TCAP Programmer’s Manual

Version 1.0.2

Table of Contents

S. No.	Version	Date	Comments
1	0.0.1	08-Feb-2012	Initial Draft
2	1.0.1	17-April-2012	First customer delivery
3	1.0.2	17-Aug-2012	GA of TCAP

1 Introduction

The basis of TCAP protocol implementation is ITU-T Specification Series Q.771-Q.779. The design of TCAP protocol layer is based on Sure Speed Architecture. This ensures that TCAP meets demands of modern telecommunication networks by providing reliability and efficient internal design.

This programmer's manual explains various API Primitives (Application Programming Interface Primitives) provided by TCAP protocol implementation (referred as just TCAP in the remaining document). It describes API parameters as well as the sequence in which API's should be invoked.

This manual should be used in conjunction with the TCAP sample application to understand the overall functioning of the TCAP library. The aim of this manual is to explain all the API services of the TCAP library. The sample application puts them into use and thus serves as a live working example for TCAP library.

2 API Return Values & Error Handling

Every API returns "–1" in case of failure and a non-negative value after successful execution. The global variable "`gTcEcode`" is set to the respective error code.

The TCAP library uses C Programming macro "`TC_SET_ECODE`" to set error code. This macro must be ported as per application developer's system requirements to raise alarms & events for monitoring error situations detected by the TCAP layer.

The macro "`TC_SET_ECODE`" is written in source file "`tcerr.h`". Presently, this macro set the global variable `gTcEcode` to the appropriate error code.

3 TCAP Library Installation

You must have downloaded the source code for TCAP in the form of gzipped tar. Name of this gzipped tar is `tcap_v_X_Y_Z.tar.gz`. Just unzip this file using command `**gzip -d tcap_v_X_Y_Z.tar.gz**'.

This will produce a file with name `tcap_v_X_Y_Z.tar`. Untar this file using command `**tar xvf tcap_v_X_Y_Z.tar**'.

Now you have complete source code for TCAP installed in the directory `./tcap'. Change to directory `./tcap' using command '**cd ./tcap**' to browse through the code. We will call the directory './tcap' as the parent installation directory.

3.1 Building the TCAP library

To build TCAP library from the source code follow the steps below:

1. Change to parent installation directory [./tcap] containing the source code for TCAP. Now change to directory `./tcap/src'.

2. Type `make' to build the library libtcap.a. This library is present in `./tcap/src' directory.

3. You may clean the program binaries and object files from the source code directory by typing `make clean'.

Please note that in case you are also using Sure Speed M3UA & CL-SCCP, then please install TCAP source code in the same directory where M3UA & CL-SCCP source code has been installed. For example if M3UA source code is present in "root" directory [./root/m3ua], then install CL-SCCP also in the "root" directory [./root/sccp] and TCAP also in the "root" [./root/tcap] directory.

3.2 Building the sample application

Please note that Sure Speed M3UA, CL-SCCP and Linux Kernel SCTP 1.0.6 or above is required to build a working sample application. To build TCAP sample applications from the source code follow the steps below:

1. Change to parent installation directory [./tcap] containing the source code for TCAP. Now change to directory `./tcap/demo'.

2. Type following command to build the sample application:
   - make

3. You can remove the program binaries and object files from the source code directory by typing `make clean'.

4. Sample application has been designed to demonstrate useful TCAP functionalities.

3.3 Compilers and Options

We have used `gcc' compiler on Linux system due to its wide spread popularity. The code is written in compiler and system independent manner. So, we expect TCAP to compile using any standard C/C++ compiler on any Operation System with very minimal porting effort.

4 Concept of NSAP

NSAP [Network Service Access Point] is a logical interface between TCAP layer and SCCP using which they communicate with each other. An NSAP is a collection of various parameters like SS7 Network Variant, SCCP Quality of Service parameters, Subsystem Number, Maximum PDU size allowed by SCCP etc.

Based on various MTP/SIGTRAN services and capabilities, SCCP can offer multiple NSAP to TCAP layer. For example one NSAP can be used for TCAP dialogues within local network and other NSAP can be used for TCAP dialogues with external networks.

5 Support for Distributed Application Development

TCAP supports distributed application development. Distribution of traffic at TCAP level can be done based on the transaction identifiers that are assigned by a particular TCAP instance. TCAP layer has been designed in such a way to provide flexibility to system designer to assign a specific range of transaction identifiers to a TCAP instance.

The assignment of a range of transaction identifiers to a TCAP instance can be done using API **TCAP_CONFIG_REQ**.

6 Sequence of API calls

This section explains sequence of API calls in different scenarios where TCAP is used.

6.1 Exchanging TCAP User Components without establishing a dialogue

Following figure shows the typical case where TCAP would be used for sending components from node-A to node-B without establishing a dialogue. The routing can be based on SSN or GT. Routing aspects are taken care by the SCCP & MTP/M3UA layers.

Before TCAP configuration is started at any node, it is assumed that SCCP, MTP/M3UA and SCTP layers have been configured and M3UA ASP/IPSP are in ACTIVE state and SCCP Subsystems are in allowed state.

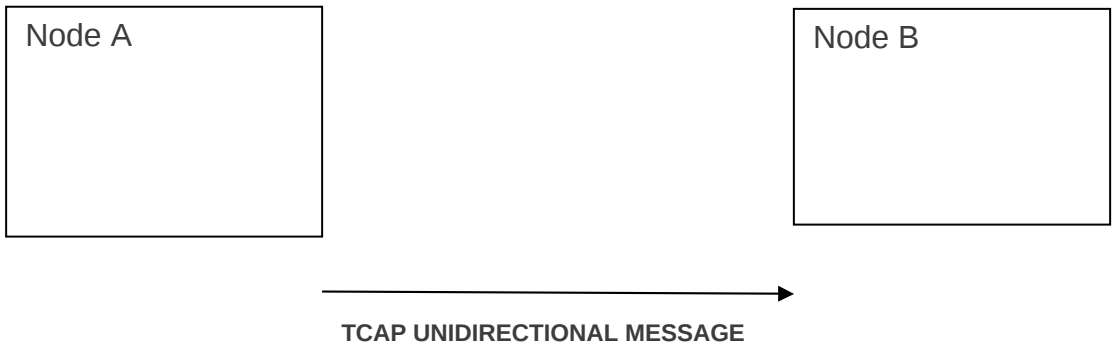

Figure 1: Example of Unidirectional TCAP Transaction

6.1.1 Configuring TCAP at Node-A

TCAP APIs need to be invoked in a specific sequence to initialize TCAP layer, configure various internal resources. Following is the sequence of APIs that need to be invoked to configure TCAP at node-A.

Note:- LM is Layer Management, USER is TCAP user

1. **TCAP_CONFIG_REQ (LM→TCAP)** – Initializes & configures the TCAP protocol layer. [Please note that all the TCAP resources are configurable and proper scaling of TCAP resources must be done as per the overall network dimensioning]
2. **TCAP_NSAP_CONFIG_REQ (LM→TCAP)** – This API is used to configure NSAP (SCCP User SAP) for TCAP layer through which it will take services from the SCCP layer.
3. **TCAP_INIT_DLG (USER →TCAP)** – TCAP user invokes this API to supply a TCAP dialogue related parameters to the TCAP layer. On receiving this API, TCAP layer reserves resources required to establish a dialogue. However, this does not guarantee successful establishment of a dialogue. This API returns dialogue identifier to the TCAP user.
4. **TCAP_INVOKE_REQ** or **TCAP_RESULT_L_REQ** or **TCAP_RESULT_NL_REQ** or **TCAP_U_ERROR_REQ** or **TCAP_U_REJECT_REQ (USER→TCAP)** – One or more of these APIs are invoked one or more times to supply components related to the dialogue.
5. **TCAP_UNIDIR_REQ (USER→TCAP)** – This API causes TCAP layer to format and send out a unidirectional message from node-A to node-B with components present in it.

6.1.2 Configuring TCAP at Node-B

Following is the sequence of TCAP APIs that need to be invoked to configure TCAP layer at node-B.

Note:- LM is Layer Management, USER is TCAP user

1. **TCAP_CONFIG_REQ (LM→TCAP)** – Initializes & configures the TCAP protocol layer. [Please note that all the TCAP resources are configurable and proper scaling of TCAP resources must be done as per the overall network dimensioning]
2. **TCAP_NSAP_CONFIG_REQ (LM→TCAP)** – This API is used to configure NSAP (SCCP User SAP) for TCAP layer through which it will take services from the SCCP layer.
3. **TCAP_N_UDT_IND (SCCP→TCAP) –** This API is used to pass information received in SCCP N-UNITDATA-IND to the TCAP.
4. **TCAP_USER_NTFY (USER→TCAP)** – This API is necessary to be invoked by the TCAP user after any event has been passed to the TCAP layer. For example, when a message or a timer expiry event has been passed to the TCAP layer. In response to this API, TCAP would provide notifications to the TCAP user with relevant information.

