains the statistics for general layer SCCP:

```

typedef struct _TB640_SS7_SCCP_STATS
{
    TBX_UINT32          un32StructVersion;
    TBX_UINT32          un32NoTransASN;
    TBX_UINT32          un32NoTransSA;
    TBX_UINT32          un32NetFail;
    TBX_UINT32          un32NetCong;
    TBX_UINT32          un32SsFail;
    TBX_UINT32          un32SsCong;
    TBX_UINT32          un32Unequip;
    TBX_UINT32          un32HopViolate;
    TBX_UINT32          un32SynError;
    TBX_UINT32          un32Unknown;
    TBX_UINT32          un32SsCongRx;
    TBX_UINT32          un32SsProhRx;
    TBX_UINT32          un32UnitDataTx;
    TBX_UINT32          un32UnitDataSrvTx;
    TBX_UINT32          un32UnitDataRx;
    TBX_UINT32          un32UnitDataSrvRx;
    TBX_UINT32          un32XUnitDataTx;
    TBX_UINT32          un32XUnitDataSrvTx;
    TBX_UINT32          un32XUnitDataRx;
    TBX_UINT32          un32XUnitDataSrvRx;
    TBX_UINT32          un32MsgHand;
    TBX_UINT32          un32MsgLoc;
    TBX_UINT32          un32GttReq;
    TBX_UINT32          un32MsgTxC0;
    TBX_UINT32          un32MsgTxC1;
    TBX_UINT32          un32MsgRxC0;
    TBX_UINT32          un32MsgRxC1;
    TBX_UINT32          un32LUnitDataTx;
    TBX_UINT32          un32LUnitDataSrvTx;
    TBX_UINT32          un32LUnitDataRx;
    TBX_UINT32          un32LUnitDataSrvRx;
    TBX_UINT32          un32SegErr;
    TBX_UINT32          un32SegErrFail;
    TBX_UINT32          un32ReassemErr;
    TBX_UINT32          un32ReassemErrTimExp;
    TBX_UINT32          un32ReassemErrNoSpc;
    TBX_UINT32          un32InaTimerExpiry;
    TBX_UINT32          un32RtFailInvInsRtReq;
    TBX_UINT32          un32RtFailInvIsniRtReq;
    TBX_UINT32          un32FailIsniConRt;
    TBX_UINT32          un32FailRedIsniRtReq;
    TBX_UINT32          un32FailIsniNetId;

} TB640_SS7_SCCP_STATS, *PTB640_SS7_SCCP_STATS;

```

General explanation of the general layer SCCP statistics parameters:

- The no translation address such nature counter (*un32NoTransASN*) indicates the number of routing failure because of a no translation for address of such nature.

- The no translation specific address counter (*un32NoTransSA*) indicates the number of routing failure because of a no translation for this specific address.
- The network failure counter (*un32NetFail*) indicates the number of routing failure because of a network failure (point code unavailable).
- The network congestion counter (*un32NetCong*) indicates the number of routing failure because of network congestion.
- The subsystem failure counter (*un32SsFail*) indicates the number of routing failure because of subsystem failure.
- The subsystem congestion counter (*un32SsCong*) indicates the number of routing failure because of subsystem congestion.
- The unequipped counter (*un32Unequip*) indicates the number of routing failure because of unequipped user.
- The hop violate counter (*un32HopViolate*) indicates the number of routing failure because of hop counter violation.
- The syntax error counter (*un32SynError*) indicates the number of syntax error.
- The unknown counter (*un32Unknown*) indicates the number of routing failure because of reason unknown.
- The subsystem congestion received counter (*un32SsCongRx*) indicates the number of SCCP / subsystem congested messages received.
- The prohibited received counter (*un32SsProhRx*) indicates the number of subsystem prohibited messages received.
- The unit data transmitted counter (*un32UnitDataTx*) indicates the number of unit data sent.
- The unit data service transmitted counter (*un32UnitDataSrvTx*) indicates the number of unit data service sent.
- The unit data received counter (*un32UnitDataRx*) indicates the number of unit data received.
- The unit data service received counter (*un32UnitDataSrvRx*) indicates the number of unit data service received.
- The extended unit data transmitted counter (*un32XUnitDataTx*) indicates the number of extended unit data sent.
- The extended unit data service transmitted counter (*un32XUnitDataSrvTx*) indicates the number of extended unit data service sent.
- The extended unit data received counter (*un32XUnitDataRx*) indicates the number of extended unit data received.

- The extended unit data service received counter (*un32XUnitDataSrvRx*) indicates the number of extended unit data service received.
- The message handle counter (*un32MsgHand*) indicates the total of messages handled.
- The message local counter (*un32MsgLoc*) indicates the total messages intended for local subsystems.
- The GTT request counter (*un32GttReq*) indicates the number of messages requiring global address translation.
- The message transmitted class 0 counter (*un32MsgTxCO*) indicates the number of class 0 messages sent.
- The message transmitted class 1 counter (*un32MsgTxCI*) indicates the number of class 1 messages sent.
- The message received class 0 counter (*un32MsgRxCO*) indicates the number of class 0 messages received.
- The message received class 1 counter (*un32MsgRxCI*) indicates the number of class 1 messages received.
- The long unit data transmitted counter (*un32LUnitDataTx*) indicates the number of long unit data sent.
- The long unit data service transmitted counter (*un32LUnitDataSrvTx*) indicates the number of long unit data service sent.
- The long unit data received counter (*un32LUnitDataRx*) indicates the number of long unit data received.
- The long unit data service received counter (*un32LUnitDataSrvRx*) indicates the number of long unit data service received.
- The segmentation error counter (*un32SegErr*) indicates the number of segmentation errors (message too large for segmentation).
- The segmentation error fail counter (*un32SegErrFail*) indicates the number of segmentation failed errors (case of LUDT segmentation at interworking node).
- The reassembly error counter (*un32ReassemErr*) indicates the number of reassembly out of sequence errors (including duplicate segment and non-first segment).
- The reassembly error timer expiry counter (*un32ReassemErrTimExp*) indicates the number of reassembly errors timer expiry.
- The reassembly error no space (*un32ReassemErrNoSpc*) indicates the number of reassembly errors (no space for reassembly).

- The inactivity timer expire counter (*un32InaTimerExpiry*) indicates the number of inactivity timer expire.
- The routing failure invalid INS routing request counter (*un32RtFailInvInsRtReq*) indicates the number of routing failure invalid INS routing request.
- The routing failure invalid ISNI routing request counter (*un32RtFailInvIsniRtReq*) indicates the number of routing failure invalid ISNI routing request.
- The failure ISNI constrained routing counter (*un32FailIsniConRt*) indicates the number of failure ISNI constrained routing.
- The failure redundant ISNI routing request counter (*un32FailRedIsniRtReq*) indicates the number of failure due to redundant ISNI routing request.
- The failure ISNI network identifier counter (*un32FailIsniNetId*) indicates the number of failure in ISNI identification of network.

7.5.2 Userpart Statistics

The TB640_MSG_ID_SS7_SCCP_STATS_USERPART_GET (request/response) message is used to obtain statistics from a SCCP Userpart.

The **request** part of the message TB640_MSG_ID_SS7_SCCP_STATS_USERPART_GET contains the field:

```
...
TB640_SS7_SCCP_USERPART_HANDLE    hSccpUserpart; /* Handle of the SCCP userpart instance */
TBX_BOOL                          fResetStats;    /* Reset stats if required */
...
```

The **response** part of the message TB640_MSG_ID_SS7_SCCP_STATS_USERPART_GET contains the field:

```
...
TB640_SS7_SCCP_USERPART_STATS     Statistics;
...
```

Structure contains the statistics for a SCCP Userpart:

```
typedef struct _TB640_SS7_SCCP_USERPART_STATS
{
    TBX_UINT32          un32StructVersion;
    TBX_UINT32          un32SsOOSReqGr;
    TBX_UINT32          un32SsOOSReqDn;
    TBX_UINT32          un32SsProhStart;
    TBX_UINT32          un32SsProhStop;
    TBX_UINT32          un32MsgTxBSS;
    TBX_UINT32          un32MsgTxC0;
    TBX_UINT32          un32MsgTxC1;
    TBX_UINT32          un32MsgRxC0;
    TBX_UINT32          un32MsgRxC1;
} TB640_SS7_SCCP_USERPART_STATS, *PTB640_SS7_SCCP_USERPART_STATS;
```

General explanation of the SCCP Userpart statistics parameters:

- The subsystem out of service request granted counter (*un32SsOOSReqGr*) indicates the number of subsystem out of service request granted.
- The subsystem out of service request denied counter (*un32SsOOSReqDn*) indicates the number of subsystem out of service request denied.
- The subsystem prohibited start counter (*un32SsProhStart*) indicates the number of start of subsystem prohibited.
- The subsystem prohibited stop counter (*un32SsProhStop*) indicates the number of stop of subsystem prohibited.
- The messages transmitted to backup subsystem counter (*un32MsgTxBSS*) indicates the number of messages sent to backup subsystem.
- The messages transmitted class 0 counter (*un32MsgTxCO*) indicates the number of class 0 messages sent by the user.
- The messages transmitted class 1 counter (*un32MsgTxCI*) indicates the number of class 1 messages sent by the user.
- The messages received class 0 counter (*un32MsgRxCO*) indicates the number of class 0 messages received for the user.
- The messages received class 1 counter (*un32MsgRxCI*) indicates the number of class 1 messages received for the user.

7.5.3 Lsap Statistics

The TB640_MSG_ID_SS7_SCCP_STATS_LSAP_GET (request/response) message is used to obtain statistics from a SCCP Lsap.

The **request** part of the message TB640_MSG_ID_SS7_SCCP_STATS_LSAP_GET contains the field:

```
...
TB640_SS7_SCCP_LSAP_HANDLE  hSccpLsap;    /* Handle of the SCCP lsap instance */
TBX_BOOL                    fResetStats; /* Reset stats if required */
...
```

The **response** part of the message TB640_MSG_ID_SS7_SCCP_STATS_LSAP_GET contains the field:

```
...
TB640_SS7_SCCP_LSAP_STATS   Statistics;
...
```

Structure contains the statistics for a SCCP Lsap:

```
typedef struct _TB640_SS7_SCCP_LSAP_STATS
{
    TBX_UINT32                un32StructVersion;
    TBX_UINT32                un32SsAllTx;
```

```
TBX_UINT32          un32SsOutGTx;  
TBX_UINT32          un32SsOutRTx;  
TBX_UINT32          un32SsProhTx;  
TBX_UINT32          un32SsStatTx;  
TBX_UINT32          un32SsAllRx;  
TBX_UINT32          un32SsOOSGRx;  
TBX_UINT32          un32SsOOSRRx;  
TBX_UINT32          un32SsProhRx;  
TBX_UINT32          un32SsStatRx;  
TBX_UINT32          un32SsCongTx;  
TBX_UINT32          un32SsCongRx;  
TBX_UINT8           aun8Padding0 [4];  
  
} TB640_SS7_SCCP_LSAP_STATS, *PTB640_SS7_SCCP_LSAP_STATS;
```

General explanation of the SCCP Lsap statistics parameters:

- The subsystem allowed transmitted counter (*un32SsAllTx*) indicates the number of subsystem allowed transmitted.
- The subsystem out of service grant transmitted counter (*un32SsOutGTx*) indicates the number of subsystem out of service grant transmitted.
- The subsystem out of service request transmitted counter (*un32SsOutRTx*) indicates the number of subsystem out of service request transmitted.
- The subsystem prohibited transmitted counter (*un32SsProhTx*) indicates the number of subsystem prohibited transmitted.
- The subsystem status transmitted counter (*un32SsStatTx*) indicates the number of subsystem status test transmitted.
- The subsystem allowed received counter (*un32SsAllRx*) indicates the number of subsystem allowed received.
- The subsystem out of service grant received counter (*un32SsOOSGRx*) indicates the number of subsystem out of service grant received.
- The subsystem out of service request received counter (*un32SsOOSRRx*) indicates the number of subsystem out of service request received.
- The subsystem prohibited received counter (*un32SsProhRx*) indicates the number of subsystem prohibited received.
- The subsystem status received counter (*un32SsStatRx*) indicates the number of subsystem status test received.
- The subsystem congested transmitted counter (*un32SsCongTx*) indicates the number of subsystem congested transmitted.
- The subsystem congested received counter (*un32SsCongRx*) indicates the number of subsystem congested received.

7.5.4 Route Statistics

The TB640_MSG_ID_SS7_SCCP_STATS_ROUTE_GET (request/response) message is used to obtain statistics from a SCCP Route.

The **request** part of the message TB640_MSG_ID_SS7_SCCP_STATS_ROUTE_GET contains the field:

```

...
TB640_SS7_SCCP_ROUTE_HANDLE  hSccpRoute;    /* Handle of the SCCP route instance */
TBX_BOOL                      fResetStats;    /* Reset stats if required */
...

```

The **response** part of the message TB640_MSG_ID_SS7_SCCP_STATS_ROUTE_GET contains the field:

```

...
TBX_UINT32                    un32SubsystemNb;    /* Number of subsystem in the
TB640_SS7_SCCP_SUBSYSTEM_STATS  SubsystemStats[ TB640_SS7_SCCP_MAX_SSN_PER_DPC ];
...

```

Structure contains the statistics for a SCCP Subsystem:

```

typedef struct _TB640_SS7_SCCP_SUBSYSTEM_STATS
{
    TBX_UINT32                    un32StructVersion;
    TBX_UINT32                    un32SSN;
    TBX_UINT32                    un32SsOOSGRx;
    TBX_UINT32                    un32SsOOSRDenied;
    TBX_UINT32                    un32SsAllRx;
    TBX_UINT8                     aun8Padding0 [4];
} TB640_SS7_SCCP_SUBSYSTEM_STATS, *PTB640_SS7_SCCP_SUBSYSTEM_STATS;

```

General explanation of the SCCP Subsystem statistics parameters:

- The subsystem number (*un32SSN*) indicates the subsystem number. Possible values: See **Table 93 - SCCP SSN value to identify SCCP user**.
- The subsystem out of service grant received counter (*un32SsOOSGRx*) indicates the number of subsystem out of service grant received.
- The subsystem out of service request denied counter (*un32SsOOSRDenied*) indicates the number of subsystem out of service request denied.
- The subsystem allowed received counter (*un32SsAllRx*) indicates the number of subsystem allowed received.

7.6 SCCP Standalone mode

The SCCP standalone mode is only active when the SCCP Userpart connection mode is set to TB640_SS7_SCCP_CONNECTION_MODE_HOST_TCAP.

7.6.1 Data Request

The TB640_MSG_ID_SS7_SCCP_OP_DATA_REQUEST (request/response) message is used to transfer user data using unit data or extended unit data (or long unit data in case underlying network support broadband) SCCP messages from the host application to a SCCP Userpart.

The **request** part of the message TB640_MSG_ID_SS7_SCCP_OP_DATA_REQUEST contains the field:

```

...
TB640_SS7_SCCP_USERPART_HANDLE      hSccpUserpart;          /* The handle of the SCCP
                                userpart */
TB640_SS7_SCCP_UNIT_DATA             UnitData;
TBX_UINT32                           un32FrameSize;          /* Byte count for this frame*/
TBX_UINT8                             aun8Payload [1];         /* Array containing the payload
                                raw data of frame */
...

```

Structure contains the unit data configuration parameters for the data frame to send:

```

typedef struct _TB640_SS7_SCCP_UNIT_DATA
{
    TBX_UINT32                          un32StructVersion;
    TB640_SS7_SCCP_PROTOCOL_CLASS       Class;
    TB640_SS7_SCCP_MSG_PRIORITY         MsgPriority;
    TBX_BOOL                             fReturnOnError;

    TB640_SS7_SCCP_ENTITY               CalledAddress;
    TB640_SS7_SCCP_ENTITY               CallingAddress;

    TBX_UINT32                          un32SequenceControl;
    TBX_UINT8                            un8ImportancePresent;
    TBX_UINT8                            un8Importance;
    TBX_UINT8                            aun8Padding0 [2];

    TB640_SS7_SCCP_ISNI                 Isni;
    TB640_SS7_SCCP_INS                   Ins;
} TB640_SS7_SCCP_UNIT_DATA, *PTB640_SS7_SCCP_UNIT_DATA;

```

General explanation of the parameters of unit data configuration:

- The class parameter (*Class*) specifies the protocol class. Allowable values:

Table 115 - SCCP protocol class

Protocol Class	Description
TB640_SS7_SCCP_PROTOCOL_CLASS_0	Connectionless non-sequenced
TB640_SS7_SCCP_PROTOCOL_CLASS_1	Connectionless sequenced

- The message priority parameter (*MsgPriority*) determines the priority of messages. Allowable values: see **Table 95 - SCCP message priority**.
- The return on error flag parameter (*fReturnOnError*) return message on error if set to TBX_TRUE otherwise discard message. Allowable values: **TBX_TRUE** or **TBX_FALSE**.







- The called address parameter (*CalledAddress*) specifies the called party address. See Entity definition in section 7.3.4 Error Performance Report.
- The calling address parameter (*CallingAddress*) specifies the calling party address. See Entity definition in section 7.3.4 Error Performance Report.
- The sequence control parameter (*un32SequenceControl*) specifies the sequence control (SC) value. Used to indicate that subsequent unit data messages with the same SC value must be delivered in sequence. Allowable values: **0 - 4294967295**.
 -  Valid for protocol class 1 message only.
- The importance present parameter (*un8ImportancePresent*) indicates the presence of the importance value. Allowable values:

Table 116 - SCCP present flag

Present Flag	Description
TB640_SS7_SCCP_PRESENT_VALUE_NOT_PRSNT	Not present.
TB640_SS7_SCCP_PRESENT_VALUE_PRSNT_NODEF	Present - no default.
TB640_SS7_SCCP_PRESENT_VALUE_PRSNT_DEF	Present - default.