6.2 Exchanging TCAP User Components after establishing a dialogue

Following figure shows the typical case where TCAP would be used for sending components from node-A to node-B after establishing a dialogue. The routing can be based on SSN or GT. Routing aspects are taken care by the SCCP & MTP/M3UA layers.

Before TCAP configuration is started at any node, it is assumed that SCCP, MTP/M3UA and SCTP layers have been configured and M3UA ASP/IPSP are in ACTIVE state and SCCP Subsystems are in allowed state.

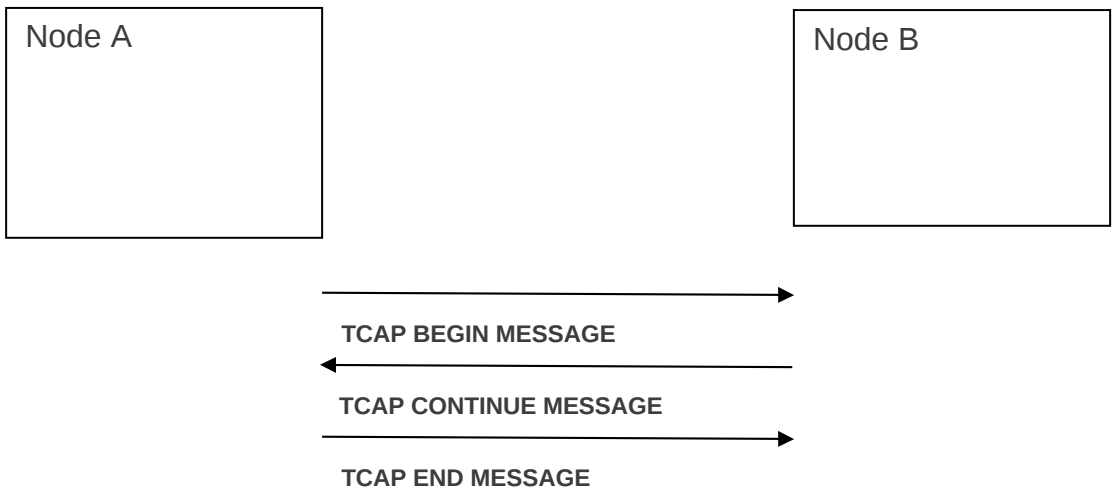

Figure 2: Example of TCAP Transaction within a dialogue

6.2.1 Configuring TCAP at Node-A

TCAP APIs need to be invoked in a specific sequence to initialize TCAP layer, configure various internal resources. Following is the sequence of APIs that need to be invoked to configure TCAP at node-A.

Note:- LM is Layer Management, USER is TCAP user

1. **TCAP_CONFIG_REQ (LM→TCAP)** – Initializes & configures the TCAP protocol layer. [Please note that all the TCAP resources are configurable and proper scaling of TCAP resources must be done as per the overall network dimensioning]
2. **TCAP_NSAP_CONFIG_REQ (LM→TCAP)** – This API is used to configure NSAP (SCCP User SAP) for TCAP layer through which it will take services from the SCCP layer.
3. **TCAP_INIT_DLG (USER →TCAP)** – TCAP user invokes this API to supply a TCAP dialogue related parameters to the TCAP layer. On receiving this API, TCAP layer reserves resources required to establish a dialogue. However, this does not guarantee successful establishment of a dialogue. This API returns dialogue identifier to the TCAP user.
4. **TCAP_INVOKE_REQ (USER→TCAP)** – This API is invoked to pass an invoke component to the TCAP layer related to the dialogue.
5. **TCAP_BEGIN_REQ (USER→TCAP)** – This API causes TCAP layer to format and send out a begin message from node-A to node-B with components present in it.

6. **TCAP_N_UDT_IND (SCCP→TCAP) –** This API is used to pass information received in SCCP N-UNITDATA-IND to the TCAP. In this case TCAP CONTINUE message is passed to the TCAP layer.
7. **TCAP_USER_NTFY (TCAP→USER)** – This API is necessary to be invoked by the TCAP user after any event has been passed to the TCAP layer. For example, when a message or a timer expiry event has been passed to the TCAP layer. In response to this API, TCAP would provide notifications to the TCAP user with relevant information. This API would provide TC_UN_CNT_IND and TC_UN_RRL_IND notifications to the node-A.
8. **TCAP_END_REQ (USER→TCAP) –** This API causes TCAP layer to format and send TCAP end message from node-A to node-B and dialogue is released at node-A.

6.2.2 Configuring TCAP at Node-B

Following is the sequence of TCAP APIs that need to be invoked to configure TCAP layer at node-B.

Note:- LM is Layer Management, USER is TCAP user

1. **TCAP_CONFIG_REQ (LM→TCAP)** – Initializes & configures the TCAP protocol layer. [Please note that all the TCAP resources are configurable and proper scaling of TCAP resources must be done as per the overall network dimensioning]
2. **TCAP_NSAP_CONFIG_REQ (LM→TCAP)** – This API is used to configure NSAP (SCCP User SAP) for TCAP layer through which it will take services from the SCCP layer.
3. **TCAP_N_UDT_IND (SCCP→TCAP) –** This API is used to pass information received in SCCP N-UNITDATA-IND to the TCAP. In this case TCAP BEGIN message is passed to the TCAP layer.
4. **TCAP_USER_NTFY (USER→TCAP)** – This API is necessary to be invoked by the TCAP user after any event has been passed to the TCAP layer. For example, when a message or a timer expiry event has been passed to the TCAP layer. In response to this API, TCAP would provide notifications to the TCAP user with relevant information. This API would provide TC_UN_BGN_IND and TC_UN_INV_IND notifications to the node-B.
5. **TCAP_RESULT_L_REQ (USER→TCAP)** – This API is invoked to pass the Return-Result-Last component to the TCAP layer related to the dialogue.
6. **TCAP_CONTINUE_REQ (USER→TCAP) –** This API causes TCAP layer to format and send TCAP continue message from node-B to node-A.
7. **TCAP_N_UDT_IND (SCCP→TCAP) –** This API is used to pass information received in SCCP N-UNITDATA-IND to the TCAP. In this case TCAP END message is passed to the TCAP layer.
8. **TCAP_USER_NTFY (USER→TCAP)** – This API is necessary to be invoked by the TCAP user after any event has been passed to the TCAP layer. For example, when a message or a timer expiry event has been passed to the TCAP layer. In response to this API, TCAP would provide notifications to the TCAP user with relevant information.

This API would provide TC_UN_END_IND notification to the node-B. This would result in freeing of dialogue & related resources at node-B.

7 Timing Management for TCAP Library

TCAP library manages timers internally and also takes appropriate actions whenever any timer expires. But timer ticks need to be supplied to the TCAP library from an external source. A timer tick is supplied to the TCAP library using TCAP_CHECK_TIMER_EXPIRY API. This API is discussed later in the document.

As TCAP timers may have granularity in milliseconds [1/1000 second], so it is recommended that at least one timer tick is provided to the TCAP library every 100 milliseconds. More than one timer tick may also be provided based on the application design.

Supplying a timer tick to the TCAP library is similar to passing an external event like a message or invoking any API.

8 Collecting Logs from TCAP Library

TCAP library logs real time messages/traces using following two C macros:

1. `TC_TRACE` – For logging messages
2. `TC_HEX_TRACE` – For logging hexadecimal contents

An application designer may port above two macros as per convenience to collect traces (logs) at desired location. These two macros are defined in include file “tctrc`.h`”. Presently, these macros are using standard C IO function (`printf`) for logging messages directly to the screen.

9 API between TCAP and SCCP

This section describes API interface between TCAP and SCCP layers. Complete details of these APIs may be found in document later.

9.1 Handling Notifications from SCCP

It is assumed that before TCAP is configured, SCCP subsystems are already provisioned and ready to use. A change in SCCP subsystems state may lead to change in overall availability state of a node for TCAP. The interface between TCAP and SCCP consists of following API's:

- **TCAP_N_UDT_IND** – This API is invoked for passing N-UNITDATA-IND to the TCAP layer for further processing.

- **TCAP_N_NOTICE_IND** – This API is invoked for passing N-NOTICE-IND to the TCAP layer for further processing.

9.2 Sending Data to SCCP

TCAP layer would send any messages to connection-less SCCP layer using the API **TCAP_N_UDT_REQ**. It is responsibility of TCAP application developer to write this API using services provided by the underlying SCCP layer.

9.3 Receiving Data from SCCP

Whenever any data is received by SCCP for TCAP layer, it needs to be passed to the TCAP layer. This can be done through **TCAP_N_UDT_IND** or **TCAP_N_NOTICE_IND** API. These TCAP API take arguments same as defined in SCCP API N-UNITDATA-IND and N-NOTICE-IND API as described in specification Q.711.

10 Application Programming Interface Usage

Following is the list (in given sequence) of include files that shall be added in the application source files which would make use of TCAP API's. **We strongly recommend you to have a look in the demonstration application source code that has been supplied along with the Sure Speed TCAP package. This would give you in-depth idea about usage of TCAP API's.**

```
#include <tctyp.h>
#include <tccfg.h>
#include <tcapi.h>
#include <tctrc.h>
#include <tcerr.h>
```

Above *include files* contain structures, enumerations, compile time defined values, error numbers and various other data structures that are required while writing application based on Sure Speed TCAP library.

The following sub-section of the document describes API's in detail, and services provided by each of the API implemented as part of Sure Speed TCAP software library. Please see section "***Data Structures Associated with API Primitives"*** for source code of structures used along with API primitives.

11 Scaling TCAP Library

Different applications would have different requirements from SCCP library in terms of number of managed objects it supports. This scaling process may be done before SCCP library is compiled.

Define Name	Include File	Default Value	Description
TCAP_MAX_PARAMS_LEN	tctyp.h	0x0200	Maximum length of parameters in a component
TC_MAX_NUM_CMPN	tctyp.h	0x08	Maximum number of components allowed in a message
TC_MAX_NUM_DLG	tctyp.h	0x4000	Maximum number of simultaneous TCAP dialogues supported
TC_MAX_NUM_INV	tctyp.h	0x4000	Maximum number of simultaneous TCAP invokes supported
TC_MAX_NUM_CMP	tctyp.h	0x4000	Maximum number of simultaneous TCAP components supported
TC_MAX_NUM_TIMERS	tctyp.h	0x2000	Maximum number of simultaneous TCAP timers supported
TC_MAX_USR_NTFY	tctyp.h	64	Maximum number of simultaneous user notifications that can be accumulated by TCAP before TCAP_USER_NTFY is invoked.

For example; if an TCAP application requires only 10 simultaneous dialogues, then TC_MAX_NUM_DLG can be set to 10. Similarly any other value above may be fine tuned as per the requirements. This would accordingly increase or decrease the overall memory requirement for TCAP layer.

12 Layer Management API (LM→TCAP)

The main function of layer management is to configure various resources within the TCAP layer. Following are Layer Management (LM) APIs that are used to configure TCAP layer.

12.1 TCAP_CONFIG_REQ

Description	This API is used by layer management to initialize the TCAP layer. The TCAP

	layer initializes all its internal data structures. **This API must be invoked before invoking any other API.** Setting of trace map can be done before this API is invoked. An application developer may scale internal data structures as per the requirement. Size of internal structures are defined in the include files “tctyp.h”.	

Prototype	`tcap_s32 tcap_config_req`
	`(`
	`tcap_u32 txnHdlOpt,`
	`tcap_u32 txnIdPrefix,`
	`tcap_u16 numDlg,`
	`tcap_u16 numInv,`
	`tcap_u16 numCmp`
	`);`	

Parameters	Associated C Data Structures	`None`
	`txnHdlOpt`	Transaction handling options; presently this needs to be set to 0 as no specific options are defined
	`txnIdPrefix`	Transaction Id prefix to be used when allocating a local transaction identifier. For example if this parameter is set to 0xABCD0000, then transaction identifiers used by this TCAP instance would be 0xABCDXXXX (where XXXX can be from 0000 to FFFF).
	`numDlg`	Number of simultaneous dialogues to be supported by TCAP. It should be taken care that this parameter does not exceed the maximum value possible with spare bits of `txnIdPrefix` parameter. (XXXX bits in `txnIdPrefix`).
	`numInv`	Number of simultaneous invokes to be supported by TCAP
	`numCmp`	Number of simultaneous components to be supported by TCAP
Restrictions/Bugs	None	
Pre-requisites	None	
Code Sample	Please refer source file demo/tcap.c	

12.2 TCAP_NSAP_CONFIG_REQ

Description	This API is used by layer management to configure SCCP NSAP. Using these NSAP, TCAP would take services from the SCCP layer.

Prototype	`tcap_s32 tcap_nsap_config_req`
	`(`
	`tcap_u32 nsapId,`
	`tcap_u32 maxDataSz,`
	`tcap_std_t ss7Std,`
	`tcap_qos_t *qos_p,`
	`tcap_u32 appId,`
	`tcap_u8 ssn,`
	`tcap_u32 opt`
	`);`

Parameters	Associated C Data Structures	`typedef struct`
		`{`
		`tcap_bool retOpt;`
		`tcap_u16 seqCtl;`
		`tcap_u8 imp;`
		`tcap_u8 hopCtr;`
		`} tcap_qos_t;`

	`nsapId`	NSAP identifier
	`maxDataSz`	Maximum data size accepted by this NSAP
	`ss7Std`	SS7 standard supported by this NSAP
	`qos_p`	Default SCCP Quality of Service parameters associated with this NSAP. Following is description of QOS parameters: 1. `retOpt` - SCCP Return Option parameter, this can be set to TC_TRUE or TC_FALSE 2. `seqCtl` - SCCP Sequence Control parameter, this can be set to a value between 0 to 0xFF which is considered to be valid values. TCAP user can set this to TC_NO_SEQCTL if sequencing is not desired. 3. `imp` - SCCP Importance parameter, TCAP user can set this parameter to TC_NO_IMP if importance is not to be sent in SCCP message 4. `hopCtr` - SCCP Hop Counter parameter, TCAP user can set this to a value between 0 to 0x0F. In case it is not be sent, then it can be set to TC_NO_HOPCTR. **Note :- The value of these parameters may need to be set according to the requirements of the underlying SCCP layer implementation.**
	`appId`	TCAP application identifier for notifications received from this NSAP
	`ssn`	Subsystem number associated with this NSAP
	`opt`	This parameter must be set to 0.
Restrictions/Bugs	None	
Pre-requisites	TCAP_CONFIG_REQ	
Code Sample	Please refer source file demo/tcap.c	

12.3 TCAP_CONFIG_REJECT_TIMEOUT

Description	This API is used by layer management to configure reject timer timeout duration.	

Prototype	`tcap_s32 tcap_config_reject_timeout`
	`(`
	`tcap_u32 rjTimeout`
	`);`	

Parameters	Associated C Data Structures	None
	`rjTimeout`	Duration of reject timer timeout in milliseconds.
Restrictions/Bugs	None	

Pre-requisites	TCAP_CONFIG_REQ
Code Sample	Please refer source file demo/tcap.c

13 Tracing Management API (LM→SCCP)

The main purpose of TCAP traces is to provide help in real-time as well as off line debugging. There are various trace levels as specified in following table for logging different types of conditions.

Trace Level	Description	Value
Debug level traces	General debugging traces	TC_DEBUG (16)
Message traces	Incoming and Outgoing Message dumps	TC_MSG (8)
Information traces	Information level traces, this is used for logging occasional events	TC_INFO (4)
Warning traces	Warning conditions are logged using this trace level	TC_WARN (2)
Error traces	Error conditions are logged using this trace level. It is recommended to keep this trace level switched ON.	TC_ERROR (1)

13.1 TCAP_SET_TRACE_REQ

Description	This API is used to enable one or more trace levels together for TCAP layer. For example to enable warning and error traces, trace level must be set to value `TC_WARN | TC_ERROR`.	
Prototype	`void tcap_set_trace_req ( tcap_u32 traceMap );`	
Parameters	`traceMap`	Trace levels that are to be enabled in the TCAP library. More than one trace level may be switched enabled by OR'ing trace levels. For example to switch ON error and message traces, following may be used: `traceMap = TC_ERROR | TC_MSG;`
Restrictions/ Bugs	None	
Pre-requisites	None	

13.2 TCAP_GET_TRACE_REQ

Description	This API is used to query the trace levels presently enabled in the TCAP layer.

Prototype	`tcap_u32 tcap_get_trace_req`
	`(`
	`void`
	`);`

Return Value	The trace map of TCAP layer is return value of this API. The return value is an unsigned 32 bit parameter.
Restrictions/Bugs	None
Pre-requisites	None

13.3 TCAP_SET_TRACE_LEVEL_REQ

Description	This API is used to enable a specific trace level for the TCAP layer. This trace level would be enabled in addition to the trace levels already enabled.	