- The importance parameter (*un8Importance*) specifies the value of importance. If the value of importance passed by user is greater than maximum allowable value for the requested primitive data then SCCP will assign the maximum allowable value to the importance. The allowable values for UDT/XUDT/LUDT SCCP messages are: **0 – 6**.
 -  Allowed for **ITU96** only.
- The ISNI parameter (*ISNI*) specifies the Intermediate Signalling Network Identification. See below for ISNI definition
 -  Allowed for **ANSI96** and **TELCORDIA** only.
 -  In other SCCP variants, its presence should be set to **TB640_SS7_SCCP_PRESENT_VALUE_NOT_PRSNT**.
- The INS parameter (*INS*) specifies the Intermediate Network Selection. See below for INS definition
 -  Allowed for **TELCORDIA** only.
 -  In other SCCP variants, its presence should be set to **TB640_SS7_SCCP_PRESENT_VALUE_NOT_PRSNT**.

Structure contains the ISNI configuration parameters for the unit data:

```
typedef struct _TB640_SS7_SCCP_ISNI
{
    TBX_UINT8          un8Present;
    TBX_UINT8          un8MiInd;
    TBX_UINT8          un8RteInd;
    TBX_UINT8          un8TypeInd;
    TBX_UINT8          un8Counter;
    TBX_UINT8          un8RteCtrlExt;
}
```

```

    TBX_UINT8          aun8Padding0 [1];
    TBX_UINT8          un8NbNids;
    TBX_UINT16         aun16Nid[TB640_SS7_SCCP_MAX_ISNI_NID];
    TBX_UINT8          aun8Padding1 [2];

} TB640_SS7_SCCP_ISNI, *PTB640_SS7_SCCP_ISNI;

```

General explanation of the parameters of ISNI configuration:

- The present parameter (*un8Present*) indicates the presence of the ISNI. Allowable values: see **Table 116 - SCCP present flag**.
- The mark identification parameter (*un8MiInd*) specifies the ISNI mark for identification (MI). Used to decide whether to identify network or not. Allowable values:

Table 117 - SCCP ISNI mark identification

ISNI Mark Identification	Description
TB640_SS7_SCCP_ISNI_MI_VALUE_NIDENT_NW	Do not identify network.
TB640_SS7_SCCP_ISNI_MI_VALUE_NIDENT_NW	Identify network.

- The route indicator parameter (*un8RteInd*) specifies the ISNI routing indicator (RI). Allowable values:

Table 118 - SCCP ISNI routing indicator

ISNI Routing Indicator	Description
TB640_SS7_SCCP_ISNI_RI_VALUE_NEITH	Neither constrained nor suggested ISNI routing.
TB640_SS7_SCCP_ISNI_RI_VALUE_CONST	Constrained ISNI routing.
TB640_SS7_SCCP_ISNI_RI_VALUE_SUGGST	Suggested ISNI routing. This is kept for further study in ANSI96 and TELCORDIA. <u>Not supported</u> in current implementation.

- The type identification parameter (*un8TypeInd*) specifies the ISNI type indicator (TI). Allowable values:

Table 119 - SCCP ISNI type indicator

ISNI Type Indicator	Description
TB640_SS7_SCCP_ISNI_TI_VALUE_0	ISNI type zero indicator.
TB640_SS7_SCCP_ISNI_TI_VALUE_1	ISNI type one indicator. ISNI Type 1 is for network specific routing and is <u>not supported</u> in SCCP.

- The counter parameter (*un8Counter*) specifies the ISNI imaginary pointer to locate routing and identification information in the NID list. Allowable values: **0 - 6**.
- The routing control extension parameter (*un8RteCtrlExt*) specifies the ISNI routing control extension octet. Present only in Type 1 ISNI format and contains information for network specific routing.
- The number of NID parameter (*un8NbNids*) specifies the number of NID in the list. Allowable values: **0 - 7**.

- The array of NID parameter (*aun16Nid*) specifies the network identifier list. Network Identifier (NID) consists of the one-octet Network Identifier followed by the one-octet Cluster Identifier. For Large networks, the second octet may be coded all zeros or may be coded with the Cluster Identifier of a particular node within the large network. For small networks, two non-zero octets are required to identify the network.

Structure contains the INS configuration parameters for the unit data:

```
typedef struct _TB640_SS7_SCCP_INS
{
    TBX_UINT8          un8Present;
    TBX_UINT8          un8InfTypeInd;
    TBX_UINT8          un8RteInd;
    TBX_UINT8          un8Counter;

    TBX_UINT8          aun8Padding0 [1];
    TBX_UINT8          un8NbNids;
    TBX_UINT16         aun16Nid[TB640_SS7_SCCP_MAX_INS_NID];
    TBX_UINT8          aun8Padding1 [6];
} TB640_SS7_SCCP_INS, *PTB640_SS7_SCCP_INS;
```

General explanation of the parameters of INS configuration:

- The present parameter (*un8Present*) indicates the presence of the INS. Allowable values: see **Table 116 - SCCP present flag**.
- The information type indicator parameter (*un8InfTypeInd*) specifies the INS information type indicator. Allowable values:

Table 120 - SCCP INS information type indicator

INS Information Type Indicator	Description
TB640_SS7_SCCP_INS_ITI_VALUE_SS7FRMT	SS7 format.
TB640_SS7_SCCP_INS_ITI_VALUE_NW1	Network specific 1.
TB640_SS7_SCCP_INS_ITI_VALUE_NW2	Network specific 2.

- The routing indicator parameter (*un8RteInd*) specifies the INS routing indicator. Allowable values:

Table 121 - SCCP INS routing indicator

INS Routing Indicator	Description
TB640_SS7_SCCP_INS_RI_VALUE_NEITH	Neither constrained nor suggested INS routing.
TB640_SS7_SCCP_INS_RI_VALUE_CONST	Constrained INS routing.
TB640_SS7_SCCP_INS_RI_VALUE_SUGGST	Suggested INS routing.

- The counter parameter (*un8Counter*) specifies the INS imaginary pointer to locate routing information in the NID list. Allowable values: **0 - 1**.
- The number of NID parameter (*un8NbNids*) specifies the number of NID in the list. Allowable values: **0 - 2**.

- The array of NID parameter (*aun16Nid*) specifies the network identifier list. Network Identifier (NID) consists of the one-octet Network Identifier followed by the one-octet Cluster Identifier. For Large networks, the second octet may be coded all zeros or may be coded with the Cluster Identifier of a particular node within the large network. For small networks, two non-zero octets are required to identify the network.

The **response** part of the message TB640_MSG_ID_SS7_SCCP_OP_DATA_REQUEST contains the fields:

```
...  
    TBX_RESULT    Result;        /* Result code of the request operation */  
...
```

7.6.2 Coordinate Request

The TB640_MSG_ID_SS7_SCCP_OP_COOR_REQUEST (request/response) message is used to send a coordinate request message from the host application to a SCCP Userpart.

A duplicated upper service uses this message to request permission to go out-of-service. A subsystem is duplicated when a backup subsystem is configured against the SCCP userpart instance. If the request is granted, the upper service receives the TB640_MSG_ID_SS7_SCCP_NOTIF_COOR_CONFIRM message. If the request is not granted, no response is given to the ITU upper service, implicitly denying the request. ANSI and TELCORDIA upper service receive the TB640_MSG_ID_SS7_SCCP_NOTIF_COOR_CONFIRM message with the subsystem multiplicity indicator set to TB640_SS7_SCCP_SMI_TYPE_DENIED (see **Table 125 - SCCP SMI type**).

The **request** part of the message TB640_MSG_ID_SS7_SCCP_OP_COOR_REQUEST contains the field:

```
...  
    TB640_SS7_SCCP_USERPART_HANDLE    hSccpUserpart;  
    TBX_UINT32                          un32SSN;  
...
```

General explanation of the parameters of coordinate request:

- The handle SCCP userpart parameter (*hSccpUserpart*) specifies the handle of the SCCP userpart.
- The subsystem number parameter (*un32SSN*) specifies the subsystem number used to identify SCCP user. The affected subsystem number is used to indicate the subsystem requesting a coordinated state change. The affected subsystem is encoded in the affected SSN field of the SCCP subsystem out-of-service request message. Allowable values: see **Table 93 - SCCP SSN value to identify SCCP user**.

The **response** part of the message TB640_MSG_ID_SS7_SCCP_OP_COOR_REQUEST contains the fields:

```
...  
    TBX_RESULT    Result;        /* Result code of the request operation */
```

...

7.6.3 Coordinate Response

The TB640_MSG_ID_SS7_SCCP_OP_COOR_RESPONSE (request/response) message is used to send a coordinate response message from the host application to a SCCP Userpart.

This message indicates that the upper service has sufficient resources to allow its mate to go out-of-service.

The **request** part of the message TB640_MSG_ID_SS7_SCCP_OP_COOR_RESPONSE contains the field:

```
...
TB640_SS7_SCCP_USERPART_HANDLE      hSccpUserpart;
TBX_UINT32                           un32SSN;
...
```

General explanation of the parameters of coordinate response:

- The handle SCCP userpart parameter (*hSccpUserpart*) specifies the handle of the SCCP userpart.
- The subsystem number parameter (*un32SSN*) specifies the subsystem number used to identify SCCP user. The affected subsystem number is used to indicate the subsystem requesting a coordinated state change. The affected subsystem is encoded in the affected SSN field of the SCCP subsystem out-of-service grant message. Allowable values: see **Table 93 - SCCP SSN value to identify SCCP user**.

The **response** part of the message TB640_MSG_ID_SS7_SCCP_OP_COOR_RESPONSE contains the fields:

```
...
TBX_RESULT      Result;          /* Result code of the request operation */
...
```

7.6.4 Point Code SSN Status Request

The TB640_MSG_ID_SS7_SCCP_OP_PC_SSN_STATUS_REQ (request/response) message is used to send status request message from the host application to a SCCP Userpart.

The upper service uses this request message to enquire the status of concerned point codes and subsystems. Normally, the SCCP userpart instance provides the status of concerned point codes and subsystems to the upper service after the upper service has initiated a bind with the SCCP userpart instance. Thereafter, any change in status is indicated via the TB640_MSG_ID_SS7_SCCP_NOTIF_SSN_STATE_CHANGE_IND and TB640_MSG_ID_SS7_SCCP_NOTIF_PC_STATE_CHANGE_IND messages.

To recover from the loss of TB640_MSG_ID_SS7_SCCP_NOTIF_SSN_STATE_CHANGE_IND and TB640_MSG_ID_SS7_SCCP_NOTIF_PC_STATE_CHANGE_IND messages, the upper

service may periodically enquire about the status of inaccessible concerned point codes and subsystems.

The **request** part of the message TB640_MSG_ID_SS7_SCCP_OP_PC_SSN_STATUS_REQ contains the field:

```

...
TB640_SS7_SCCP_USERPART_HANDLE          hSccpUserpart;
TB640_SS7_SCCP_REQUEST_STATUS_TYPE     RequestType;
TBX_UINT32                               un32SSN;
TB640_SS7_POINT_CODE                    Dpc;
...

```

General explanation of the parameters of point code SSN status request:

- The handle SCCP userpart parameter (*hSccpUserpart*) specifies the handle of the SCCP userpart.
- The request type parameter (*RequestType*) specifies status request type. Allowable values:

Table 122 - SCCP request status type

Request Status Type	Description
TB640_SS7_SCCP_REQUEST_STATUS_TYPE_PC	Point code status.
TB640_SS7_SCCP_REQUEST_STATUS_TYPE_SSN	Subsystem status.

- The subsystem number parameter (*un32SSN*) specifies the subsystem number used to identify SCCP user. Allowable values: see **Table 93 - SCCP SSN value to identify SCCP user**.
- The destination point code parameter (*Dpc*) specifies the enquired concerned point code. Proper point code length is: see **Table 3 - DPC Length**.

The **response** part of the message TB640_MSG_ID_SS7_SCCP_OP_PC_SSN_STATUS_REQ contains the fields:

```

...
TBX_RESULT    Result;          /* Result code of the request operation */
...

```

7.6.5 SSN State Change Request

The TB640_MSG_ID_SS7_SCCP_OP_SSN_STATE_CHANGE_REQ (request/response) message is used to send state request message from the host application to a SCCP Userpart.

This request message is used to inform the SCCP userpart instance about the state of the upper service. The upper service expects TB640_MSG_ID_SS7_SCCP_NOTIF_SSN_STATE_CHANGE_CFM as a response. The upper service retries the state change request if the SCCP userpart instance does not send a TB640_MSG_ID_SS7_SCCP_NOTIF_SSN_STATE_CHANGE_CFM response.

The **request** part of the message TB640_MSG_ID_SS7_SCCP_OP_SSN_STATE_CHANGE_REQ contains the field:

```

...
TB640_SS7_SCCP_USERPART_HANDLE      hSccpUserpart;
TB640_SS7_SCCP_STATUS_TYPE          Status;
...

```

General explanation of the parameters of SSN state change request:

- The handle SCCP userpart parameter (*hSccpUserpart*) specifies the handle of the SCCP userpart.
- The status parameter (*Status*) indicates the status of the service user. Allowable values: **TB640_SS7_SCCP_STATUS_TYPE_OUT_OF_SERVICE** OR **TB640_SS7_SCCP_STATUS_TYPE_IN_SERVICE**.

Table 123 - SCCP status type

Status Type	Description
TB640_SS7_SCCP_STATUS_TYPE_ACCESSIBLE	Signalling Point Accessible.
TB640_SS7_SCCP_STATUS_TYPE_INACCESSIBLE	Signalling Point Inaccessible.
TB640_SS7_SCCP_STATUS_TYPE_CONGESTED	Signalling Point Congested.
TB640_SS7_SCCP_STATUS_TYPE_OUT_OF_SERVICE	Subsystem user out of service.
TB640_SS7_SCCP_STATUS_TYPE_IN_SERVICE	Subsystem user in service.

The **response** part of the message TB640_MSG_ID_SS7_SCCP_OP_SSN_STATE_CHANGE_REQ contains the fields:

```

...
    TBX_RESULT      Result;          /* Result code of the request operation */
...

```

7.6.6 Data Received

The TB640_MSG_ID_SS7_SCCP_NOTIF_DATA_RECEIVED (event) message is used to send a received data notification message, unit data or extended unit data (or long unit data in case underlying network support broadband) from a SCCP Userpart to the host application.

Structure contains the **event** notification data received from a SCCP Userpart:

```

typedef struct _TB640_EVT_SS7_SCCP_NOTIF_DATA_RECEIVED
{
    TBX_MSG_HEADER                Header;
    TBX_UINT32                    un32MsgVersion;
    TB640_SS7_SCCP_USERPART_HANDLE hSccpUserpart;

    TB640_SS7_POINT_CODE          Opc;
    TB640_SS7_SCCP_UNIT_DATA      UnitData;
    TBX_UINT32                    un32FrameSize;
    TBX_UINT8                     aun8Payload [1];
} TB640_EVT_SS7_SCCP_NOTIF_DATA_RECEIVED, *PTB640_EVT_SS7_SCCP_NOTIF_DATA_RECEIVED;

```

General explanation of the field of event notification data received:

- The handle SCCP userpart parameter (*hSccpUserpart*) specifies the handle of the SCCP userpart.
- The OPC parameter (*Opc*) specifies the originating point code received from the MTP3 routing label.

- The unit data (*UnitData*) parameter. For description see section 7.6.1 Data Request.
- The frame size parameter (*un32FrameSize*) specifies the byte count for this frame in the *aun8Payload* parameter.
- The payload parameter (*aun8Payload*) contains the raw payload data of the received frame.

7.6.7 Data Error Indication

The TB640_MSG_ID_SS7_SCCP_NOTIF_DATA_ERROR_INDICATION (event) message is used to send an undelivered SCCP data messages from a SCCP Userpart to the host application. This message is returned if the return option field (*fReturnOnError*) in TB640_SS7_SCCP_UNIT_DATA is set to return a message upon detection of an error in the TB640_MSG_ID_SS7_SCCP_OP_DATA_REQUEST message used to send the message.

Structure contains the **event** notification data error indication from a SCCP Userpart:

```
typedef struct _TB640_EVT_SS7_SCCP_NOTIF_DATA_ERROR_INDICATION
{
    TBX_MSG_HEADER                Header;
    TBX_UINT32                    un32MsgVersion;
    TB640_SS7_SCCP_USERPART_HANDLE hSccpUserpart;

    TB640_SS7_POINT_CODE          Opc;
    TB640_SS7_SCCP_UNIT_DATA      UnitData;
    TB640_SS7_SCCP_RETURN_CAUSE   ReturnCause;
    TBX_UINT32                    un32FrameSize;
    TBX_UINT8                     aun8Payload [1];
} TB640_EVT_SS7_SCCP_NOTIF_DATA_ERROR_INDICATION;
*PTB640_EVT_SS7_SCCP_NOTIF_DATA_ERROR_INDICATION;
```

General explanation of the field of event notification data error indication:

- The handle SCCP userpart parameter (*hSccpUserpart*) specifies the handle of the SCCP userpart.
- The OPC parameter (*Opc*) specifies the originating point code received from the MTP3 routing label.
- The unit data (*UnitData*) parameter. For description see section 7.6.1 Data Request.
- The return cause parameter (*ReturnCause*) is used to indicate the reason why a message was not able to be delivered to its final destination. Possible values:

Table 124 - SCCP return cause

Return Cause	Description
TB640_SS7_SCCP_RETURN_CAUSE_NO_TRANSLATION_AVAIL	No translation, address of such nature.
TB640_SS7_SCCP_RETURN_CAUSE_UNKNOW_ADDRESS	No translation, specific address.
TB640_SS7_SCCP_RETURN_CAUSE_SUBSYSTEM_CONGESTION	Subsystem congestion.
TB640_SS7_SCCP_RETURN_CAUSE_SUBSYSTEM_FAILURE	Subsystem failure.
TB640_SS7_SCCP_RETURN_CAUSE_UNEQUIPPED	Unequipped user.
TB640_SS7_SCCP_RETURN_CAUSE_NETWORK_FAILURE	Network Failure.
TB640_SS7_SCCP_RETURN_CAUSE_NETWORK_CONGESTION	Network Congestion.
TB640_SS7_SCCP_RETURN_CAUSE_UNQUALIFIED	Unqualified.

TB640_SS7_SCCP_RETURN_CAUSE_HOP_VIOLATION	Hop counter violation (valid in ANS92).
TB640_SS7_SCCP_RETURN_CAUSE_ERR_MSG_TRANSPORT	Error in message transport (valid in ITU92 & ANS96).
TB640_SS7_SCCP_RETURN_CAUSE_ERR_LOCAL_PROCESSING	Error in local processing (valid in ITU92 & ANS96).
TB640_SS7_SCCP_RETURN_CAUSE_ERR_REASSEMBLY	Destination cannot perform reassembly (valid in ITU92 & ANS96).
TB640_SS7_SCCP_RETURN_CAUSE_SCCP_FAILURE	SCCP failure (valid in ITU92).
TB640_SS7_SCCP_RETURN_CAUSE_HOP_VIOLATION2	Hop counter violation.
TB640_SS7_SCCP_RETURN_CAUSE_SEGMENT_NOT_SUPPORTED	Segmentation not supported.
TB640_SS7_SCCP_RETURN_CAUSE_SEGMENT_FAILURE	Segmentation failure.
TB640_SS7_SCCP_RETURN_CAUSE_MSG_CHANGE_FAILURE	Message change failure.
TB640_SS7_SCCP_RETURN_CAUSE_INVALID_INS_REQUEST	Invalid INS routing request.
TB640_SS7_SCCP_RETURN_CAUSE_INVALID_ISNI_REQUEST	Invalid ISNI routing request.
TB640_SS7_SCCP_RETURN_CAUSE_UNAUTHORIZED_MSG	Unauthorized message.
TB640_SS7_SCCP_RETURN_CAUSE_MSG_INCOMPATIBILITY	Message incompatibility.
TB640_SS7_SCCP_RETURN_CAUSE_ISNI_ROUTING_ERR	Cannot perform ISNI constrained routing.
TB640_SS7_SCCP_RETURN_CAUSE_ISNI_REDUNDANT	Redundant ISNI constrained routing information.
TB640_SS7_SCCP_RETURN_CAUSE_ISNI_IDENTIFICATION_FAIL	Unable to perform ISNI identification.

- The frame size parameter (*un32FrameSize*) specifies the byte count for this frame in the *aun8Payload* parameter.
- The payload parameter (*aun8Payload*) contains the raw payload data of the received frame.

7.6.8 Coordinate Indication

The TB640_MSG_ID_SS7_SCCP_NOTIF_COOR_INDICATION (event) message is used to send a coordinated state change messages from a SCCP Userpart to the host application. This notification message is used to indicate that an upper service, not co-located, has requested permission to go out-of-service. If the request is granted by the upper service, the upper service responds with the TB640_MSG_ID_SS7_SCCP_OP_COOR_RESPONSE message. If the request is not granted, no response is returned to the upper service, implicitly denying the request.

Structure contains the **event** notification coordinate indication from a SCCP Userpart:

```
typedef struct _TB640_EVT_SS7_SCCP_NOTIF_COOR_INDICATION
{
    TBX_MSG_HEADER                Header;
    TBX_UINT32                    un32MsgVersion;
    TB640_SS7_SCCP_USERPART_HANDLE hSccpUserpart;

    TBX_UINT32                    un32SSN;
    TB640_SS7_SCCP_SMI_TYPE       Smi;
} TB640_EVT_SS7_SCCP_NOTIF_COOR_INDICATION, *PTB640_EVT_SS7_SCCP_NOTIF_COOR_INDICATION;
```

General explanations of the fields of event notification coordinate indication:

- The handle SCCP userpart parameter (*hSccpUserpart*) specifies the handle of the SCCP userpart.

- The subsystem number parameter (*un32SSN*) specifies a subsystem number. The affected subsystem number is used to indicate the subsystem requesting a coordinated state change. The affected subsystem is decoded from the affected SSN field in the SCCP subsystem out-of-service request message. Possible values are from **1** to **255**. See **Table 93 - SCCP SSN value to identify SCCP user**.
- The subsystem multiplicity indicator parameter (*Smi*) specifies a subsystem multiplicity indicator. The subsystem multiplicity indicator is decoded from the subsystem multiplicity indicator field of the SCCP subsystem out-of-service request message. Possible values:

Table 125 - SCCP SMI type

Subsystem Multiplicity Indicator Type	Description
TB640_SS7_SCCP_SMI_TYPE_UNKNOW	Affected subsystem multiplicity unknown.
TB640_SS7_SCCP_SMI_TYPE_NO_BACKUP	Affected subsystem is solitary.
TB640_SS7_SCCP_SMI_TYPE_BACKUP_ROUTE_AVAILABLE	Affected subsystem is duplicated.
TB640_SS7_SCCP_SMI_TYPE_DENIED	Indicates to service users that the coordinated state change has been denied. Give explicit OOS request denial if the route is ANSI92, ANSI96 or TELCORDIA.

7.6.9 Coordinate Confirmation

The TB640_MSG_ID_SS7_SCCP_NOTIF_COOR_CONFIRM (event) message is used to send a response of a coordinated state change request messages from a SCCP Userpart to the host application. This notification message acknowledges a coordinated state change request.

Structure contains the **event** notification coordinate indication from a SCCP Userpart:

```
typedef struct _TB640_EVT_SS7_SCCP_NOTIF_COOR_CONFIRM
{
    TBX_MSG_HEADER                Header;
    TBX_UINT32                    un32MsgVersion;
    TB640_SS7_SCCP_USERPART_HANDLE hSccpUserpart;

    TBX_UINT32                    un32SSN;
    TB640_SS7_SCCP_SMI_TYPE      Smi;
} TB640_EVT_SS7_SCCP_NOTIF_COOR_CONFIRM, *PTB640_EVT_SS7_SCCP_NOTIF_COOR_CONFIRM;
```

General explanations of the fields of event notification coordinate confirmation:

- The handle SCCP userpart parameter (*hSccpUserpart*) specifies the handle of the SCCP userpart.
- The subsystem number parameter (*un32SSN*) specifies a subsystem number. The affected subsystem number is used to indicate the subsystem requesting a coordinated state change. The affected subsystem is decoded from the affected SSN field in the SCCP subsystem out-of-service grant message. Possible values are from **1** to **255**. See **Table 93 - SCCP SSN value to identify SCCP user**.
- The subsystem multiplicity indicator parameter (*Smi*) specifies a subsystem multiplicity indicator. The subsystem multiplicity indicator is decoded from the subsystem

multiplicity indicator field of the SCCP subsystem out-of-service grant message. Possible values: see **Table 125 - SCCP SMI type**.

7.6.10 Point Code SSN Status Confirmation

The TB640_MSG_ID_SS7_SCCP_NOTIF_PC_SSN_STATUS_CFM (event) message is used to send a response of a PC / SSN status request messages from a SCCP Userpart to the host application. This notification message acknowledges a PC/SSN status request from upper layer.

Structure contains the **event** notification PC/SSN status confirmation from a SCCP Userpart:

```
typedef struct _TB640_EVT_SS7_SCCP_NOTIF_PC_SSN_STATUS_CFM
{
    TBX_MSG_HEADER                Header;
    TBX_UINT32                    un32MsgVersion;
    TB640_SS7_SCCP_USERPART_HANDLE hSccpUserpart;

    TB640_SS7_SCCP_REQUEST_STATUS_TYPE RequestType;
    TBX_UINT32                          un32SSN;
    TB640_SS7_POINT_CODE                Dpc;
    TB640_SS7_SCCP_STATUS_TYPE          Status;
    TB640_SS7_SCCP_REMOTE_STATUS_TYPE   RemoteStatus;
    TB640_SS7_SCCP_SMI_TYPE             Smi;
    TBX_UINT32                          un32Ril;
} TB640_EVT_SS7_SCCP_NOTIF_PC_SSN_STATUS_CFM, *PTB640_EVT_SS7_SCCP_NOTIF_PC_SSN_STATUS_CFM;
```

General explanations of the fields of event notification PC/SSN status confirmation:

- The handle SCCP userpart parameter (*hSccpUserpart*) specifies the handle of the SCCP userpart.
- The request type parameter (*RequestType*) specifies the status request type. Possible values: see **Table 122 - SCCP request status type**.
- The subsystem number parameter (*un32SSN*) specifies subsystem number whose status is requested. Possible values are from **1** to **255**. See **Table 93 - SCCP SSN value to identify SCCP user**.
- The DPC parameter (*Dpc*) specifies the destination point code number whose status is requested.
- The status parameter (*Status*) indicates the status of the point code or subsystem as requested by the service user. Possible values: see **Table 123 - SCCP status type**.
- The remote status parameter (*RemoteStatus*) indicates the status of the remote SCCP. Possible values:

Table 126 - SCCP remote status type

Remote Status Type	Description
TB640_SS7_SCCP_REMOTE_STATUS_TYPE_AVAILABLE	Remote SCCP available.
TB640_SS7_SCCP_REMOTE_STATUS_TYPE_UNAVAILABLE	Remote SCCP unavailable, reason unknown.
TB640_SS7_SCCP_REMOTE_STATUS_TYPE_UNEQUIPPED	Remote SCCP unequipped.
TB640_SS7_SCCP_REMOTE_STATUS_TYPE_INACCESSIBLE	Remote SCCP inaccessible.
TB640_SS7_SCCP_REMOTE_STATUS_TYPE_CONGESTED	Remote SCCP congested.

- ✍ This parameter is relevant only in **ITU96** implementation.
- The subsystem multiplicity indicator parameter (*Smi*) specifies a subsystem multiplicity indicator. Possible values: see **Table 125 - SCCP SMI type**.
 - ✍ Excepted TB640_SS7_SCCP_SMI_TYPE_DENIED.
- The restricted importance level parameter (*un32Ril*) is used to indicate the user that any message for the affected **aDpc** having message importance value below this level shall be discarded by SCCP. A message with the importance value same as Restriction Level (**RL**), may or may not be discarded depending upon the value of Restriction Sub-level (**RSL**) maintained for that destination. Possible values of *un32Ril* are in the range **0** to **7**. This parameter is relevant **only in ITU96** implementation and is used if traffic limitation mechanism is set *fUseTrafficLimit* in the general configuration.

7.6.11 SSN State Change Indication

The TB640_MSG_ID_SS7_SCCP_NOTIF_SSN_STATE_CHANGE_IND (event) message is used to inform the state of the service at a concerned point code.

Structure contains the **event** notification SSN state change indication from a SCCP Userpart:

```
typedef struct _TB640_EVT_SS7_SCCP_NOTIF_SSN_STATE_CHANGE_IND
{
    TBX_MSG_HEADER                Header;
    TBX_UINT32                    un32MsgVersion;
    TB640_SS7_SCCP_USERPART_HANDLE hSccpUserpart;

    TB640_SS7_POINT_CODE          Dpc;
    TBX_UINT32                    un32SSN;
    TB640_SS7_SCCP_STATUS_TYPE    Status;
    TB640_SS7_SCCP_SMI_TYPE       Smi;
    TBX_UINT8                     aun8Padding0 [4];
} TB640_EVT_SS7_SCCP_NOTIF_SSN_STATE_CHANGE_IND;
*PTB640_EVT_SS7_SCCP_NOTIF_SSN_STATE_CHANGE_IND;
```

General explanations of the fields of event notification SSN state change indication:

- The handle SCCP userpart parameter (*hSccpUserpart*) specifies the handle of the SCCP userpart.
- The DPC parameter (*Dpc*) specifies the affected destination point code.
- The subsystem number parameter (*un32SSN*) specifies a subsystem number. The affected subsystem number identifies the affected Userpart. Possible values are from **1** to **255**. See **Table 93 - SCCP SSN value to identify SCCP user**.
- The status parameter (*Status*) indicates the status of the service user. Possible values: **TB640_SS7_SCCP_STATUS_TYPE_OUT_OF_SERVICE** OR **TB640_SS7_SCCP_STATUS_TYPE_IN_SERVICE**. See **Table 123 - SCCP status type**.
- The subsystem multiplicity indicator parameter (*Smi*) specifies a subsystem multiplicity indicator. The subsystem multiplicity indicator is decoded from the subsystem

multiplicity indicator field of the SCCP subsystem out-of-service request message. Possible values: see **Table 125 - SCCP SMI type**.

7.6.12 SSN State Change Confirmation

The TB640_MSG_ID_SS7_SCCP_NOTIF_SSN_STATE_CHANGE_CFM (event) message is used to send a response of a state change request messages from a SCCP Userpart to the host application. This notification message acknowledges a state change request.

Structure contains the **event** notification SSN state change confirmation from a SCCP Userpart:

```
typedef struct _TB640_EVT_SS7_SCCP_NOTIF_SSN_STATE_CHANGE_CFM
{
    TBX_MSG_HEADER                Header;
    TBX_UINT32                    un32MsgVersion;
    TB640_SS7_SCCP_USERPART_HANDLE hSccpUserpart;

    TBX_BOOL                      fResultOk;
    TBX_UINT8                     aun8Padding0 [4];
} TB640_EVT_SS7_SCCP_NOTIF_SSN_STATE_CHANGE_CFM,
*PTB640_EVT_SS7_SCCP_NOTIF_SSN_STATE_CHANGE_CFM;
```

General explanations of the fields of event notification SSN state change confirmation:

- The handle SCCP userpart parameter (*hSccpUserpart*) specifies the handle of the SCCP userpart.
- The result ok flag parameter (*fResultOk*) specifies a success (TBX_TRUE) or a failure (TBX_FALSE). Possible values: **TBX_TRUE** or **TBX_FALSE**.

7.6.13 PC State Change Indication

The TB640_MSG_ID_SS7_SCCP_NOTIF_PC_STATE_CHANGE_IND (event) message is used to inform the state changed of a remote point code. This notification message is used to inform the service user about the state change of a concerned point code.

Structure contains the **event** notification PC state change indication from a SCCP Userpart:

```
typedef struct _TB640_EVT_SS7_SCCP_NOTIF_PC_STATE_CHANGE_IND
{
    TBX_MSG_HEADER                Header;
    TBX_UINT32                    un32MsgVersion;
    TB640_SS7_SCCP_USERPART_HANDLE hSccpUserpart;

    TB640_SS7_POINT_CODE          Dpc;
    TB640_SS7_SCCP_STATUS_TYPE    Status;
    TB640_SS7_SCCP_REMOTE_STATUS_TYPE RemoteStatus;
    TBX_UINT32                    un32Ril;
    TBX_UINT8                     aun8Padding0 [4];
} TB640_EVT_SS7_SCCP_NOTIF_PC_STATE_CHANGE_IND,
*PTB640_EVT_SS7_SCCP_NOTIF_PC_STATE_CHANGE_IND;
```

General explanations of the fields of event notification PC state change indication:

- The handle SCCP userpart parameter (*hSccpUserpart*) specifies the handle of the SCCP userpart.
- The DPC parameter (*Dpc*) specifies the affected destination point code.

- The status parameter (*Status*) indicates the status of the service user. Possible values: `TB640_SS7_SCCP_STATUS_TYPE_ACCESSIBLE` OR `TB640_SS7_SCCP_STATUS_TYPE_INACCESSIBLE` OR `TB640_SS7_SCCP_STATUS_TYPE_CONGESTED`. See **Table 123 - SCCP status type**.
- The remote status parameter (*RemoteStatus*) indicates the status of the remote SCCP. Possible values: see **Table 126 - SCCP remote status type**.
 - ✍ This parameter is relevant only in **ITU96** implementation.
- The restricted importance level parameter (*un32Ril*) is used to indicate the user that any message for the affected **aDpc** having message importance value below this level shall be discarded by SCCP. A message with the importance value same as Restriction Level (**RL**), may or may not be discarded depending upon the value of Restriction Sub-level (**RSL**) maintained for that destination. Possible values of *un32Ril* are in the range **0** to **7**. This parameter is relevant **only in ITU96** implementation.

8 TCAP

8.1 Overview

This section gives a description of the TB640 Transaction Capabilities Application Part (TCAP) layer architecture and usage. This layer is referred to as TCAP in the rest of the section.