Prototype	`void tcap_set_trace_level_req`
	`(`
	`tcap_u32 traceLevel`
	`);`	

Parameters	`traceLevel`	Trace level that is to be enabled in the TCAP library. It can take any valid TCAP trace level value.
Restrictions/Bugs	None	
Pre-requisites	None	

14 User APIs (USER→TCAP)

Following APIs are used by TCAP Users to take services provided by the TCAP Layer.

14.1 Dialogue Management API's

This section describes TCAP dialogue management API's.

14.1.1 TCAP_INIT_DLG

Description	This API is used by TCAP User to initiate setup of a dialogue. Dialogue related parameters are supplied to the TCAP layer through this API. **In case of success, this API returns the dialogue identifier for the newly allocated dialogue.**

Prototype	`tcap_s32 tcap_init_dialogue`
	`(`
	`tcap_u32 nsapId,`

		tcap_addr_t *srcAdd_p, tcap_addr_t *dstAdd_p, tcap_qos_t *qos_p, tcap_u32 opt);
Parameters	Associated C Data Structures	typedef struct { tcap_u8 natResv:1; tcap_u8 routeInd:1; tcap_u8 gtInd:4; tcap_u8 ssnInd:1; tcap_u8 pcInd:1; } tcap_sccp_addr_ind_t; typedef struct { tcap_u8 nai; tcap_u8 es; tcap_u8 np; tcap_u8 tt; tcap_u8 numDigits; tcap_u8 digits[TC_MAX_GT_DIGITS]; } tcap_gt_t; /* * If pcInd is set to 1, then include PC in out-going address * If pc is set to TC_PC_VAL_UNDEF, then PC not supplied by TCU */ #define TC_PC_VAL_UNDEF (~0) typedef struct { tcap_sccp_addr_ind_t ai; tcap_u8 ssn; tcap_u32 pc; tcap_gt_t gt; } tcap_addr_t; typedef struct { tcap_bool retOpt; tcap_u16 seqCtl; tcap_u8 imp; tcap_u8 hopCtr; } tcap_qos_t;
	nsapId	SCCP service access point through which TCAP layer is taking services from the SCCP layer.
	srcAdd_p	Pointer to source address; This parameter consists of “Address Indicator”, SSN, PC and GT. The point code parameter needs to be set to a valid point code value or “TC_PC_VAL_UNDEF” [numeric value

		0xFFFFFFFF] if point code is not available. For more details about these parameters, please refer to SCCP specification.
	dstAdd_p	Pointer to destination address. This address is sent to the SCCP layer, where it may be subject to modification post Global Title Translation. For more details about these parameters, please refer to SCCP specification.
	qos_p	SCCP quality of service parameters to be used for this TCAP transaction.
	opt	This parameter must be set to 0.
Restrictions/Bugs	None	
Pre-requisites	TCAP_CONFIG_REQ, TCAP_NSAP_CONFIG_REQ	
Code Sample	Please refer source file demo/tcap.c	

14.1.2 TCAP_UNIDIR_REQ

Description	This API is used by TCAP User to send a TCAP UNIDIRECTIONAL message without establishing any dialogue.	

Prototype	`tcap_s32 tcap_unidirectional_req`
	`(`
	`tcap_u32 dlgId,`
	`tcap_acn_t *acn_p,`
	`tcap_usr_info_t *usrInfo_p`
	`);`	

Parameters	Associated C Data Structures	`typedef struct`
		`{`
		`tcap_bool avbl;`
		`tcap_u8 len;`
		`tcap_u8 acn[TC_ACN_SIZE];`
		`} tcap_acn_t;`
		
		`typedef struct`
		`{`
		`tcap_bool avbl;`
		`tcap_u32 len;`
		`tcap_u8 objId[TC_MAX_OBJ_ID_SIZE];`
		`} tcap_obj_id_t;`
		
		`typedef struct`
		`{`
		`tcap_bool avbl;`
		`tcap_u32 val;`
		`} tcap_int_val_t;`
		
		`typedef struct`
		`{`
		`tcap_bool avbl;`
		`tcap_u32 len;`

		tcap_u8 objDesc[TC_MAX_OBJ_DESC_SIZE]; } tcap_obj_desc_t; typedef struct { tcap_bool avbl; tcap_obj_id_t objId; tcap_int_val_t idrRef; tcap_obj_desc_t objDesc; tcap_u32 encLen; tcap_u8 encType; tcap_u8 tagForm; tcap_u8 encInfo[TC_MAX_USR_INFO_SIZE]; } tcap_usr_info_t;
	dlgId	Dialogue Identifier
	acn_p	Pointer to Application context name parameter
	usrinfo_p	Pointer to TCAP dialogue portion user information parameter; • avbl - If user information parameter is to be sent in the dialogue portion, then this must be set to TC_TRUE. Otherwise, set this to TC_FALSE, so that it is not sent in the dialogue portion of TCAP message • objId - direct reference parameter of user information • idrRef - Indirect reference parameter of user information • objDesc - Object Descriptor parameter of user information
Restrictions/Bugs	None	
Pre-requisites	• TCAP_CONFIG_REQ • TCAP_NSAP_CONFIG_REQ • TCAP_INIT_DLG • TCAP_INVOKE_REQ/TCAP_RESULT_L_REQ/TCAP_RESULT_NL_REQ/TCAP_U_ERROR_REQ/TCAP_U_REJECT_REQ	
Code Sample	Please refer source file demo/tcap.c	

14.1.3 TCAP_BEGIN_REQ

Description	This API is used by TCAP User to send a TCAP BEGIN message to initiate establishment of a dialogue	
Prototype	tcap_s32 tcap_begin_req (tcap_u32 dlgId, tcap_acn_t *acn_p, tcap_usr_info_t *usrInfo_p);	
Parameters	Associated C Data Structures	typedef struct { tcap_bool avbl;

		tcap_u8 len; tcap_u8 acn[TC_ACN_SIZE]; } tcap_acn_t; typedef struct { tcap_bool avbl; tcap_u32 len; tcap_u8 objId[TC_MAX_OBJ_ID_SIZE]; } tcap_obj_id_t; typedef struct { tcap_bool avbl; tcap_u32 val; } tcap_int_val_t; typedef struct { tcap_bool avbl; tcap_u32 len; tcap_u8 objDesc[TC_MAX_OBJ_DESC_SIZE]; } tcap_obj_desc_t; typedef struct { tcap_bool avbl; tcap_obj_id_t objId; tcap_int_val_t idrRef; tcap_obj_desc_t objDesc; tcap_u32 encLen; tcap_u8 encType; tcap_u8 tagForm; tcap_u8 encInfo[TC_MAX_USR_INFO_SIZE]; } tcap_usr_info_t;
	dlgId	Dialogue Identifier
	acn_p	Pointer to Application context name parameter
	usrinfo_p	Pointer to TCAP dialogue portion user information parameter; - avbl - If user information parameter is to be sent in the dialogue portion, then this must be set to TC_TRUE. Otherwise, set this to TC_FALSE, so that it is not sent in the dialogue portion of TCAP message - objId - direct reference parameter of user information - idrRef - Indirect reference parameter of user information - objDesc - Object Descriptor parameter of user information
Restrictions/Bugs	None	
Pre-requisites	- TCAP_CONFIG_REQ - TCAP_NSAP_CONFIG_REQ - TCAP_INIT_DLG	
Code Sample	Please refer source file demo/tcap.c	

14.1.4 TCAP_END_REQ

<table>
<tr><td>Description</td><td colspan="2">This API is used by TCAP User to send a TCAP END message to release a dialogue</td></tr>
<tr><td>Prototype</td><td colspan="2">

```
tcap_s32 tcap_end_req
      (
            tcap_u32 dlgId,
            tcap_qos_t *qos_p,
            tcap_acn_t *acn_p,
            tcap_usr_info_t *usrInfo_p,
            tcap_u8 termType
      );
```

</td></tr>
<tr><td>Parameters</td><td>Associated C Data Structures</td><td>

```
typedef struct
{
      tcap_bool retOpt;
      tcap_u16 seqCtl;
      tcap_u8 imp;
      tcap_u8 hopCtr;
} tcap_qos_t;

typedef struct
{
      tcap_bool         avbl;
      tcap_u8           len;
      tcap_u8           acn[TC_ACN_SIZE];
} tcap_acn_t;

typedef struct
{
      tcap_bool         avbl;
      tcap_u32          len;
      tcap_u8           objId[TC_MAX_OBJ_ID_SIZE];
} tcap_obj_id_t;

typedef struct
{
      tcap_bool         avbl;
      tcap_u32          val;
} tcap_int_val_t;

typedef struct
{
      tcap_bool         avbl;
      tcap_u32          len;
      tcap_u8     objDesc[TC_MAX_OBJ_DESC_SIZE];
} tcap_obj_desc_t;

typedef struct
{
      tcap_bool         avbl;
      tcap_obj_id_t           objId;
```

</td></tr>
</table>

		tcap_int_val_t idrRef; tcap_obj_desc_t objDesc; tcap_u32 encLen; tcap_u8 encType; tcap_u8 tagForm; tcap_u8 encInfo[TC_MAX_USR_INFO_SIZE]; } tcap_usr_info_t;
	dlgId	Dialogue Identifier
	qos_p	SCCP Quality of service parameters
	acn_p	Pointer to Application context name parameter
	usrinfo_p	Pointer to TCAP dialogue portion user information parameter; • avbl - If user information parameter is to be sent in the dialogue portion, then this must be set to TC_TRUE. Otherwise, set this to TC_FALSE, so that it is not sent in the dialogue portion of TCAP message • objId - direct reference parameter of user information • idrRef - Indirect reference parameter of user information • objDesc - Object Descriptor parameter of user information
	termType	TCAP dialogue termination type :- • TC_TERM_TYPE_NORMAL_END (0x01) - basic ending of dialogue • TC_TERM_TYPE_PREARRANGED_END (0x02) - prearranged ending of dialogue
Restrictions/Bugs	None	
Pre-requisites	• TCAP_CONFIG_REQ • TCAP_NSAP_CONFIG_REQ • Dialogue must be established or in process of establishment	
Code Sample	Please refer source file demo/tcap.c	

14.1.5 TCAP_CONTINUE_REQ

Description	This API is used by TCAP User to send a TCAP CONTINUE message to continue/accept a dialogue	
Prototype	tcap_s32 tcap_continue_req (tcap_u32 dlgId, tcap_addr_t *srcAdd_p,/* if NULL, use received */ tcap_qos_t *qos_p, tcap_acn_t *acn_p, tcap_usr_info_t *usrInfo_p);	
Parameters	Associated C Data Structures	typedef struct { tcap_bool retOpt;

```
      tcap_u16 seqCtl;
      tcap_u8 imp;
      tcap_u8 hopCtr;
} tcap_qos_t;

typedef struct
{
      tcap_bool         avbl;
      tcap_u8           len;
      tcap_u8           acn[TC_ACN_SIZE];
} tcap_acn_t;

typedef struct
{
      tcap_bool         avbl;
      tcap_u32          len;
      tcap_u8           objId[TC_MAX_OBJ_ID_SIZE];
} tcap_obj_id_t;

typedef struct
{
      tcap_bool         avbl;
      tcap_u32          val;
} tcap_int_val_t;

typedef struct
{
      tcap_bool         avbl;
      tcap_u32          len;
      tcap_u8     objDesc[TC_MAX_OBJ_DESC_SIZE];
} tcap_obj_desc_t;

typedef struct
{
      tcap_bool         avbl;
      tcap_obj_id_t           objId;
      tcap_int_val_t          idrRef;
      tcap_obj_desc_t         objDesc;
      tcap_u32          encLen;
      tcap_u8                 encType;
      tcap_u8                 tagForm;
      tcap_u8     encInfo[TC_MAX_USR_INFO_SIZE];
} tcap_usr_info_t;

typedef struct
{
    tcap_u8 natResv:1;
    tcap_u8 routeInd:1;
    tcap_u8 gtInd:4;
    tcap_u8 ssnInd:1;
    tcap_u8 pcInd:1;
} tcap_sccp_addr_ind_t;

typedef struct
{
```

		tcap_u8 nai; tcap_u8 es; tcap_u8 np; tcap_u8 tt; tcap_u8 numDigits; tcap_u8 digits[TC_MAX_GT_DIGITS]; } tcap_gt_t; /* * If pcInd is set to 1, then include PC in out-going address * If pc is set to TC_PC_VAL_UNDEF, then PC not supplied by TCU */ #define TC_PC_VAL_UNDEF (~0) typedef struct { tcap_sccp_addr_ind_t ai; tcap_u8 ssn; tcap_u32 pc; tcap_gt_t gt; } tcap_addr_t;
	dlgId	Dialogue Identifier
	srcAdd_p	Source/Calling party address to be used in the SCCP connectionless message; this parameter is to be used only in case of first CONTINUE message. If this parameter is set to NULL, then this parameter is not used by TCAP.
	qos_p	SCCP Quality of service parameters; if this parameter is set to NULL then default QOS value is used by TCAP.
	acn_p	Pointer to application context name parameter
	usrinfo_p	Pointer to TCAP dialogue portion user information parameter; • avbl - If user information parameter is to be sent in the dialogue portion, then this must be set to TC_TRUE. Otherwise, set this to TC_FALSE, so that it is not sent in the dialogue portion of TCAP message • objId - direct reference parameter of user information • idrRef - Indirect reference parameter of user information • objDesc - Object Descriptor parameter of user information
Restrictions/Bugs	None	
Pre-requisites	• TCAP_CONFIG_REQ • TCAP_NSAP_CONFIG_REQ • Dialogue must be established or in process of establishment	
Code Sample	Please refer source file demo/tcap.c	

14.1.6 TCAP_U_ABORT_REQ

<table>
<tr><td>Description</td><td colspan="2">This API is used by TCAP User to send a TCAP ABORT message to abort a dialogue</td></tr>
<tr><td>Prototype</td><td colspan="2"><pre>
tcap_s32 tcap_u_abort_req
      (
            tcap_u32 dlgId,
            tcap_qos_t *qos_p,
            tcap_u8 abtRsn,
            tcap_acn_t *acn_p,
            tcap_usr_info_t *usrInfo_p
      );
</pre></td></tr>
<tr><td>Parameters</td><td>Associated C Data Structures</td><td><pre>
typedef struct
{
      tcap_bool retOpt;
      tcap_u16 seqCtl;
      tcap_u8 imp;
      tcap_u8 hopCtr;
} tcap_qos_t;

typedef struct
{
      tcap_bool   avbl;
      tcap_u8   len;
      tcap_u8   acn[TC_ACN_SIZE];
} tcap_acn_t;

typedef struct
{
      tcap_bool   avbl;
      tcap_u32   len;
      tcap_u8   objId[TC_MAX_OBJ_ID_SIZE];
} tcap_obj_id_t;

typedef struct
{
      tcap_bool   avbl;
      tcap_u32   val;
} tcap_int_val_t;

typedef struct
{
      tcap_bool   avbl;
      tcap_u32   len;
      tcap_u8   objDesc[TC_MAX_OBJ_DESC_SIZE];
} tcap_obj_desc_t;

typedef struct
{
      tcap_bool   avbl;
      tcap_obj_id_t   objId;
      tcap_int_val_t   idrRef;
      tcap_obj_desc_t   objDesc;
      tcap_u32   encLen;
</pre></td></tr>
</table>

		tcap_u8 encType; tcap_u8 tagForm; tcap_u8 encInfo[TC_MAX_USR_INFO_SIZE]; } tcap_usr_info_t;
	dlgId	Dialogue Identifier
	qos_p	SCCP Quality of service parameters; if this parameter is set to NULL then default QOS value is used by TCAP.
	abtRsn	Abort Reason; following are valid values for this parameter: • TC_NO_ABT_RSN (0x00) - Abort reason not available • TC_ABT_RSN_DLG_REF (0x01) - Dialogue refused • TC_ABT_RSN_ACN_NOT_SUPP (0x02) - ACN not supported • TC_ABT_RSN_USR_DEF (0x03) - User determined abort of dialogue
	acn_p	Pointer to application context name parameter
	usrinfo_p	Pointer to TCAP dialogue portion user information parameter.
Restrictions/Bugs	None	
Pre-requisites	• TCAP_CONFIG_REQ • TCAP_NSAP_CONFIG_REQ • Dialogue must be established or in process of establishment	
Code Sample	Please refer source file demo/tcap.c	

14.2 Component Management API's

This section describes TCAP component management API's.