8.1.1 Summary

The TCAP layer provides communication capabilities for interactive applications in a distributed environment. TCAP originated from ITU Recommendations X.219 and X.229 (ROSE, Remote Operation Service Element). TCAP layer uses transport services provided by SCCP to implement transaction capabilities application services. TCAP defines the end-to-end protocol between TCAP users, which may be located in a SS7 network or in another network supporting TCAP protocol.

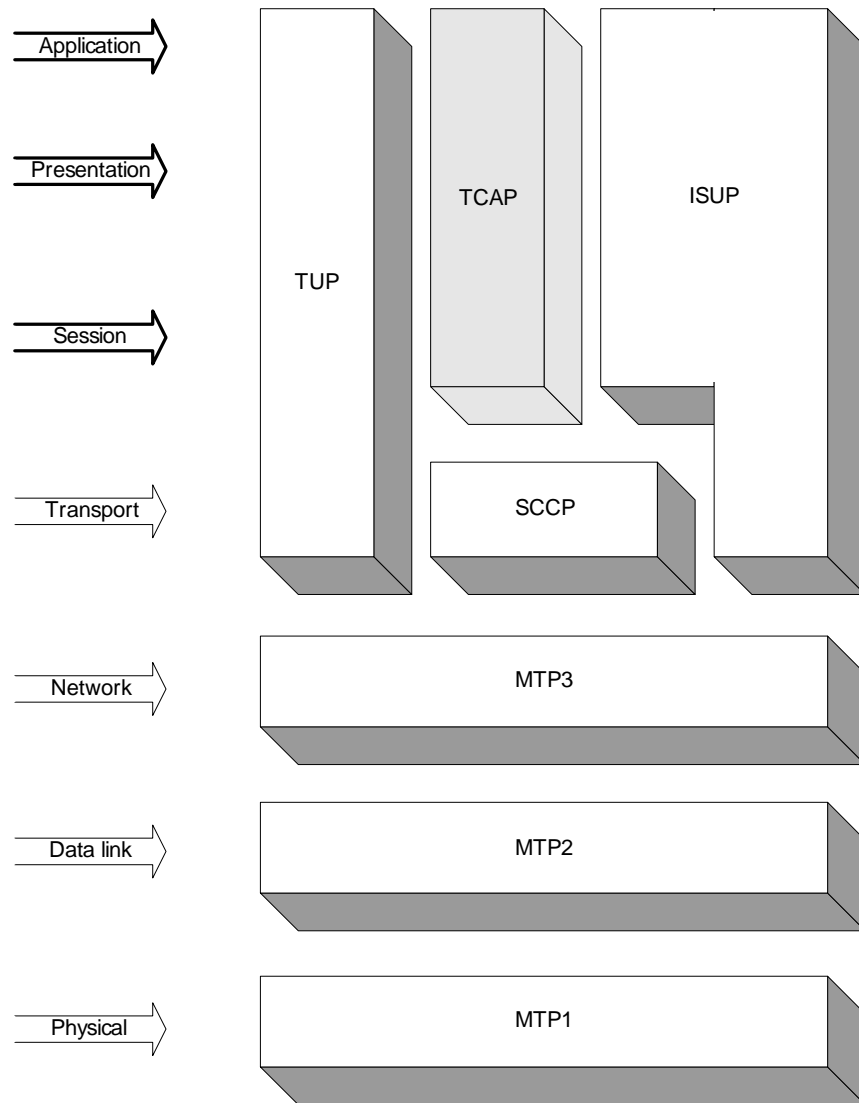


Figure 23 - TCAP OSI model

8.1.2 Features

TCAP supports the following features:

- Single transaction/dialogue identifier specified by service user;
- Simultaneous dialogues per application/subsystem number;
- Components (operations and their replies) and parameters assembly into TCAP messages;
- Dialogue and user information assembly into TCAP messages;
- Association of operations and their replies;
- Abnormal situations handling;
- Optional operations/replies timers (ITU);
- Unstructured dialogue to send one or more components that do not expect replies;
- Structured dialogue to send one or more components that expect replies;
- Structured dialogue supporting the following transaction message types: begin, continuation, end, query with/without permission, conversation with/without permission, response, abort and exception reporting;
- All classes of structured dialogue operation (class 1: both success and failure are reported, class 2: Only failure is reported, class 3: Only success is reported and class 4: Neither success nor failure is reported);
- Statistics information to determine traffic loads and quality-of-service;
- Point code/subsystem number and/or global title addressing capabilities while using SCCP transport services;
- Point code/subsystem states change indications while using SCCP transport services.

8.1.3 Architecture

TCAP software supports only a connectionless network service. No functionality is needed from the ISP (Intermediate Service Part); only the connectionless functionality of SCCP is required. Therefore, the transaction sublayer always interfaces with SCCP through the connectionless primitives.

TCAP is intended to be used by real-time, sensitive applications (with small amounts of transferred data). The data transfer rate capability is limited by MTP2 links bandwidth. Refer to section 4.2.1.2 to find how to configure MTP2 links to increase MTP2 links bandwidth by using multiple timeslots by link or HSL (High Speed Link) links. Each timeslot could transfer a maximum of 64KBPS. In multiple timeslots mode, it is possible to transfer a maximum of 256KBPS by MTP2 link. When using HSL, it is possible to transfer a maximum of 1984KBPS by MTP2 link. Thus the data transfer rate capability depends on the available MTP2 links and its configuration. The application real-time requirement and available MTP2 links bandwidth will mainly determine the maximum TCAP message size that could be used.

Each TCAP instance can handle a maximum of 8 userparts, 80000 dialogs and 80000 components. Each TCAP userpart can handle a maximum of 10000 dialogs and 10000 components. These limits could be modified if required by TelcoBridges upon request.

TCAP provides the functions and procedures to create dialogues between applications and invoke application operations across an SS7 network. TCAP is a service provider to MAP (Mobile

Application Part) and INAP (Intelligent Network Application Protocol) and a service user of SCCP (as shown on Figure 24 - TCAP layer organization within TB640).

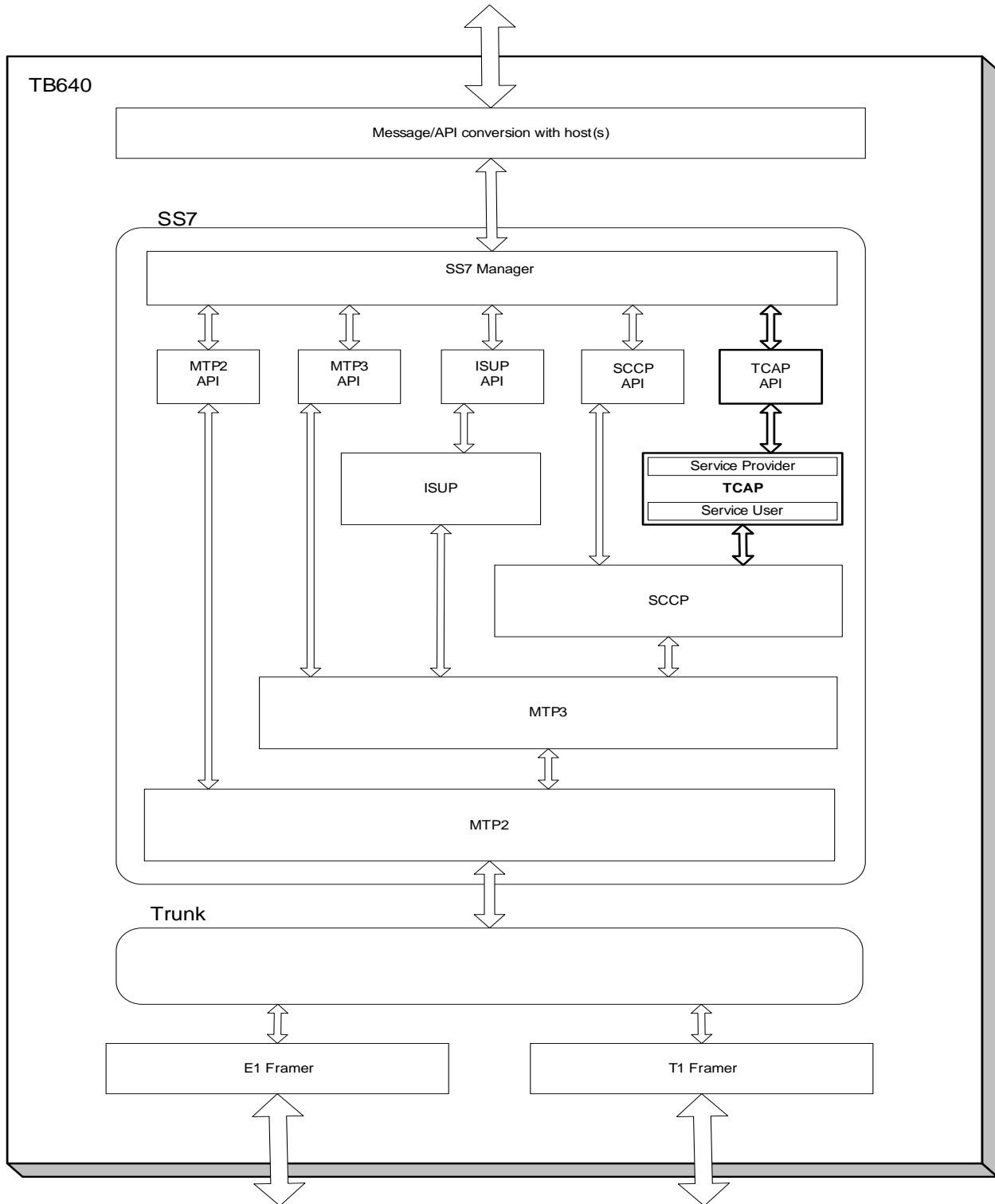


Figure 24 - TCAP layer organization within TB640

A TCAP layer has userpart sections which represents a subsystem for a specific protocol variant. TCAP userparts are extension of SCCP userparts.

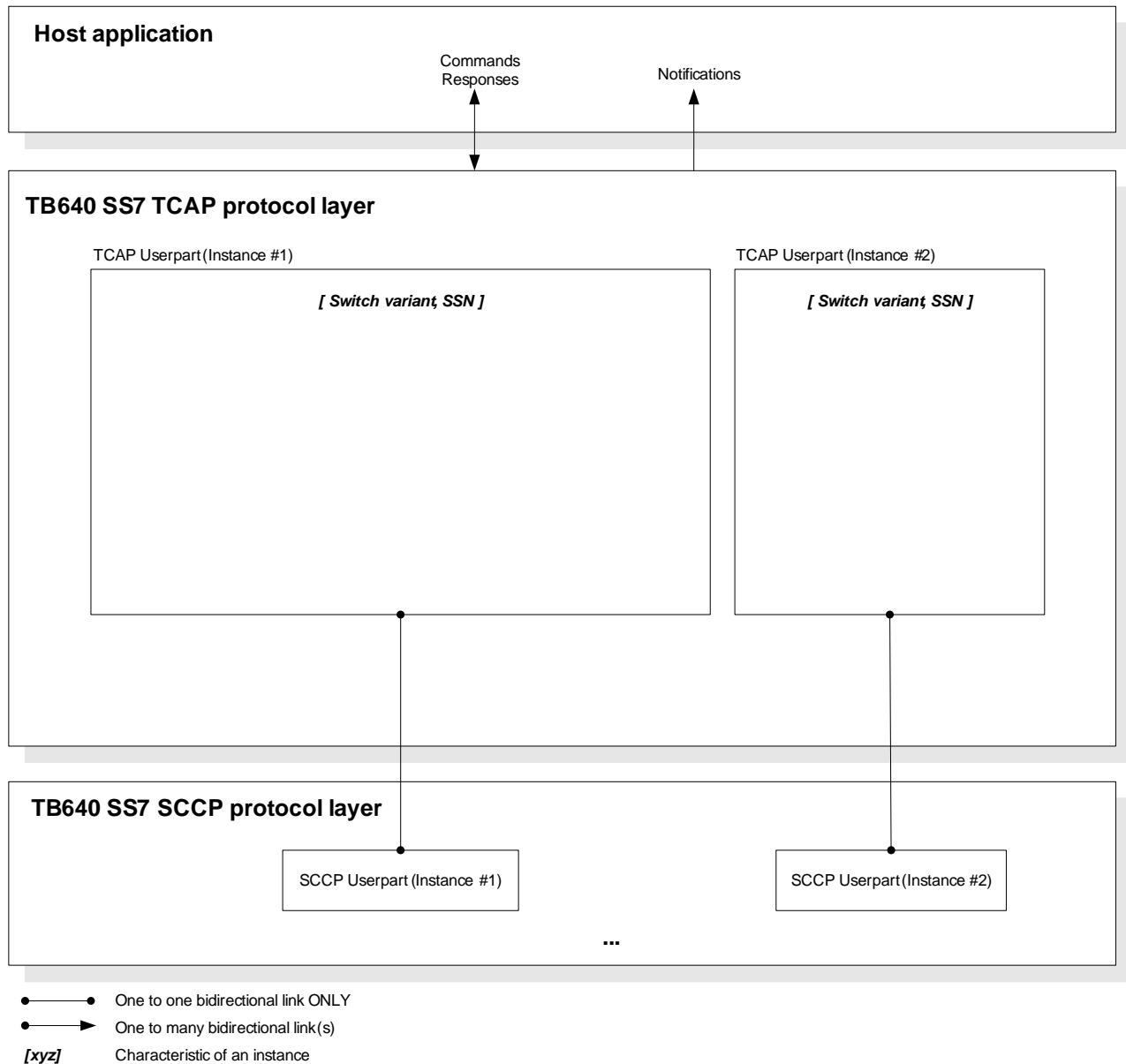


Figure 25 - TCAP layer hierarchy

TCAP userpart must be associated with one SCCP Userpart. It's a 1 to 1 association.

TCAP userpart can be used in standalone mode to serve remote TCAP service users like MAP (Mobile Application Part) or INAP (Intelligent Network Application Protocol).

A TCAP layer has a userpart section which represents a subsystem number.

8.1.3.1 Message

A TCAP message consists of three portions: the transaction portion (which contains the message type and parameters), the dialogue portion (which contains the application context name), and the component portion.

8.1.3.1.1 *Transaction portion*

These types of messages can be received or transmitted by the transaction portion of TCAP across the SCCP transport service:

ITU, ETSI:

- Unidirectional
- Begin
- Continue
- End
- Abort

ANSI:

- Unidirectional
- Query with permission
- Query without permission
- Conversation with permission
- Conversation without permission
- Response
- Abort

All TCAP messages except unidirectional message are implied for structured dialogue. Unidirectional message is implied for unstructured dialogue.

8.1.3.1.2 *Dialogue portion*

The dialogue portion is optional in a TCAP message and can only be present in 1992, 1996 ITU TCAP, 1996 ETSI TCAP, and 1996 ANSI TCAP messages. The dialogue portion is used to exchange the application context name and user-specific information.

8.1.3.1.3 *Component portion*

The component portion is mandatory in 1988 ITU TCAP, and 1988, 1992 ANSI TCAP messages, and is optional in other supported versions of TCAP. The following protocol data units (PDUs) can be received or transmitted in the component portion of a TCAP message:

ITU, ETSI:

- Invoke
- Return result not last
- Return result last
- Return error

- Reject

ANSI:

- Invoke not last
- Invoke last
- Return result not last
- Return result last
- Return error
- Reject

8.1.4 Specification

The TCAP software conforms to the following standards:

- *ETS 300 287-1 Transaction Capabilities, Version 2*, ETSI 1996
- *JT Q.771 Functional Description of Transaction Capabilities*, TTC (Japan)
- *JT Q.772 Transaction Capabilities Information Element Definitions*, TTC (Japan)
- *JT Q.773 Transaction Capabilities Formats and Encoding*, TTC (Japan)
- *JT Q.774 Transaction Capabilities Procedures*, TTC (Japan)
- *Q.752 Monitoring and Measurements of Signalling System No. 7 Networks*, 1993
- *Q.771 Functional Description of Transaction Capabilities*, ITU (88, 92, and 96)
- *Q.772 Transaction Capabilities Information Element Definitions*, ITU (88, 92, and 96)
- *Q.773 Transaction Capabilities Formats and Encoding*, ITU (88, 92, and 96)
- *Q.774 Transaction Capabilities Procedures*, ITU (88, 92, and 96)
- *Q.775 Guidelines for Using Transaction Capabilities*, ITU (88, 92, and 96)
- *Q.787 Transaction Capabilities (TC) Test Specification*, 1993
- *T1.114, Signalling System Number 7 - Transaction Capability Application Part (TCAP)*, ANSI (88, 92, and 96)

TCAP can support different protocol variants:


- ITU (used for ITU88, ITU92, and ITU96)
- ANSI (used for ANSI88, ANSI92, and ANSI96)
- ETSI 1996 (*ETS 300 287-1*)

8.2 TCAP Configuration

8.2.1 Configuration of layer

General guidelines for configuration of TCAP for a proper operation include the following:

1. The TCAP general allocation⁹ (TB640_MSG_ID_SS7_TCAP_OP_ALLOC) must precede all other messages (other configuration TCAP alloc, get, states and stats). The response of this message is a TCAP handle.
2. The TCAP userpart allocation (TB640_MSG_ID_SS7_TCAP_OP_USERPART_ALLOC) must be made (with the TCAP handle from **step 1**) for a particular subsystem number. The response of this message is a TCAP userpart handle.

 For any reconfiguration with a TCAP (TB640_MSG_ID_SS7_TCAP_OP_xyz_SET_PARAMS) message, all reconfigurable parameters must be filled appropriately even if the intention is to modify a single parameter.

 The **step 2** must be repeated for each userpart to configure.

8.2.1.1 General configuration

The TB640_MSG_ID_SS7_TCAP_OP_ALLOC (request/response) message is used to initialize the general parameters of the TCAP layer.

The **request** part of the message TB640_MSG_ID_SS7_TCAP_OP_ALLOC contains the following fields:

```

...
TB640_SS7_HANDLE      hLayer; /* Contains the layer handle from system manager module */
TB640_SS7_TCAP_CFG    Cfg;      /* Contains the configuration of the TCAP layer */
...

```

The following structure contains the general configuration parameters of TCAP layer:

```

typedef struct _TB640_SS7_TCAP_CFG
{
    TBX_UINT32      un32StructVersion;
    TBX_BOOL        fReturnOnError;
} TB640_SS7_TCAP_CFG, *PTB640_SS7_TCAP_CFG;

```

General explanation of the fields of previous structure:

- The structure version parameter (*un32StructVersion* field in structure TB640_SS7_TCAP_CFG), is the structure version identifier. This parameter must be set to 1.
- The return on error parameter (*fReturnOnError* field in structure TB640_SS7_TCAP_CFG), specifies whether TCAP should generate protocol messages (ABORT or REJECT) alarms (to take care of the local protocol errors). For example: receiving a data request (with message type other than BEGIN) for which the dialogue does not already exist, or receiving a duplicate invoke ID. This parameter is reconfigurable.

⁹ All fields of a configuration message alloc must be filled unless explicitly optional or not defined for certain variants.

8.2.1.2 Userpart configuration

The TB640_MSG_ID_SS7_TCAP_OP_USERPART_ALLOC (request/response) message is used to initialize the userpart parameters of the TCAP layer.

The **request** part of the message TB640_MSG_ID_SS7_TCAP_OP_USERPART_ALLOC contains the following fields:

```

...
TB640_SS7_TCAP_HANDLE      hTcap; /* Handle to the TCAP layer */
TB640_SS7_TCAP_USERPART_CFG Cfg;    /* Contains the configuration of the TCAP
                                     * Userpart instance */
...

```

The following structure contains the configuration parameters of TCAP userpart instance:

```

typedef struct _TB640_SS7_TCAP_USERPART_CFG
{
    TBX_UINT32          un32StructVersion;
    TB640_SS7_UID      UidTcapUserpart;
    TB640_SS7_UID      UidSccpUserpart;
    TBX_UINT8          un8SSN;
    TBX_UINT8          aun8Padding0[3];

    TB640_SS7_TCAP_PROTOCOL_VARIANT ProtocolVariant;
    TBX_UINT32          un32T1Timer;
    TBX_UINT32          un32T2Timer;
    TBX_UINT32          un32LostConnTimer;
} TB640_SS7_TCAP_USERPART_CFG, *PTB640_SS7_TCAP_USERPART_CFG;

```

General explanation of the fields of previous structure:

- The structure version parameter (*un32StructVersion* field in structure TB640_SS7_TCAP_CFG), is the structure version identifier. This parameter must be set to 1.
- The unique identifier TCAP Userpart parameter (*UidTcapUserpart*) specifies the unique ID of the TCAP Userpart for a system SS7. This parameter is not reconfigurable.
- The unique identifier SCCP Userpart parameter (*UidSccpUserpart*) specifies the unique ID of the SCCP Userpart for a system SS7. This parameter is not reconfigurable.
- The subsystem number parameter (*un8SSN*) specifies a subsystem number for this userpart. This field is not reconfigurable. Allowable values are from 1 to 255. See **Table 93 - SCCP SSN value to identify SCCP user.**
- The protocol variant parameter (*ProtocolVariant*) specifies the TCAP protocol variant required to provide service to the userpart connected through this upper SAP (Service Access Point). This parameter is not reconfigurable. Allowable values:

Table 127 - TCAP protocol variant

Protocol Variant	Description
TB640_SS7_TCAP_PROTOCOL_VARIANT_ANSI88	Used for ANSI 1988.
TB640_SS7_TCAP_PROTOCOL_VARIANT_ANSI92	Used for ANSI 1992.

TB640_SS7_TCAP_PROTOCOL_VARIANT_ANSI96	Used fro ANSI 1996.
TB640_SS7_TCAP_PROTOCOL_VARIANT_ITU88	Used for ITU 1988.
TB640_SS7_TCAP_PROTOCOL_VARIANT_ITU92	Used for ITU 1992.
TB640_SS7_TCAP_PROTOCOL_VARIANT_ITU96	Used for ITU 1996.
TB640_SS7_TCAP_PROTOCOL_VARIANT_ETSI96	Used for ETSI 1996.

- The T1 timer parameter (*un32T1Timer*) specifies the TCAP invoke timer default value. If the invoke timer value received in component request is zero, then TCAP uses the configured invoke timer default value to start the invoke timer. This field is reconfigurable. Allowable values are from 0 to 65535.

 This parameter is relevant only in **ITU** implementation.

- The T2 timer parameter (*un32T2Timer*) specifies the TCAP reject timer default value. This field is reconfigurable. Allowable values are from 0 to 65535.

 This parameter is relevant only in **ITU** implementation.

- The lost connection timer parameter (*un32LostConnTimer*) specifies the TCAP provider connection timer. This is used by TCAP provider to detect lost connections. This field is reconfigurable. Allowable values are from 0 to 65535.

The **response** part of the message TB640_MSG_ID_SS7_TCAP_OP_USERPART_ALLOC contains the following field:

```

...
TB640_SS7_TCAP_USERPART_HANDLE          hTcapUserpart;          /* The handle of the instance of
...                                     * the TCAP userpart. */

```


8.2.2 Compatibility

8.2.2.1 Variants

See section 7.2.2.1

8.3 TCAP Transaction

TCAP service user has to send TCAP message by fragments. All components of a TCAP message have to be sent individually. They are queued until TCAP receives the transaction and dialogue portions. TCAP message are sent to peer TCAP/service user on message assembly completion.

 No TCAP message is sent on the SS7 network (no transaction occurs) until a TB640_MSG_ID_SS7_TCAP_OP_TRANS_DATA_SEND message is sent by TCAP service user then received by TCAP layer.

To create a new TCAP transaction, send the TB640_MSG_ID_SS7_TCAP_OP_TRANS_INIT message to register a new transaction identifier. The transaction identifier serves while building

TCAP message to link the different fragments of a TCAP message (i.e. while adding components to TCAP message). It is not required to send the TB640_MSG_ID_SS7_TCAP_OP_TRANS_INIT message for already active outgoing and incoming transaction. Active outgoing transaction is a transaction that has been created locally and hasn't been terminated. Active incoming transaction is a transaction that has been created remotely by peer TCAP/service user and hasn't been terminated.

To add components to previously created TCAP message, use message TB640_MSG_ID_SS7_TCAP_OP_TRANS_COMPONENT_ADD.

To add transaction and dialogue portions and activate TCAP message assembling and transmission, use message TB640_MSG_ID_SS7_TCAP_OP_TRANS_DATA_SEND.

The peer application will receive TB640_MSG_ID_SS7_TCAP_OP_TRANS_DATA_RECEIVED message then TB640_MSG_ID_SS7_TCAP_OP_TRANS_COMPONENT_RECEIVED message(s). The data received indication contains transaction and dialogue portions of TCAP message. The component received indication contains component portion of TCAP message. The peer TCAP/service user will receive a component received indication for all components that are part of a TCAP message. The TCAP service user has to match the components and the transaction and dialogue portions using the transaction identifier.

8.3.1 Creating new transaction

The TB640_MSG_ID_SS7_TCAP_OP_TRANS_INIT (request/response) message is used to create and initialize a new TCAP transaction. The new transaction instance stays active until sending the transaction and dialogue portions using the following message types:

TB640_SS7_TCAP_MESSAGE_TYPE_END
TB640_SS7_TCAP_MESSAGE_TYPE_RESPONSE
TB640_SS7_TCAP_MESSAGE_TYPE_USER_ABORT
TB640_SS7_TCAP_MESSAGE_TYPE_ANSI_UABORT

Each transaction must be uniquely identified. The transaction identifier of outgoing transaction is determined by TCAP service user and must be within the range 0x80000000 to 0xFFFFFFFF. The transaction identifier of incoming transaction is determined by TCAP layer and will always be within the range 0x00000001 to 0x7FFFFFFF.

To complete the TCAP message, components and parameters must be individually added using the TB640_MSG_ID_SS7_TCAP_CMD_TRANS_COMPONENT_ADD message. Message type, called and calling addresses, quality of service, dialogue and user information... must be initialized using the TB640_MSG_ID_SS7_TCAP_OP_TRANS_DATA_SEND message. The TCAP message assembly and transmission to SCCP layer occurs only when TCAP layer receives the TB640_MSG_ID_SS7_TCAP_OP_TRANS_DATA_SEND message.

8.3.2 Adding component portion

The `TB640_MSG_ID_SS7_TCAP_OP_TRANS_COMPONENT_ADD` (request/response) message is used to add component to TCAP message. Each invoke component must have a unique invoke identifier assigned within the current transaction.

Upon receiving this request, TCAP builds the component in ASN.1 format and queues it until it gets the transaction message `TB640_MSG_ID_SS7_TCAP_OP_TRANS_DATA_SEND`. The components are queued in the order in which they are added. The components that can be queued are limited to system memory. If any mandatory information is missing in the component structure (which is required to build the component) then an alarm or a reject component with problem code `TB640_SS7_TCAP_ANSI_PROBLEM_CODE_GEN_ENCODING_FAIL` is generated (if `fReturnOnError` is set to `TBX_TRUE` as part of the general configuration).

An invoke request can be canceled by sending a component add message (for both ITU and ANSI TCAP) with component type set to `TB640_SS7_TCAP_COMPONENT_TYPE_UNKNOW` and `fCancel` flag of component structure set to `TBX_TRUE`. If the invoke component was not sent then it is to be removed from the queue and the instance of the invoke context will be deleted. If the invoke component was already sent and the instance of the invoke context exists then it would be deleted.

In the case of ANSI TCAP, there is no invoke timer maintained in TCAP (recommendation does not specify any timer). There is a possibility that a component is not received in response to an invoke component sent for operation class 2 (only failure is reported), 3 (only success is reported), and 4 (neither failure nor success is reported). The instance of the invoke context may remain allocated even after the operation is over. To resolve this situation, TCAP assumes that the TCAP user application maintains the operation timer, and upon expiry of this timer, application will generate a cancel request to TCAP (to delete the invoke instance).

The component portion contains the following parameters:

- TCAP userpart handle;
- TCAP transaction identifier;
- Component parameters;
- Component specific parameters buffer.

8.3.3 Sending transaction and dialogue portions

The `TB640_MSG_ID_SS7_TCAP_OP_TRANS_DATA_SEND` (request/response) message is used to add transaction and dialogue portions to TCAP message. This message also activates TCAP message assembly and transmission to destination signaling point (peer TCAP/service user).

The transaction portion contains the following parameters:

- TCAP userpart handle;
- TCAP transaction identifier;
- Message type identifier;

- End flag indicator;
- Calling address;
- Called address;
- Quality of service parameters;
- ISNI parameters;
- Message importance indicator.

The dialogue portion contains the following parameters:

- Dialogue parameters;
- User information buffer.

8.3.4 Receiving component portion

The TB640_MSG_ID_SS7_TCAP_NOTIF_TRANS_COMPONENT_RECEIVED (event) message is used by TCAP to notify and send received component to TCAP service user.

If TCAP detects any syntax or semantic error in the received component (for example, unexpected component type), then it builds and queues a reject component and it generates a local reject indication to the TCAP service user. Then, the TCAP service user can send TB640_MSG_ID_SS7_TCAP_OP_TRANS_COMPONENT_ADD (reject component) and TB640_MSG_ID_SS7_TCAP_OP_TRANS_DATA_SEND messages to the peer TCAP/service user. Previous statement isn't valid when an erroneous component was received in a unidirectional, end (ITU TCAP), or response (ANSI TCAP) message. When this occurs, the rejected component is not queued.

The component portion contains the following parameters:

- TCAP userpart handle;
- TCAP transaction identifier;
- Component parameters;
- Component reject source indicator;
- Component specific parameters buffer.

8.3.5 Receiving transaction and dialogue portions

The TB640_MSG_ID_SS7_TCAP_NOTIF_TRANS_DATA_RECEIVED (event) message is used by TCAP to notify and send received transaction and dialogue portions to TCAP service user.

The transaction identifier of received transaction remains valid until:

TCAP receives a message among these types and the last component of the TCAP message:

- TB640_SS7_TCAP_MESSAGE_TYPE_END
- TB640_SS7_TCAP_MESSAGE_TYPE_RESPONSE
- TB640_SS7_TCAP_MESSAGE_TYPE_USER_ABORT

- TB640_SS7_TCAP_MESSAGE_TYPE_PROVIDER_ABORT
- TB640_SS7_TCAP_MESSAGE_TYPE_ANSI_UABORT
- TB640_SS7_TCAP_MESSAGE_TYPE_ANSI_PABORT

TCAP service user sends TCAP message with one of the following message types:

- TB640_SS7_TCAP_MESSAGE_TYPE_END
- TB640_SS7_TCAP_MESSAGE_TYPE_RESPONSE
- TB640_SS7_TCAP_MESSAGE_TYPE_USER_ABORT
- TB640_SS7_TCAP_MESSAGE_TYPE_ANSI_UABORT

If TCAP detects that the received message is incorrect, it sends an abort indication to TCAP service user. The message type is set to TB640_SS7_TCAP_MESSAGE_TYPE_PROVIDER_ABORT in the case of ITU, and TB640_SS7_TCAP_MESSAGE_TYPE_ANSI_PABORT in the case of ANSI92 and 96. In the case of ANSI88, the TCAP service user is informed by sending the local reject indication using the component received message. The abort cause parameter is used to indicate the cause of abort.

The transaction portion contains the following parameters:

- TCAP userpart handle;
- TCAP transaction identifier;
- Message type identifier;
- Component present indicator;
- Calling address;
- Called address;
- Quality of service parameters;
- ISNI parameters;
- Abort cause indicator.

The dialogue portion contains the following parameters:

- Dialogue parameters;
- User information buffer.

8.3.6 Receiving transaction error indication

The TB640_MSG_ID_SS7_TCAP_NOTIF_TRANS_ERROR_INDICATION (event) message is used by TCAP to notify TCAP service user of undelivered TCAP message and the cause. This message is passed to the TCAP service user when TCAP receives a status indication from its lower layer. The cause of status indication from the lower layer is passed transparently to the TCAP service user. The lower layer uses the status indication message to indicate to TCAP the lower layer's inability to fulfill TCAP's transaction request (which is due to any protocol error or network error).

The transaction error contains the following parameters:

- TCAP userpart handle;
- TCAP transaction identifier;
- Calling address;
- Called address;
- ISNI parameters;
- Message importance indicator;
- Return cause indicator.

8.3.7 Transaction flow and scenarios

The following sections describe different transaction scenarios between two TCAP service users exchanging transaction through SS7 TCAP layer. The scenarios show the TB640 API requests and notifications that are sent and received to obtain the specific transaction flow.

To get more details about specific SS7 primitive, the application designer must refer to the appropriate SS7 specifications. With this basic view of the messages, the following sections will demonstrate basic transaction flows for simple SS7 TCAP transactions:

8.3.7.1 Successful unstructured dialogue (UNIDIRECTIONAL)

The TCAP service user uses the unstructured dialogue facility, to send the invoke components, from which it does not expect any response (operation class 4). TCAP uses the unidirectional message type for the unstructured dialogue mode.

Since the component portion is mandatory in a unidirectional message, the TCAP service user must issue one or more TB640_MSG_ID_SS7_TCAP_OP_TRANS_COMPONENT_ADD message(s) before issuing TB640_MSG_ID_SS7_TCAP_OP_TRANS_DATA_SEND message.

TCAP clears the transaction instance immediately after sending the unidirectional message because a message is not expected in response to the unidirectional message.