14.2.1 TCAP_INVOKE_REQ

Description	This API is used to send an invoke component in a TCAP dialogue	
Prototype	tcap_s32 tcap_invoke_req (tcap_u32 dlgId, tcap_u8 opClass, tcap_u8 invokeId, tcap_bool linkedOp, tcap_u8 linkedId, tcap_opcode_t *opCode_p, tcap_params_t *params_p, tcap_u32 timeout);	
Parameters	Associated C Data Structures	typedef struct { tcap_u8 avbl; tcap_u8 type;

		tcap_u32 opCode; } tcap_opcode_t; typedef struct { tcap_u8 avbl; tcap_u32 len; tcap_u8 params[TC_MAX_PARAMS_LEN]; } tcap_params_t;
	dlgId	TCAP dialogue identifier
	opClass	Operation Class; valid values are 0x01 to 0x04
	invokeId	Invoke identifier
	linkedOp	Specifies if this invoke is a linked operation
	linkedId	Linked operation identifier; this parameter needs to be given a valid value if linkedOp parameter is set to TC_TRUE.
	opCode_p	Pointer to operation code parameter
	params_p	Pointer to parameters to be sent with the invoke component. This parameter includes entire encoding of the parameters portion of component. This means parameter tag, parameter length and parameter value.
	timeout	Time out for the operation in milliseconds
Restrictions/Bugs	None	
Pre-requisites	• TCAP_CONFIG_REQ • TCAP_NSAP_CONFIG_REQ • TCAP_INIT_DLG	
Code Sample	Please refer source file demo/tcap.c	

14.2.2 TCAP_U_CANCEL_REQ

Description	This API is used to cancel a pending operation/invoke component	
Prototype	tcap_s32 tcap_u_cancel_req (tcap_u32 dlgId, tcap_u8 invId);	
Parameters	Associated C Data Structures	None
	dlgId	TCAP dialogue identifier
	invokeId	Invoke identifier
Restrictions/Bugs	None	
Pre-requisites	• TCAP_CONFIG_REQ • TCAP_NSAP_CONFIG_REQ	

	• Invoke component is pending with TCAP layer
Code Sample	Please refer source file demo/tcap.c

14.2.3 TCAP_RESULT_L_REQ

Description	This API is used to send a return-result-last component in a TCAP dialogue	
Prototype	`tcap_s32 tcap_result_l_req ( tcap_u32 dlgId, tcap_u8 invokeId, tcap_opcode_t *opCode_p, tcap_params_t *params_p );`	

Parameters	Associated C Data Structures	`typedef struct { tcap_u8 avbl; tcap_u8 type; tcap_u32 opCode; } tcap_opcode_t;`
		
		`typedef struct { tcap_u8 avbl; tcap_u32 len; tcap_u8 params[TC_MAX_PARAMS_LEN]; } tcap_params_t;`

	`dlgId`	TCAP dialogue identifier
	`invokeId`	Invoke identifier
	`opCode_p`	Pointer to operation code parameter
	`params_p`	Pointer to parameters to be sent with the return-result-last component. This parameter includes entire encoding of the parameters portion of component. This means parameter tag, parameter length and parameter value.
Restrictions/Bugs	None	
Pre-requisites	• TCAP_CONFIG_REQ • TCAP_NSAP_CONFIG_REQ • Operation is already initiated	
Code Sample	Please refer source file demo/tcap.c	

14.2.4 TCAP_RESULT_NL_REQ

Description	This API is used to send a return-result-not-last component in a TCAP dialogue

Prototype	tcap_s32 tcap_result_nl_req (tcap_u32 dlgId, tcap_u8 invokeId, tcap_opcode_t *opCode_p, tcap_params_t *params_p);	
Parameters	Associated C Data Structures	typedef struct { tcap_u8 avbl; tcap_u8 type; tcap_u32 opCode; } tcap_opcode_t; typedef struct { tcap_u8 avbl; tcap_u32 len; tcap_u8 params[TC_MAX_PARAMS_LEN]; } tcap_params_t;
	dlgId	TCAP dialogue identifier
	invokeId	Invoke identifier
	opCode_p	Pointer to operation code parameter
	params_p	Pointer to parameters to be sent with the return-result-not-last component. This parameter includes entire encoding of the parameters portion of component. This means parameter tag, parameter length and parameter value.
Restrictions/Bugs	None	
Pre-requisites	• TCAP_CONFIG_REQ • TCAP_NSAP_CONFIG_REQ • Operation is already initiated	
Code Sample	Please refer source file demo/tcap.c	

14.2.5 TCAP_U_ERROR_REQ

Description	This API is used to send a return-error component in a TCAP dialogue	
Prototype	tcap_s32 tcap_u_error_req (tcap_u32 dlgId, tcap_u8 invokeId, tcap_error_code_t *errCode_p, tcap_params_t *params_p);	
Parameters	Associated C Data Structures	typedef struct { tcap_u8 avbl;

		tcap_u8 type; tcap_u8 len; tcap_u8 val[TC_MAX_OBJ_ID_SIZE]; } tcap_error_code_t; typedef struct { tcap_u8 avbl; tcap_u32 len; tcap_u8 params[TC_MAX_PARAMS_LEN]; } tcap_params_t;
	dlgId	TCAP dialogue identifier
	invokeId	Invoke identifier
	errCode_p	Pointer to error code parameter
	params_p	Pointer to parameters to be sent with the return-error component. This parameter includes entire encoding of the parameters portion of component. This means parameter tag, parameter length and parameter value.
Restrictions/Bugs	None	
Pre-requisites	• TCAP_CONFIG_REQ • TCAP_NSAP_CONFIG_REQ • Operation is already initiated	
Code Sample	Please refer source file demo/tcap.c	

14.2.6 TCAP_U_REJECT_REQ

Description	This API is used to send a reject component in a TCAP dialogue	
Prototype	tcap_s32 tcap_u_reject_req (tcap_u32 dlgId, tcap_u8 invokeId, tcap_problem_code_t *prbCode_p);	
Parameters	Associated C Data Structures	typedef struct { tcap_u8 avbl; tcap_u8 type; tcap_u32 code; } tcap_problem_code_t;
	dlgId	TCAP dialogue identifier
	invokeId	Invoke identifier
	prbCode_p	Pointer to problem code parameter
Restrictions/Bugs	None	

Pre-requisites	• TCAP_CONFIG_REQ • TCAP_NSAP_CONFIG_REQ • Operation is already initiated
Code Sample	Please refer source file `demo/tcap.c`

14.2.7 TCAP_TIMER_RESET_REQ

Description	This API is used to reset/restart invoke timer	

Prototype	`tcap_s32 tcap_timer_reset_req`
	`(`
	`tcap_u32 dlgId,`
	`tcap_u8 invId`
	`);`	

Parameters	Associated C Data Structures	`None`
	`dlgId`	TCAP dialogue identifier
	`invokeId`	Invoke identifier
Restrictions/Bugs	None	
Pre-requisites	• TCAP_CONFIG_REQ • TCAP_NSAP_CONFIG_REQ • Response to invoke component is not yet received	
Code Sample	Please refer source file `demo/tcap.c`	

15 User APIs (TCAP→User)

This section lists those APIs which are used by TCAP layer for passing notifications/events to the TCAP users. These TCAP users are identified using SCCP-SAP identifier or application identifier.

15.1 TCAP_USER_NTFY

Description	This API must be used by TCAP user to collect notifications from TCAP. This API must be invoked every time after an external event has been passed to the TCAP layer. For example, this API must be invoked after invoking any other TCAP API that is used to pass an SCCP event or after passing a timer tick to the TCAP layer. The list of APIs after which TCAP_USER_NTFY API must be invoked is as following: • TCAP_N_UDT_IND • TCAP_N_NOTICE_IND • TCAP_CHECK_TIMER_EXPIRY A timer tick can result in an internal timer expiry that can be a notification to the layer

<table>
<tr><td></td><td colspan="2">management or TCAP user.

This API must be repeatedly invoked until it returns failure (-1).

This API is used to notify following conditions:

1. Receipt of unidirectional message
2. Receipt of begin message
3. Receipt of continue message
4. Receipt of end message
5. U-Abort event
6. P-Abort event
7. U-Error event
8. Invoke Component
9. Return-result-Last component
10. Return-result-not-last component
11. Local Reject event
12. Remote Reject event
13. User Reject event
14. Local Cancel event
15. SCCP notice event</td></tr>
<tr><td>Prototype</td><td colspan="2"><pre>tcap_s32 tcap_user_ntfy
      (
            tcap_user_ntf_t *ntf_p
      );</pre></td></tr>
<tr><td>Parameters</td><td>Associated C Data Structures</td><td><pre>typedef struct
{
      tcap_qos_t qos;
      tcap_addr_t srcAdd;
      tcap_addr_t dstAdd;
      tcap_acn_t acn;
      tcap_usr_info_t usrInfo;
      tcap_bool cmpPres;
} tcap_uni_ind_t;

typedef struct
{
      tcap_u32 dlgId;
      tcap_qos_t qos;
      tcap_addr_t srcAdd;
      tcap_addr_t dstAdd;
      tcap_acn_t acn;
      tcap_usr_info_t usrInfo;
      tcap_bool cmpPres;
} tcap_begin_ind_t;