The data flow is:

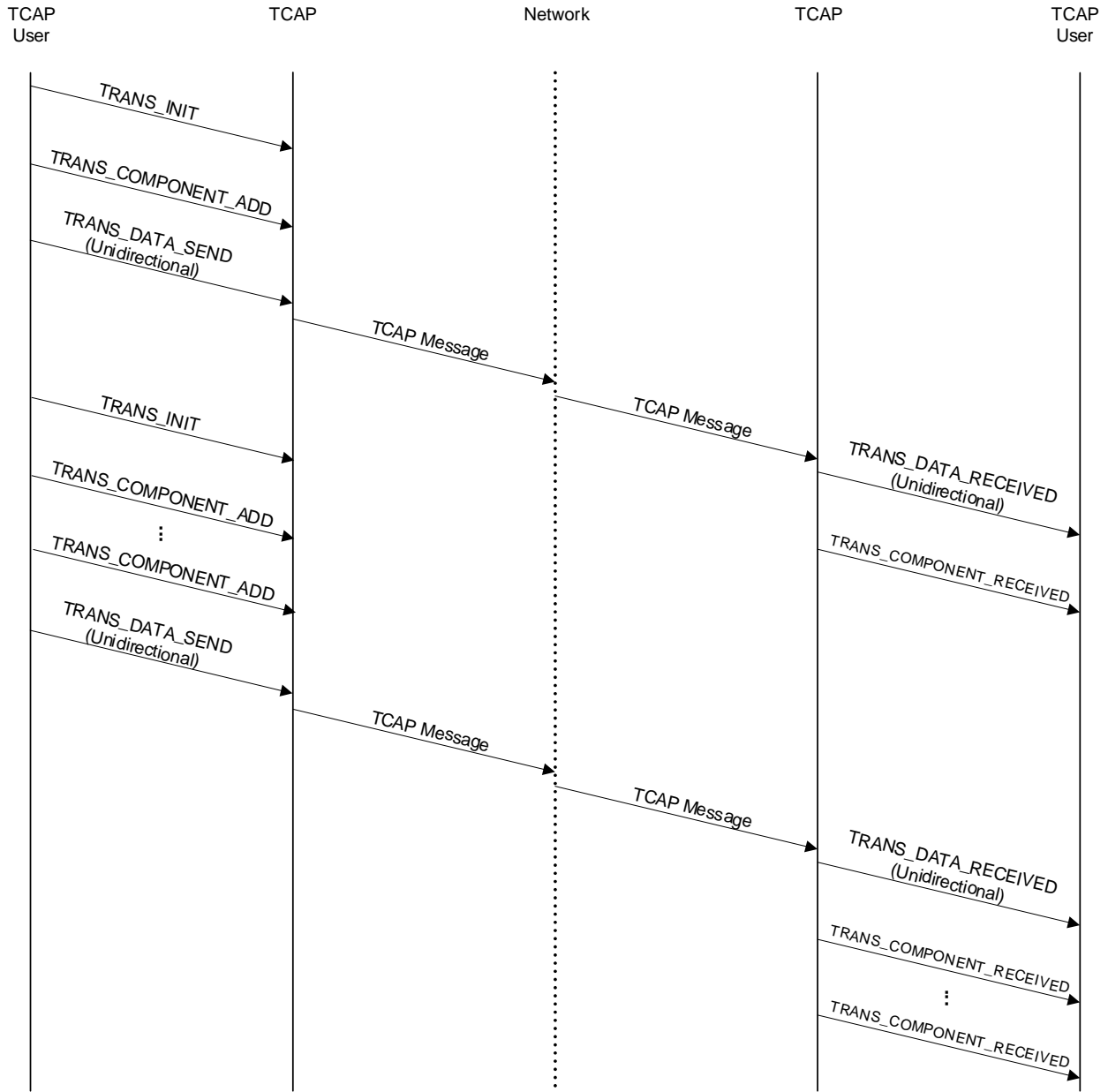


Figure 26 - TCAP successful unstructured dialogue (UNIDIRECTIONAL)

8.3.7.2 Successful structured dialogue initiation

The TCAP user initiates the structured dialogue establishment procedure to establish a dialogue, and to exchange operation invokes and replies between the peer user entities. The TCAP user sends the data transaction TB640_MSG_ID_SS7_TCAP_OP_TRANS_DATA_SEND, with the Begin/Query message type, to initiate a dialogue. If the component portion is a mandatory part of the TCAP message, the TCAP user must use the component add message TB640_MSG_ID_SS7_TCAP_OP_TRANS_COMPONENT_ADD before.

The data flow is:

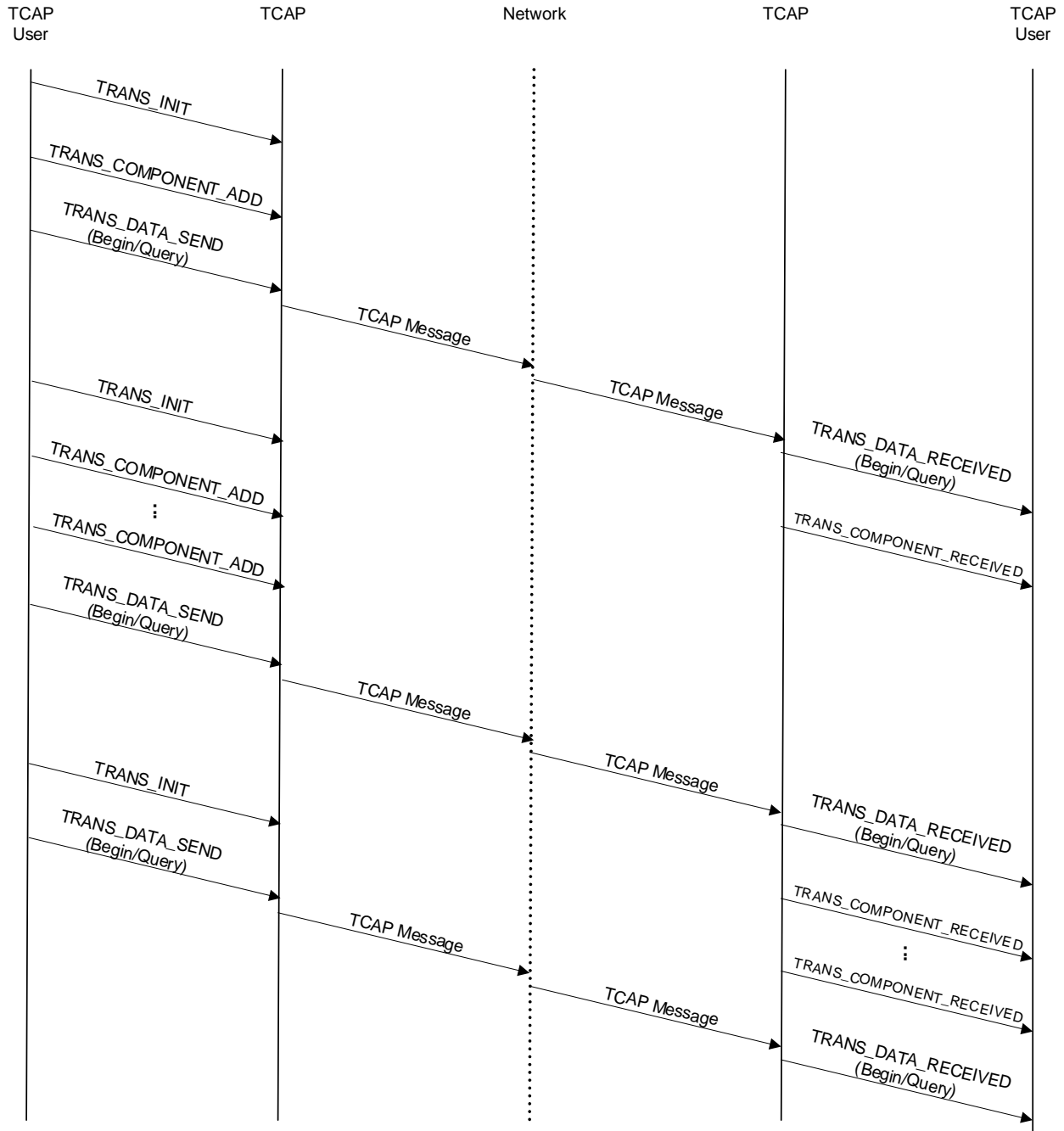


Figure 27 - TCAP successful structured dialogue initiation

8.3.7.3 Successful structured dialogue acceptance

A dialogue can be initiated after receiving a data received indication message, which carries a Begin/Query message type from the lower layer (SCCP). The TCAP service provider indicates the receipt of a dialogue initiation request (from the peer TCAP/service user) by generating a data received indication message (TB640_MSG_ID_SS7_TCAP_NOTIF_TRANS_DATA_RECEIVED); this step is followed by a component received indication message(s)

(TB640_MSG_ID_SS7_TCAP_NOTIF_TRANS_COMPONENT_RECEIVED), only if the component(s) are present in the TCAP message.

The TCAP service user indicates that it wants to continue the dialogue by issuing a TB640_MSG_ID_SS7_TCAP_OP_TRANS_DATA_SEND message with the Continue/Conversation message type. This establishes the dialogue proposed in the received Begin/Query indication message.

The data flow is:

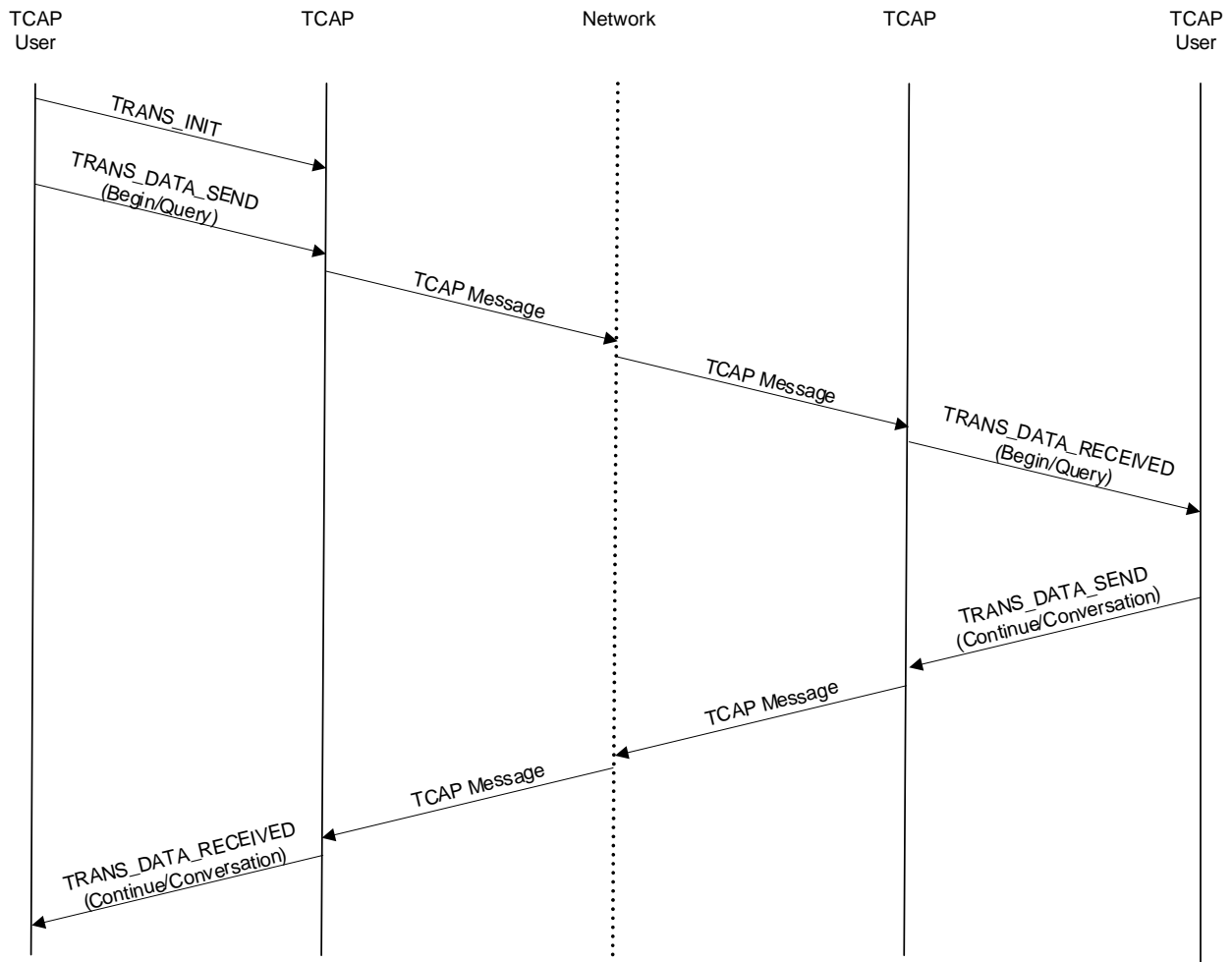


Figure 28 - TCAP successful structured dialogue acceptance

8.3.7.4 Successful structured dialogue continuation

If the dialogue initiation is acceptable to the TCAP service user, and the TCAP service user has more operation invokes and replies to exchange with the peer, the TCAP service user issues a component add message(s) (which may not be present if a component portion is optional in the TCAP message), followed by the data send message (with message type continue, or conversation w/ or w/o permission).

After receiving a unit data indication from SCCP, which carries a TCAP message of type continue or conversation, TCAP issues the data received indication with message type continue or conversation to the TCAP service user followed by component received indication (if any component is present in the message).

The sequence of continue/conversation messages continue, between the peer TCAP entities and the respective TCAP service users, until the TCAP service user decides to terminate the dialogue.

The data flow is:

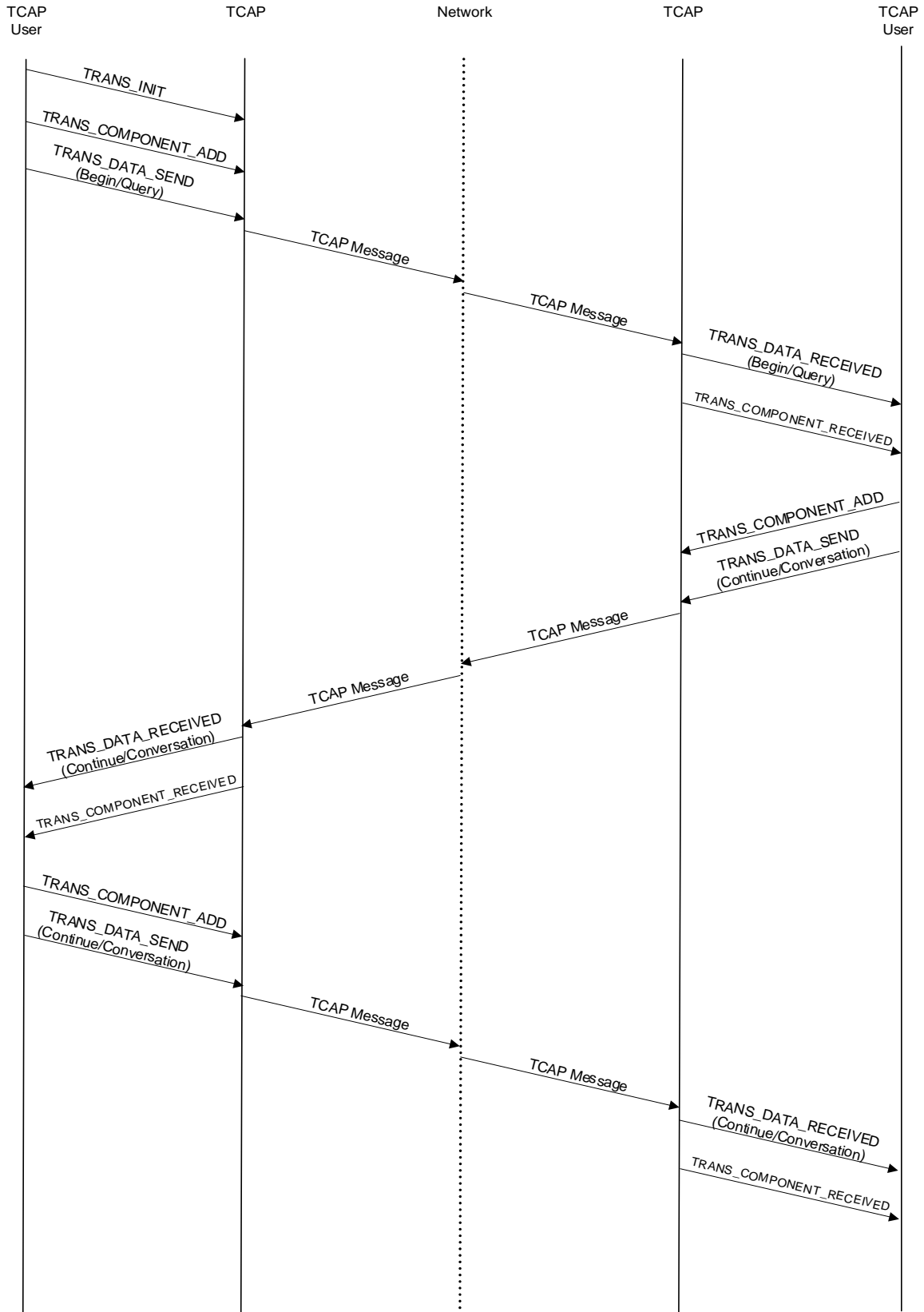


Figure 29 - TCAP successful structured dialogue continuation

8.3.7.5 Successful structured dialogue termination

The TCAP service user can terminate the dialogue by sending a data send message with message type end/response. The dialogue can end either in immediate response to begin or query w/wo permission message or after at least one continuation message is exchanged between the peer entities. The TCAP service user can end the dialogue in either of the two ways:

- a) Basic end
- b) Prearranged end

If the TCAP service user specifies in the data send message that the dialogue end should be "basic/normal" (see *fEndFlag*), TCAP transmits an end/response message. If the TCAP service user specifies in the data send message that the dialogue end should be "prearranged" (see *fEndFlag*), TCAP terminates the dialogue locally and does not transmit an end/response message.

The data flow is:

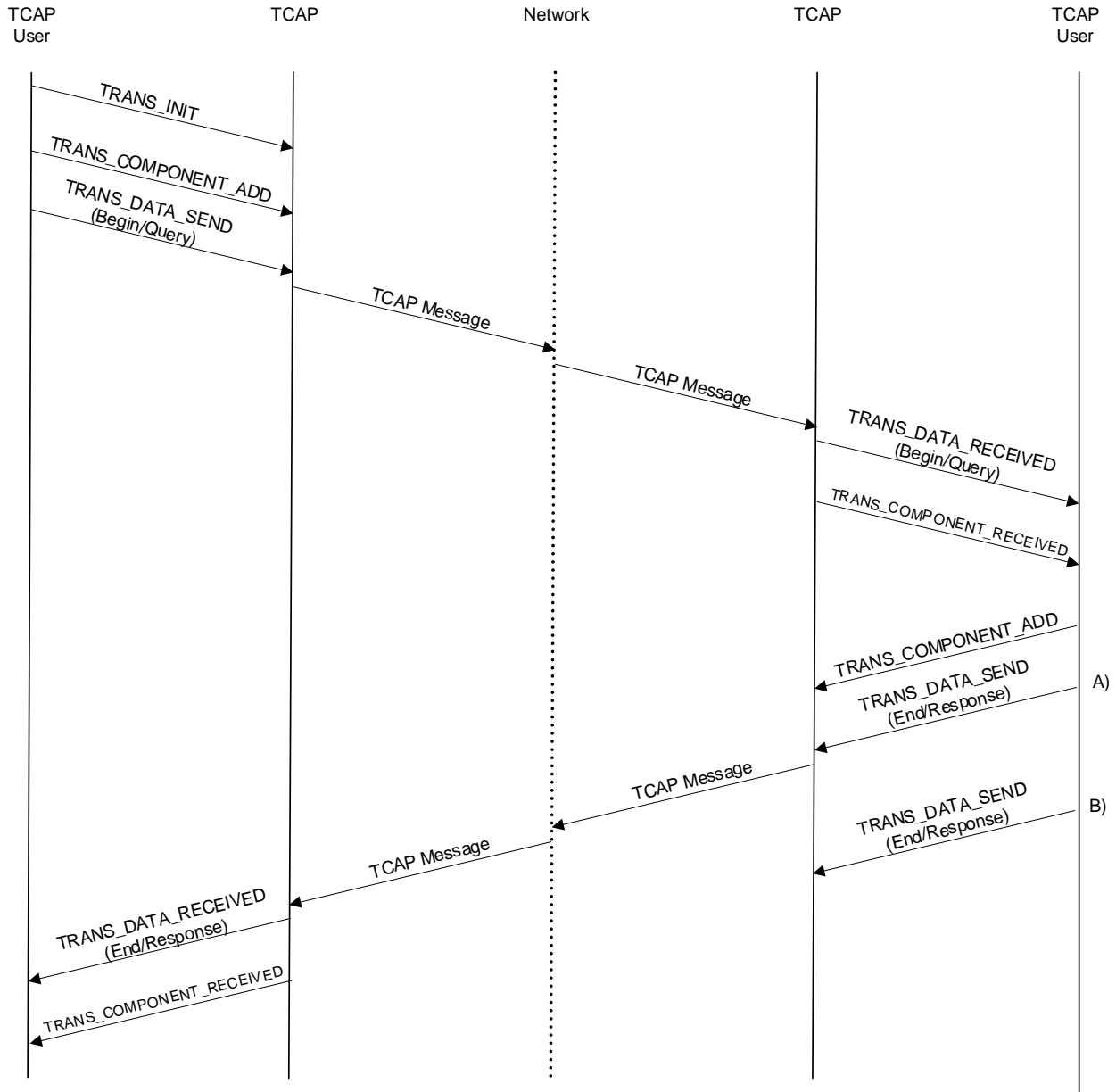


Figure 30 - TCAP successful structured dialogue termination

8.3.7.6 Successful structured dialogue abort

The TCAP service user can abort a dialogue by sending an abort request to TCAP. Upon receiving an abort request, TCAP terminates the dialogue locally and sends an abort message to the peer. The TCAP service user can abort a dialogue either in the a) dialogue establishment state (application context name not supported, or user information not acceptable), or in the b) active state of the dialogue (any syntax error or application defined error).

The data flow is:

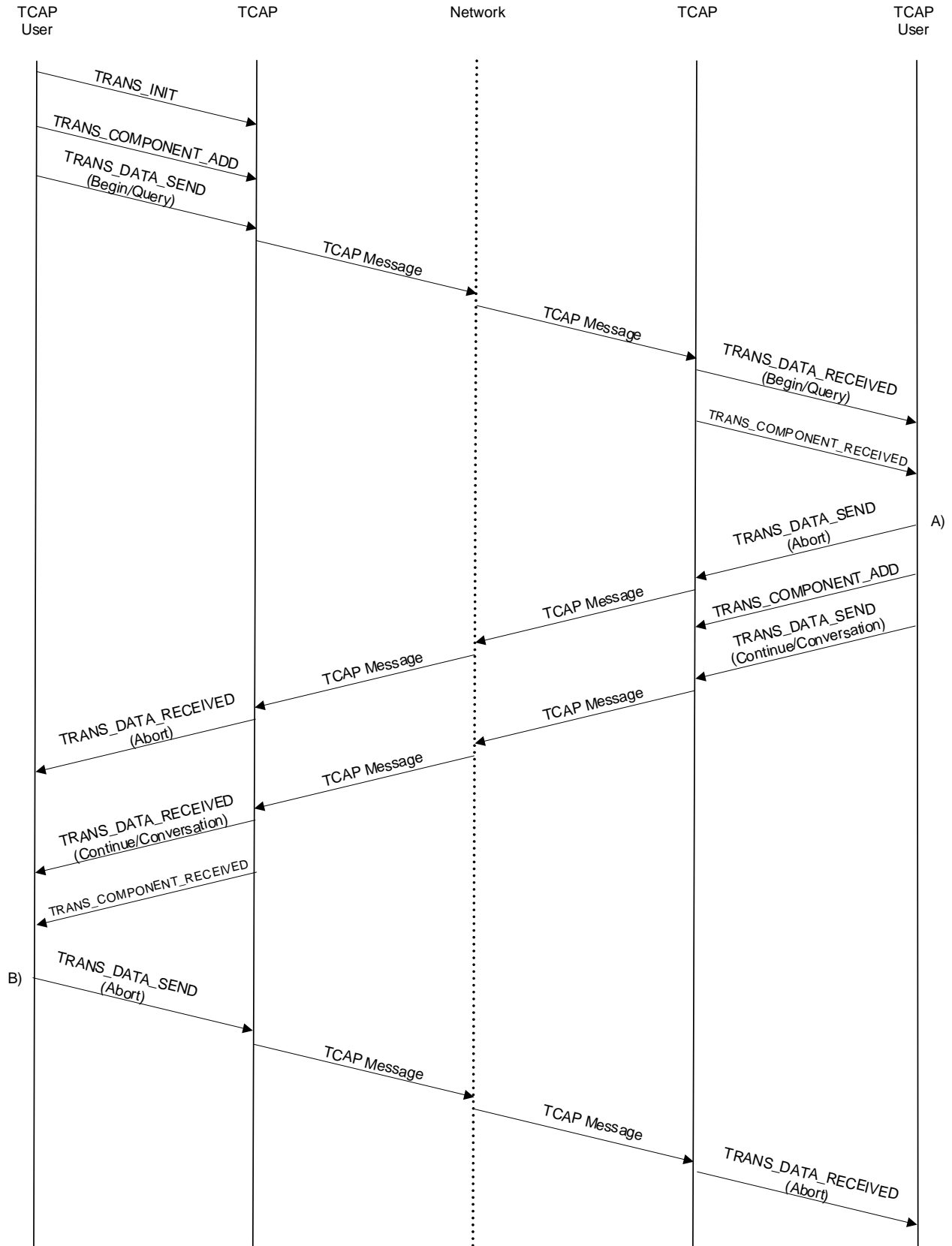


Figure 31 - TCAP successful structured dialogue abort

8.4 TCAP Alarms

8.4.1 Alarm

The TB640_MSG_ID_SS7_TCAP_NOTIF_ALARM (event) notification message is received by the TCAP service user when TCAP is reporting an error.

The following structure contains information concerning TCAP layer alarm:

```
typedef struct _TB640_EVT_SS7_TCAP_NOTIF_ALARM
{
    TBX_MSG_HEADER                Header;
    TBX_UINT32                    un32MsgVersion;
    TB640_SS7_TCAP_HANDLE        hTcap;

    TB640_SS7_TCAP_ALARM_CATEGORY    Category;
    TB640_SS7_TCAP_ALARM_EVENT       Event;
    TB640_SS7_TCAP_ALARM_CAUSE       Cause;
    TBX_UINT8                        aun8Padding0 [4];
} TB640_EVT_SS7_TCAP_NOTIF_ALARM, *PTB640_EVT_SS7_TCAP_NOTIF_ALARM;
```

General explanation of the fields of previous structure:

- The TCAP handle field (*hTcap*) specifies the handle of the TCAP instance.
- The category field (*Category*) indicates the category of the alarm. Possible values:

Table 128 - TCAP alarm category

Alarm Category	Description
TB640_SS7_TCAP_ALARM_CATEGORY_NONE	No category
TB640_SS7_TCAP_ALARM_CATEGORY_PROTOCOL	Protocol related alarms.
TB640_SS7_TCAP_ALARM_CATEGORY_INTERFACE	Interface related alarms.
TB640_SS7_TCAP_ALARM_CATEGORY_INTERNAL	Internal software error related alarms.
TB640_SS7_TCAP_ALARM_CATEGORY_RESOURCES	System resources related alarms.

- The event field (*Event*) specifies the event that cause the alarm.

Table 129 - TCAP alarm event

Alarm Event	Description
TB640_SS7_TCAP_ALARM_EVENT_NONE	No event
TB640_SS7_TCAP_ALARM_EVENT_INVALID_UI_EVENT	Invalid upper interface event.
TB640_SS7_TCAP_ALARM_EVENT_INVALID_LI_EVENT	Invalid lower interface event.
TB640_SS7_TCAP_ALARM_EVENT_INVALID_EVENT	Invalid network message.
TB640_SS7_TCAP_ALARM_EVENT_INVALID_TIMER_EVENT	Invalid timer expiration event.
TB640_SS7_TCAP_ALARM_EVENT_INVALID_MI_EVENT	Invalid layer manager event.
TB640_SS7_TCAP_ALARM_EVENT_BIND_FAIL	Bind with service provider failed.
TB640_SS7_TCAP_ALARM_EVENT_MSG_ALLOC_FAIL	Message allocation failed.
TB640_SS7_TCAP_ALARM_EVENT_ALLOC_FAIL	Buffer allocation failed.
TB640_SS7_TCAP_ALARM_EVENT_DUPLICATE_INVOKE_ID	Duplicate invoke ID.
TB640_SS7_TCAP_ALARM_EVENT_MANDATORY_MISSING	Mandatory element is missing.
TB640_SS7_TCAP_ALARM_EVENT_UNEXPECTED_RX_MSG	Received unexpected message.
TB640_SS7_TCAP_ALARM_EVENT_UNKNOW_RX_MSG	Received unknown message.
TB640_SS7_TCAP_ALARM_EVENT_INVALID_DIALOG_ID	Invalid dialog ID.
TB640_SS7_TCAP_ALARM_EVENT_UNRECOG_INVOKE_ID	Unrecognized invoke ID.

TB640_SS7_TCAP_ALARM_EVENT_ALLOC_DIALOG_FAIL	Dialog allocation failed.
TB640_SS7_TCAP_ALARM_EVENT_HASH_FAIL	Hash table operation failed.
TB640_SS7_TCAP_ALARM_EVENT_MAX_CONFIG	Reached configuration limit.
TB640_SS7_TCAP_ALARM_EVENT_INVALID_COMPONENT	Invalid component.

- The cause field (*Cause*) specifies the cause of the alarm.

Table 130 - TCAP alarm cause

Alarm Cause	Description
TB640_SS7_TCAP_ALARM_CAUSE_UNKNOW	No cause
TB640_SS7_TCAP_ALARM_CAUSE_INV_SAP	Invalid SAP.
TB640_SS7_TCAP_ALARM_CAUSE_INV_SP	Invalid service provider (out-of-range).
TB640_SS7_TCAP_ALARM_CAUSE_INV_SU	Invalid service user (out_of-range).
TB640_SS7_TCAP_ALARM_CAUSE_PROTOCOL_NOT_ACTIVE	Protocol layer is not active.
TB640_SS7_TCAP_ALARM_CAUSE_INTERFACE_VERSION	Invalid interface version.
TB640_SS7_TCAP_ALARM_CAUSE_UNBOUND	SAP unbound.
TB640_SS7_TCAP_ALARM_CAUSE_DIALOG_ALLOC_FAIL	Dialog allocation failed.
TB640_SS7_TCAP_ALARM_CAUSE_INVOKE_ALLOC_FAIL	Invoke allocation failed.
TB640_SS7_TCAP_ALARM_CAUSE_TUSAP_ASSOC_FAIL	User-part association failed.
TB640_SS7_TCAP_ALARM_CAUSE_SPSAP_ASSOC_FAIL	Interface association failed.
TB640_SS7_TCAP_ALARM_CAUSE_DECODE_FAIL	Message decoding problem.

8.4.2 Userpart alarm

The TB640_MSG_ID_SS7_TCAP_NOTIF_USERPART_ALARM (event) notification message is received by the TCAP service user when a TCAP userpart is reporting an error.

The following structure contains information concerning TCAP userpart alarm:

```
typedef struct _TB640_EVT_SS7_TCAP_NOTIF_USERPART_ALARM
{
    TBX_MSG_HEADER                Header;
    TBX_UINT32                    un32MsgVersion;
    TB640_SS7_TCAP_HANDLE         hTcap;
    TB640_SS7_TCAP_USERPART_HANDLE hTcapUserpart;

    TB640_SS7_TCAP_ALARM_CATEGORY Category;
    TB640_SS7_TCAP_ALARM_EVENT    Event;
    TB640_SS7_TCAP_ALARM_CAUSE    Cause;
} TB640_EVT_SS7_TCAP_NOTIF_USERPART_ALARM, *PTB640_EVT_SS7_TCAP_NOTIF_USERPART_ALARM;
```

General explanation of the fields of previous structure:

- The TCAP handle field (*hTcap*) specifies the handle of the TCAP instance.
- The TCAP Userpart handle field (*hTcapUserpart*) specifies the handle of the TCAP Userpart instance.
- The category field (*Category*) indicates the category of the Userpart alarm. Possible values: see Table 128 - TCAP alarm category.
- The event field (*Event*) specifies the event that cause the Userpart alarm. Possible values: see Table 129 - TCAP alarm event.

- The cause field (*Cause*) specifies the cause of the Userpart alarm. Possible values: see Table 130 - TCAP alarm cause.

8.5 TCAP States

States information, which can be gathered at any time by TCAP, indicates the current state of the userpart. This information can be used to determine the quality of service. The collection of status information does not affect any of the information examined.

8.5.1 Userpart states get

The TB640_MSG_ID_SS7_TCAP_STATES_USERPART_GET (request/response) message is used to obtain states from a TCAP userpart.

The **request** part of the message TB640_MSG_ID_SS7_TCAP_STATES_USERPART_GET contains the following field:

```

...
TB640_SS7_TCAP_USERPART_HANDLE      hTcapUserpart;      /* The handle of the instance of
...                                  * the TCAP userpart. */

```

The **response** part of the message TB640_MSG_ID_SS7_TCAP_STATES_USERPART_GET contains the following fields:

```

...
TB640_SS7_TCAP_PROTOCOL_VARIANT      ProtocolVariant;
TB640_SS7_TCAP_USERPART_STATUS_TYPE   Status;
...

```

General explanation of the parameters of response to TCAP userpart states get request:

- The protocol variant field (*ProtocolVariant*) indicates TCAP protocol variant of userpart. Possible values: see Table 127 - TCAP protocol variant.
- The Status field (*Status*) indicates the status of the userpart.

Table 131 - TCAP userpart status

Alarm Category	Description
TB640_SS7_TCAP_USERPART_STATUS_TYPE_UNBOUND	Unbound to SCCP user-part.
TB640_SS7_TCAP_USERPART_STATUS_TYPE_CONFIGURED	User-part configured.
TB640_SS7_TCAP_USERPART_STATUS_TYPE_BOUND	Bound to SCCP user-part.
TB640_SS7_TCAP_USERPART_STATUS_TYPE_WAIT_BIND_CONFIRM	Waiting for SCCP user-part bind confirm.
TB640_SS7_TCAP_USERPART_STATUS_TYPE_BIND_PENDING	Bind pending.

8.6 TCAP Statistics

The TCAP can gather statistics information at any time to measure the performance of the TCAP software. This information can be used to determine the distribution of traffic loads and Quality of Service (QoS) parameters, and to assist in TCAP software debugging. The collection of statistics information may or may not result in the counters being reset.

8.6.1 Userpart statistics

The TB640_MSG_ID_SS7_TCAP_STATS_USERPART_GET (request/response) message is used to obtain statistics from a TCAP userpart.