typedef struct
{
      tcap_u32 dlgId;
      tcap_qos_t qos;
      tcap_addr_t origAdd;
      tcap_acn_t acn;
      tcap_usr_info_t usrInfo;</pre></td></tr>
</table>

```
	tcap_bool cmpPres;
} tcap_continue_ind_t;

typedef struct
{
	tcap_u32 dlgId;
	tcap_qos_t qos;
	tcap_acn_t acn;
	tcap_usr_info_t usrInfo;
	tcap_bool cmpPres;
} tcap_end_ind_t;

typedef struct
{
	tcap_u32 dlgId;
	tcap_qos_t qos;
	tcap_u8 abtRsn;
	tcap_acn_t acn;
	tcap_usr_info_t usrInfo;
} tcap_u_abort_ind_t;

typedef struct
{
	tcap_u32 dlgId;
	tcap_u8 retCause;
	tcap_addr_t dstAdd;
	tcap_addr_t srcAdd;
} tcap_notice_ind_t;

typedef struct
{
	tcap_u32 dlgId;
	tcap_u8 invId;
	tcap_int_val_t linkedId;
	tcap_opcode_t opCode;
	tcap_params_t params;
	tcap_bool lastCmp;
} tcap_invoke_ind_t;

typedef struct
{
	tcap_u32 dlgId;
	tcap_u8 invId;
	tcap_opcode_t opCode;
	tcap_params_t params;
	tcap_bool lastCmp;
} tcap_result_l_ind_t;

typedef struct
{
	tcap_u32 dlgId;
	tcap_u8 invId;
	tcap_opcode_t opCode;
	tcap_params_t params;
	tcap_bool lastCmp;
```

```
} tcap_result_nl_ind_t;

typedef struct
{
      tcap_u32 dlgId;
      tcap_u8 invId;
      tcap_error_code_t ecode;
      tcap_params_t params;
      tcap_bool lastCmp;
} tcap_u_error_ind_t;

typedef struct
{
      tcap_u32 dlgId;
      tcap_u8 invId;
      tcap_problem_code_t prbCode;
      tcap_bool lastCmp;
} tcap_u_reject_ind_t;

typedef struct
{
      tcap_u32 dlgId;
      tcap_u8 invId;
} tcap_l_cancel_ind_t;

typedef struct
{
      tcap_u32 dlgId;
      tcap_u8 invId;
      tcap_bool invIdAvbl;
      tcap_problem_code_t prbCode;
      tcap_bool lastCmp;
} tcap_l_reject_ind_t;

typedef struct
{
      tcap_u32 dlgId;
      tcap_u8 invId;
      tcap_bool invIdAvbl;
      tcap_problem_code_t prbCode;
      tcap_bool lastCmp;
} tcap_r_reject_ind_t;

typedef struct
{
      tcap_u32 dlgId;
      tcap_qos_t qos;
      tcap_u8 abtRsn;
} tcap_p_abort_ind_t;

typedef struct
{
      tcap_u32 ntf_type;
      tcap_u32 nsapId;
      tcap_u32 appId;
```

		union
		{
		tcap_uni_ind_t uni;
		tcap_begin_ind_t bgn;
		tcap_continue_ind_t cnt;
		tcap_end_ind_t end;
		tcap_u_abort_ind_t u_abt;
		tcap_p_abort_ind_t p_abt;
		tcap_u_error_ind_t u_err;
		tcap_invoke_ind_t inv;
		tcap_result_l_ind_t rrl;
		tcap_result_nl_ind_t rrnl;
		tcap_l_reject_ind_t l_rjc;
		tcap_r_reject_ind_t r_rjc;
		tcap_u_reject_ind_t u_rjc;
		tcap_l_cancel_ind_t l_cnc;
		tcap_notice_ind_t ntc;
		};
		} tcap_user_ntf_t;

	ntf_type	Type of user notification reported by TCAP layer. This parameter is used to determine the valid parameters in the union. This is an unsigned 32 bit parameter with following values: • TC_UN_UNI_IND (0x01) • TC_UN_BGN_IND (0x02) • TC_UN_CNT_IND (0x03) • TC_UN_END_IND (0x04) • TC_UN_U_ABT_IND (0x05) • TC_UN_P_ABT_IND (0x06) • TC_UN_U_ERR_IND (0x07) • TC_UN_INV_IND (0x08) • TC_UN_RRL_IND (0x09) • TC_UN_RRNL_IND (0x0A) • TC_UN_L_RJC_IND (0x0B) • TC_UN_R_RJC_IND (0x0C) • TC_UN_U_RJC_IND (0x0D) • TC_UN_L_CNCL_IND (0x0E) • TC_UN_NOTICE_IND (0x0F)
	nsapId	SCCP service access point identifier for which this notification has been reported by the TCAP layer
	appId	Application identifier of the TCAP user
	Case I – If ntf_type is equal to TC_UN_UNI_IND then parameter "uni" of the union is to be interpreted. This notification is used to inform about receipt of TCAP UNIDIRECTIONAL message	
	qos	Quality of Service parameters received from SCCP
	srcAdd	Source/Calling party address received from SCCP
	dstAdd	Destination/Called party address received from SCCP
	acn	Application Context name received in the TCAP UNIDIRECTIONAL message

	usrInfo	User Information received in the TCAP UNIDIRECTIONAL message
	cmpPres	This is set to TC_TRUE if components are received along with TCAP UNIDIRECTIONAL message otherwise it is set to TC_FALSE
	Case II – If ntf_type is equal to TC_UN_BGN_IND then parameter "bgn" of the union is to be interpreted. This notification is used to inform about receipt of TCAP BEGIN message	
	dlgId	Dialogue Identifier
	qos	Quality of Service parameters received from SCCP
	srcAdd	Source/Calling party address received from SCCP
	dstAdd	Destination/Called party address received from SCCP
	acn	Application Context name received in the TCAP BEGIN message
	usrInfo	User Information received in the TCAP BEGIN message
	cmpPres	This is set to TC_TRUE if components are received along with TCAP BEGIN message otherwise it is set to TC_FALSE
	Case III – If ntf_type is equal to TC_UN_CNT_IND then parameter "cnt" of the union is to be interpreted. This notification is used to inform about receipt of TCAP CONTINUE message	
	dlgId	Dialogue Identifier
	qos	Quality of Service parameters received from SCCP
	origAdd	Source/Calling party address received in the TCAP CONTINUE message
	acn	Application Context name received in the TCAP CONTINUE message
	usrInfo	User Information received in the TCAP CONTINUE message
	cmpPres	This is set to TC_TRUE if components are received along with TCAP CONTINUE message otherwise it is set to TC_FALSE
	Case IV – If ntf_type is equal to TC_UN_END_IND then parameter "end" of the union is to be interpreted. This notification is used to inform about receipt of TCAP END message	
	dlgId	Dialogue Identifier
	qos	Quality of Service parameters received from SCCP
	acn	Application Context name received in the TCAP END message
	usrInfo	User Information received in the TCAP END message
	cmpPres	This is set to TC_TRUE if components are received along with TCAP END message otherwise it is set to TC_FALSE
	Case V – If ntf_type is equal to TC_UN_U_ABT_IND then parameter "u_abt" of the union is to be interpreted.	
	dlgId	Dialogue Identifier
	qos	Quality of Service parameters received from SCCP
	abtRsn	Abort Reason
	acn	Application Context name received in the TCAP END message
	usrInfo	User Information received in the TCAP END message
	Case VI – If ntf_type is equal to TC_UN_P_ABT_IND then parameter "p_abt" of the union is to be interpreted.	
	dlgId	Dialogue Identifier

	qos	Quality of Service parameters received from SCCP
	abtRsn	Abort Reason
	Case VII – If ntf_type is equal to TC_UN_U_ERR_IND then parameter "u_err" of the union is to be interpreted.	
	dlgId	Dialogue Identifier
	invId	Invoke Identifier
	ecode	Error code received in the return-error component
	params	Parameters portion as received in the return-error component
	lastCmp	This parameter is set to TC_TRUE if this is the last component
	Case VIII – If ntf_type is equal to TC_UN_INV_IND then parameter "inv" of the union is to be interpreted. This notification type is used for informing receipt of an invoke component.	
	dlgId	Dialogue Identifier
	invId	Invoke Identifier
	linkedId	Linked operation invoke identifier
	opCode	Operation code received in the invoke component
	params	Parameters portion as received in the invoke component
	lastCmp	This parameter is set to TC_TRUE if this is the last component
	Case IX – If ntf_type is equal to TC_UN_RRL_IND then parameter "rrl" of the union is to be interpreted. This notification type is used for informing receipt of a return-result-last component.	
	dlgId	Dialogue Identifier
	invId	Invoke Identifier
	opCode	Operation code received in the return-result-last component
	params	Parameters portion as received in the return-result-last component
	lastCmp	This parameter is set to TC_TRUE if this is the last component
	Case X – If ntf_type is equal to TC_UN_RRNL_IND then parameter "rrnl" of the union is to be interpreted. This notification type is used for informing receipt of a return-result-not-last component.	
	dlgId	Dialogue Identifier
	invId	Invoke Identifier
	opCode	Operation code received in the return-result-not-last component
	params	Parameters portion as received in the return-result-not-last component
	lastCmp	This parameter is set to TC_TRUE if this is the last component
	Case XI – If ntf_type is equal to TC_UN_L_RJC_IND then parameter "l_rjc" of the union is to be interpreted. This notification type is used for informing local reject of a component.	
	dlgId	Dialogue Identifier
	invId	Invoke Identifier
	invIdAvbl	If this parameter is set to TC_TRUE, then this parameter specifies that invoke identifier is valid. Otherwise invoke identifier does not contain valid value and must not be interpreted.