The **request** part of the message TB640_MSG_ID_SS7_TCAP_STATS_USERPART_GET contains the following fields:

```
...
TB640_SS7_TCAP_USERPART_HANDLE    hTcapUserpart; /* Handle of the TCAP userpart instance */
TBX_BOOL                          fResetStats;    /* Reset stats if required */
...
```

The **response** part of the message TB640_MSG_ID_SS7_TCAP_STATS_USERPART_GET contains the following field:

```
...
TB640_SS7_TCAP_USERPART_STATS     Statistics;
...
```

The following structure contains the statistic counters of TCAP userpart:

```
typedef struct _TB640_SS7_TCAP_USERPART_STATS
{
    TBX_UINT32    un32StructVersion;
    TBX_UINT8     aun8Padding0[4];

    /* Message Tx Statistics */

    /* The following counters are maintained for both ITU and ANSI. */
    TBX_UINT32    un32MsgTx;
    TBX_UINT32    un32UniTx;
    TBX_UINT32    un32AbtTx;

    /* The following counters are maintained for ITU only. */
    TBX_UINT32    un32BgnTx;
    TBX_UINT32    un32CntTx;
    TBX_UINT32    un32EndTx;

    /* The following counters are maintained for ANSI only. */
    TBX_UINT32    un32QwpTx;
    TBX_UINT32    un32QnpTx;
    TBX_UINT32    un32CwpTx;
    TBX_UINT32    un32CnpTx;
    TBX_UINT32    un32RspTx;

    /* Message Rx Statistics */

    /* The following counters are maintained for both ITU and ANSI. */
    TBX_UINT32    un32MsgRx;
    TBX_UINT32    un32UniRx;
    TBX_UINT32    un32AbtRx;

    /* The following counters are maintained for ITU only. */
    TBX_UINT32    un32BgnRx;
    TBX_UINT32    un32CntRx;
    TBX_UINT32    un32EndRx;

    /* The following counters are maintained for ANSI only. */
    TBX_UINT32    un32QwpRx;
    TBX_UINT32    un32QnpRx;
    TBX_UINT32    un32CwpRx;
    TBX_UINT32    un32CnpRx;
    TBX_UINT32    un32RspRx;

    /* Component Tx Statistics */

    /* The following counters are maintained for both ITU and ANSI. */
```



```

TBX_UINT32      un32CmpTx;
TBX_UINT32      un32InvTx;
TBX_UINT32      un32ResTx;
TBX_UINT32      un32ErrTx;
TBX_UINT32      un32RejTx;

/* Component Rx Statistics */

/* The following counters are maintained for both ITU and ANSI. */
TBX_UINT32      un32CmpRx;
TBX_UINT32      un32InvRx;
TBX_UINT32      un32ResRx;
TBX_UINT32      un32ErrRx;
TBX_UINT32      un32RejRx;

/* The following counters are maintained for both ITU and ANSI. */
TBX_UINT32      un32ActTrns;
TBX_UINT32      un32ActInv;
TBX_UINT32      un32TrnsId;
TBX_UINT32      un32Drop;

/* Fault statistics */

/* The following counters are maintained for both ITU and ANSI. */
TBX_UINT32      un32UrMsgRx;
TBX_UINT32      un32InTrnRx;
TBX_UINT32      un32BdTrnRx;
TBX_UINT32      un32UrTidRx;
TBX_UINT32      un32RsrcLRx;

/* The following counters are maintained for ANSI only. */
TBX_UINT32      un32PrRlsRx;
TBX_UINT32      un32UrDlgRx;
TBX_UINT32      un32BdDlgRx;
TBX_UINT32      un32MsDlgRx;
TBX_UINT32      un32InDlgRx;

/* The following counters are maintained for both ITU and ANSI. */
TBX_UINT32      un32UrCmpRx;
TBX_UINT32      un32InCmpRx;
TBX_UINT32      un32BdCmpRx;
TBX_UINT32      un32UrLidRx;
TBX_UINT32      un32UrIdRRRx;
TBX_UINT32      un32UxResRx;
TBX_UINT32      un32UrIdRERx;
TBX_UINT32      un32UxErrRx;

/* The following counter is maintained for ANSI only. */
TBX_UINT32      un32InEncRx;

/* The following counter is maintained for all protocol variant. */
TBX_UINT32      un32DupIdRx;
TBX_UINT32      un32UrOprRx;
TBX_UINT32      un32InPrmINRx;
TBX_UINT32      un32RsrcInvRx;
TBX_UINT32      un32RlsInvRx;
TBX_UINT32      un32UxLrspRx;
TBX_UINT32      un32UxLoprRx;
TBX_UINT32      un32InPrmRRRx;
TBX_UINT32      un32UrErrRx;
TBX_UINT32      un32UxEcdRx;
TBX_UINT32      un32InPrmRERx;

/* The following counters are maintained for both ITU and ANSI. */
TBX_UINT32      un32UrMsgTx;
TBX_UINT32      un32InTrnTx;
TBX_UINT32      un32BdTrnTx;
TBX_UINT32      un32UrTidTx;
TBX_UINT32      un32RsrcLTx;

/* The following counters are maintained for ANSI only. */
TBX_UINT32      un32PrRlsTx;
TBX_UINT32      un32UrDlgTx;
TBX_UINT32      un32BdDlgTx;
TBX_UINT32      un32MsDlgTx;
TBX_UINT32      un32InDlgTx;

/* The following counters are maintained for both ITU and ANSI. */
TBX_UINT32      un32UrCmpTx;
TBX_UINT32      un32InCmpTx;
TBX_UINT32      un32BdCmpTx;

```

```
TBX_UINT32      un32UrLidTx;
TBX_UINT32      un32UrIdRRTx;
TBX_UINT32      un32UxResTx;
TBX_UINT32      un32UrIdRETx;
TBX_UINT32      un32UxErrTx;

/* The following counter is maintained for ANSI only. */
TBX_UINT32      un32InEncTx;

/* The following counter is maintained for all protocol variant. */
TBX_UINT32      un32DupIdTx;
TBX_UINT32      un32UrOprTx;
TBX_UINT32      un32InPrmINTx;
TBX_UINT32      un32RsrcInvTx;
TBX_UINT32      un32RlsInvTx;
TBX_UINT32      un32UxLrspTx;
TBX_UINT32      un32UxLoprTx;
TBX_UINT32      un32InPrmRRTx;
TBX_UINT32      un32UrErrTx;
TBX_UINT32      un32UxEcdTx;
TBX_UINT32      un32InPrmRETx;

} TB640_SS7_TCAP_USERPART_STATS, *PTB640_SS7_TCAP_USERPART_STATS;
```

General explanation of the fields of previous structure:

- The structure version parameter (*un32StructVersion* field in structure TB640_SS7_TCAP_USERPART_STATS), is the structure version identifier. This parameter must be set to 1.
- The messages transmitted counter (*un32MsgTx*) indicates the total number of messages transmitted.
- The unidirectional transmitted counter (*un32UniTx*) indicates the total number of unidirectional messages transmitted.
- The abort transmitted counter (*un32AbtTx*) indicates the total number of abort messages transmitted.
- The begin messages transmitted counter (*un32BgnTx*) indicates the total number of begin messages transmitted.
- The continue messages transmitted counter (*un32CntTx*) indicates the total number of continue messages transmitted.
- The end transmitted counter (*un32EndTx*) indicates the total number of end messages transmitted.
- The query with permission transmitted counter (*un32QwpTx*) indicates the total number of query with permission messages transmitted.
- The query without permission transmitted counter (*un32QnpTx*) indicates the total number of query without permission messages transmitted.
- The conversation with permission transmitted counter (*un32CwpTx*) indicates the total number of conversation with permission messages transmitted.

- The conversation without permission transmitted counter (*un32CnpTx*) indicates the total number of conversation without permission messages transmitted.
- The conversation with permission transmitted counter (*un32CwpTx*) indicates the total number of conversation with permission messages transmitted.
- The response transmitted counter (*un32RspTx*) indicates the total number of response messages transmitted.
- The messages received counter (*un32MsgRx*) indicates the total number of messages received.
- The unidirectional received counter (*un32UniRx*) indicates the total number of unidirectional messages received.
- The abort received counter (*un32AbtRx*) indicates the total number of abort messages received.
- The begin messages received counter (*un32BgnRx*) indicates the total number of begin messages received.
- The continue messages received counter (*un32CntRx*) indicates the total number of continue messages received.
- The end received counter (*un32EndRx*) indicates the total number of end messages received.
- The query with permission received counter (*un32QwpRx*) indicates the total number of query with permission messages received.
- The query without permission received counter (*un32QnpRx*) indicates the total number of query without permission messages received.
- The conversation with permission received counter (*un32CwpRx*) indicates the total number of conversation with permission messages received.
- The conversation without permission counter (*un32CnpRx*) indicates the total number of conversation without permission messages.
- The response received counter (*un32RspRx*) indicates the total number of response messages received.
- The components transmitted counter (*un32CmpTx*) indicates the total number of components transmitted.

- The invoke components transmitted counter (*un32InvTx*) indicates the total number of invoke components transmitted.
- The result transmitted counter (*un32ResTx*) indicates the total number of return-result components transmitted.
- The error transmitted counter (*un32ErrTx*) indicates the total number of return-error components transmitted.
- The reject transmitted counter (*un32RejTx*) indicates the total number of reject component transmitted.
- The components received counter (*un32CmpRx*) indicates the total number of components received.
- The invoke components received counter (*un32InvRx*) indicates the total number of invoke components received.
- The result received counter (*un32ResRx*) indicates the total number of return-result components received.
- The error received counter (*un32ErrRx*) indicates the total number of return-error components received.
- The reject received counter (*un32RejRx*) indicates the total number of reject component received.
- The active transactions counter (*un32ActTrns*) indicates the total number of transactions currently active.
- The active invocations counter (*un32ActInvs*) indicates the total number of invocations currently active.
- The transaction IDs counter (*un32TrnsId*) indicates the total number of used transaction IDs.
- The dropped counter (*un32Drop*) indicates the total number of received messages dropped.
- The unrecognized message received counter (*un32UrMsgRx*) indicates the total number of unrecognized message type received.
- The unrecognized message transmitted counter (*un32UrMsgTx*) indicates the total number of unrecognized message type transmitted.

- The incorrect transaction received counter (*un32InTrnRx*) indicates the total number of incorrect transaction portion received.
- The incorrect transaction transmitted counter (*un32InTrnTx*) indicates the total number of incorrect transaction portion transmitted.
- The bad transaction received counter (*un32BdTrnRx*) indicates the total number of badly structured transaction portion received.
- The bad transaction transmitted counter (*un32BdTrnTx*) indicates the total number of badly structured transaction portion transmitted.
- The unrecognized transaction ID received counter (*un32UrTidRx*) indicates the total number of unrecognized transaction ID received.
- The unrecognized transaction ID transmitted counter (*un32UrTidTx*) indicates the total number of unrecognized transaction ID transmitted.
- The resource limitation received counter (*un32RsrcLRx*) indicates the total number of resource limitation received.
- The resource limitation transmitted counter (*un32RsrcLTx*) indicates the total number of resource limitation transmitted.
- The permission to release received counter (*un32PrRlsRx*) indicates the total number of transaction portion permission to release problem received.
- The permission to release transmitted counter (*un32PrRlsTx*) indicates the total number of transaction portion permission to release problem transmitted.
- The unrecognized dialogue received counter (*un32UrDlgRx*) indicates the total number of unrecognized dialogue portion ID received.
- The unrecognized dialogue transmitted counter (*un32UrDlgTx*) indicates the total number of unrecognized dialogue portion ID transmitted.
- The bad dialogue received counter (*un32BdDlgRx*) indicates the total number of badly structured dialogue portion received.
- The bad dialogue transmitted counter (*un32BdDlgTx*) indicates the total number of badly structured dialogue portion transmitted.
- The missing dialogue received counter (*un32MsDlgRx*) indicates the total number of missing dialogue portion received.

- The missing dialogue transmitted counter (*un32MsDlGTx*) indicates the total number of missing dialogue portion transmitted.
- The inconsistent dialogue received counter (*un32InDlGRx*) indicates the total number of inconsistent dialogue portion received.
- The inconsistent dialogue transmitted counter (*un32InDlGTx*) indicates the total number of inconsistent dialogue portion transmitted.
- The unrecognized component received counter (*un32UrCmpRx*) indicates the total number of unrecognized component portion received (general problem).
- The unrecognized component transmitted counter (*un32UrCmpTx*) indicates the total number of unrecognized component portion transmitted (general problem).
- The incorrect component received counter (*un32InCmpRx*) indicates the total number of incorrect component portion received (general problem).
- The incorrect component transmitted counter (*un32InCmpTx*) indicates the total number of incorrect component portion transmitted (general problem).
- The bad component received counter (*un32BdCmpRx*) indicates the total number of badly structured component portion received (general problem).
- The bad component transmitted counter (*un32BdCmpTx*) indicates the total number of badly structured component portion transmitted (general problem).
- The unrecognized linked ID received counter (*un32UrLidRx*) indicates the total number of unrecognized linked ID received (invoke problem).
- The unrecognized linked ID transmitted counter (*un32UrLidTx*) indicates the total number of unrecognized linked ID transmitted (invoke problem).
- The unrecognized invoke ID result received counter (*un32UrIidRRRx*) indicates the total number of unrecognized invoke ID received (return-result problem).
- The unrecognized invoke ID transmitted counter (*un32UrIidRRTx*) indicates the total number of unrecognized invoke ID transmitted (return-result problem).
- The unexpected result received counter (*un32UxResRx*) indicates the total number of unexpected return result received (return-result problem).
- The unexpected result transmitted counter (*un32UxResTx*) indicates the total number of unexpected return result transmitted (return-result problem).

- The unrecognized invoke ID error received counter (*un32UrIdRERx*) indicates the total number of unrecognized invoke ID received (return-error problem).
- The unrecognized invoke ID error transmitted counter (*un32UrIdRETx*) indicates the total number of unrecognized invoke ID transmitted (return-error problem).
- The unexpected error received counter (*un32UxErrRx*) indicates the total number of unexpected return error received (Return-error problem).
- The unexpected error transmitted counter (*un32UxErrTx*) indicates the total number of unexpected return error transmitted (return-error problem).
- The incorrect encoding received counter (*un32InEncRx*) indicates the total number of incorrect component encoding received (general problem).
- The incorrect encoding transmitted counter (*un32InEncTx*) indicates the total number of incorrect component encoding transmitted (general problem).
- The duplicate invoke ID received counter (*un32DupIdRx*) indicates the total number of duplicate invoke ID received (invoke problem).
- The duplicate invoke ID transmitted counter (*un32DupIdTx*) indicates the total number of duplicate invoke ID transmitted (invoke problem).
- The unrecognized operation code received counter (*un32UrOprRx*) indicates the total number of unrecognized operation code received (invoke problem).
- The unrecognized operation transmitted counter (*un32UrOprTx*) indicates the total number of unrecognized operation code transmitted (invoke problem).
- The incorrect invoke parameters received counter (*un32InPrmINRx*) indicates the total number of incorrect parameters received (invoke problem).
- The incorrect invoke parameters transmitted counter (*un32InPrmINTx*) indicates the total number of incorrect parameters transmitted (invoke problem).
- The resource invoke received counter (*un32RsrcInvRx*) indicates the total number of resource limitation received (invoke problem).
- The resource invoke transmitted counter (*un32RsrcInvTx*) indicates the total number of resource limitation transmitted (invoke problem).
- The release invoke received counter (*un32RlsInvRx*) indicates the total number of initiating release problem received (invoke problem).

- The release invoke transmitted counter (*un32RlsInvTx*) indicates the total number of initiating release problem transmitted (invoke problem).
- The unexpected linked response received counter (*un32UxLrspRx*) indicates the total number of unexpected linked response received (invoke problem).
- The unexpected linked response transmitted counter (*un32UxLrspTx*) indicates the total number of unexpected linked response transmitted (invoke problem).
- The unexpected linked operation received counter (*un32UxLoprRx*) indicates the total number of unexpected linked operation received (invoke problem).
- The unexpected linked operation transmitted counter (*un32UxLoprTx*) indicates the total number of unexpected linked operation transmitted (invoke problem).
- The incorrect result parameters received counter (*un32InPrmRRRx*) indicates the total number of incorrect parameters received (return-result problem).
- The incorrect result parameters transmitted counter (*un32InPrmRRTx*) indicates the total number of incorrect parameters transmitted (return-result problem).
- The unrecognized error received counter (*un32UrErrRx*) indicates the total number of unrecognized error code received (return-error problem).
- The unrecognized error transmitted counter (*un32UrErrTx*) indicates the total number of unrecognized error code transmitted (return-error problem).
- The unexpected error code received counter (*un32UxEcdRx*) indicates the total number of unexpected return error code received (return-error problem).
- The unexpected error code transmitted counter (*un32UxEcdTx*) indicates the total number of unexpected return error code transmitted (return-error problem).
- The incorrect error parameters received counter (*un32InPrmRERx*) indicates the total number of incorrect parameters received (return-error problem).
- The incorrect error parameters transmitted counter (*un32InPrmRETx*) indicates the total number of incorrect parameters transmitted (return-error problem).

9 REVISION HISTORY

9.1 Changed in revision 9010-00030-1U

- ✓ Clarified descriptions of Table 47 regarding the controlling state of a circuit (i.e. incoming, outgoing, bothways, controlling and controlled).

9.2 Changed in revision 9010-00030-1V

- ✓ Added network id parameter in SCCP Gtt Association and SCCP Gtt Address Map messages.

9.3 Changed in revision 9010-00030-1W

- ✓ Added use translation type parameter (*fUseTypeValue*) in GT format structures.

9.4 Changed in revision 9010-00030-1X

- ✓ Added ISUP ETSI, ETSIv3 and UK variant.

9.5 Changed in revision 9010-00030-1Y

- ✓ Added MTP3/SCCP default routing notifications.

9.6 Changed in revision 9010-00030-1Z

- ✓ Added a note which specifies that LUDT and LUDTS (SCCP) are not supported for now.

End of the document