	prbCode	Problem code specifying reason for local reject of a component
	lastCmp	This parameter is set to TC_TRUE if this is the last component
	Case XII – If ntf_type is equal to TC_UN_R_RJC_IND then parameter "r_rjc" of the union is to be interpreted. This notification type is used for informing remote reject of a component.	
	dlgId	Dialogue Identifier
	invId	Invoke Identifier
	invIdAvbl	If this parameter is set to TC_TRUE, then this parameter specifies that invoke identifier is valid. Otherwise invoke identifier does not contain valid value and must not be interpreted.
	prbCode	Problem code specifying reason for remote reject of a component
	lastCmp	This parameter is set to TC_TRUE if this is the last component
	Case XIII – If ntf_type is equal to TC_UN_U_RJC_IND then parameter "u_rjc" of the union is to be interpreted. This notification type is used for informing user reject of a component.	
	dlgId	Dialogue Identifier
	invId	Invoke Identifier
	prbCode	Problem code specifying reason for user reject of a component
	lastCmp	This parameter is set to TC_TRUE if this is the last component
	Case XIV – If ntf_type is equal to TC_UN_L_CNCL_IND then parameter "l_cnc" of the union is to be interpreted. This notification type is used for informing local cancellation of an invoke component. This would occur due to invoke timer timeout.	
	dlgId	Dialogue Identifier
	invId	Invoke Identifier
	Case XV – If ntf_type is equal to TC_UN_NOTICE_IND then parameter "ntc" of the union is to be interpreted. This notification is used for informing receipt of SCCP-N-NOTICE-IND.	
	dlgId	Dialogue Identifier
	retCause	Return cause as present in SCCP UDTS/XUDTS/LUDTS message
	dstAdd	Called party address as present in SCCP UDTS/XUDTS/LUDTS message
	srcAdd	Calling party address as present in SCCP UDTS/XUDTS/LUDTS message
Restrictions/Bugs	None	
Pre-requisites	• TCAP_CONFIG_REQ • TCAP_NSAP_CONFIG_REQ	
Code Sample	Please refer source file demo/tcap.c	

16 TCAP to SCCP API

16.1 TCAP_N_UDT_REQ

Description	This API is used to pass TCAP messages to underlying SCCP protocol layer for further processing and transmission over MTP/SIGTRAN network. It is responsibility of application developer to write this API using services provided by the underlying SCCP layer.	

Prototype	`tcap_s32 tcap_n_udt_req`
	`(`
	`tcap_u32 nsapId,`
	`tcap_addr_t *srcAddr_p,`
	`tcap_addr_t *dstAddr_p,`
	`tcap_qos_t *qos_p,`
	`tcap_data_t *data_p`
	`)`	

Parameters	`nsapId`	SCCP SAP Identifier
	`srcAddr_p`	Source or Calling Party address
	`dstAddr_p`	Destination or Called Party address
	`qos_p`	SCCP Quality of Service parameters
	`data_p`	Encoded TCAP message
Restrictions/Bugs	None	
Pre-requisites	TCAP_CONFIG_REQ TCAP_NSAP_CONFIG_REQ SCCP layer is initialized and ready to accept messages	
Code Sample	Please refer source file demo/tcap.c	

17 SCCP to TCAP API

This section lists TCAP APIs that need to be invoked when an event or notification is received from the underlying SCCP layer.

17.1 TCAP_N_UDT_IND

Description	This API is used to pass SCCP N-UNITDATA-IND notification to the TCAP layer. TCAP layer then decodes and processes the received TCAP message.

Prototype	`void tcap_n_udt_ind`
	`(`
	`tcap_u32 nsap_id,`
	`tcap_u32 app_id,`
	`tcap_addr_t *cdAdd_p,`
	`tcap_addr_t *cgAdd_p,`
	`tcap_data_t *data_p,`
	`tcap_u8 imp,`
	`tcap_u8 prtCls,`

	tcap_mtp_rtlbl_t *rtLbl_p);	
Parameters	nsap_id	SCCP SAP identifier
	app_id	TCAP Application Identifier
	cdAdd_p	SCCP Called Party address
	cgAdd_p	SCCP Calling Party address
	data_p	SCCP user data or encoded TCAP message
	imp	SCCP importance parameter
	prtCls	SCCP Protocol Class
	rtLbl_p	MTP Routing Label received in the message
Restrictions/Bugs	None	
Pre-requisites	• TCAP_CONFIG_REQ • TCAP_NSAP_CONFIG_REQ	
Code Sample	Please refer source file demo/tcap.c	

17.2 TCAP_N_NOTICE_IND

Description	This API is used to pass SCCP N-NOTICE-IND to the TCAP layer.	
Prototype	void tcap_n_notice_ind (tcap_u32 nsap_id, tcap_u32 app_id, tcap_u8 retCause, tcap_addr_t *cdAdd_p, tcap_addr_t *cgAdd_p, tcap_data_t *data_p);	
Parameters	nsap_id	SCCP SAP identifier
	app_id	TCAP Application Identifier
	retCause	SCCP Return Cause
	cdAdd_p	SCCP Called Party address
	cgAdd_p	SCCP Calling Party address
	data_p	SCCP user data or encoded TCAP message
Restrictions/Bugs	None	
Pre-requisites	• TCAP_CONFIG_REQ • TCAP_NSAP_CONFIG_REQ	
Code Sample	Please refer source file demo/tcap.c	

18 Timing Management API

18.1 TCAP_ CHECK_TIMER_EXPIRY

Description	This API is used for supplying timer ticks to the TCAP library. This means that this API must be invoked at regular intervals. As TCAP timers are in range of 100ms, so it is recommended that this API is invoked at least once every 100ms.	
Prototype	`tcap_s32 tcap_check_timer_expiry(void);`	
Parameters		
Restrictions/Bugs	None	
Pre-requisites	TCAP_CONFIG_REQ	
Code Sample	Please refer source file demo/tcap.c	

19 TCAP Parameters

This section provides details about TCAP parameters that are used while invoking TCAP APIs and handling TCAP notifications.

19.1 Base Parameters

TCAP Parameter	Parameter Description
`tcap_u8`	Unsigned 8 bits parameter
`tcap_u16`	Unsigned 16 bits parameter
`tcap_u32`	Unsigned 32 bits parameter
`tcap_s8`	Signed 8 bits parameter
`tcap_s16`	Signed 16 bits parameter
`tcap_s32`	Signed 32 bits parameter
`tcap_u64`	Unsigned 64 bits parameter
`tcap_s64`	Signed 64 bits parameter
`tcap_bool`	Boolean value parameter with values set as true or false; the values for this parameter are TC_TRUE (for true) and TC_FALSE (for false)

19.2 QOS

TCAP Parameter	Parameter Description

`typedef struct`
`{`
`  tcap_bool retOpt;`
`  tcap_u16 seqCtl;`
`  tcap_u8 imp;`
`  tcap_u8 hopCtr;`
`} tcap_qos_t;`
	This parameter specifies SCCP Quality of Service parameters that are associated with a given SCCP NSAP or TCAP transaction.
	
	1. **retOpt** – SCCP Return Option; this can be set to TC_TRUE for setting “return message on error” or TC_FALSE for setting “no return of message on error”.
	2. **seqCtl** – SCCP Sequence Control Parameter; if this parameter has a valid value then SCCP protocol class 1 is chosen. For SCCP Protocol class 0, this parameter must be set to TCAP_NO_SEQCTL.
	3. **imp** – SCCP importance parameter.
	4. **hopCtr** – SCCP hop counter parameter.

19.3 SCCP Address Indicator

TCAP Parameter	Parameter Description
typedef struct { tcap_u8 natResv:1; tcap_u8 routeInd:1; tcap_u8 gtInd:4; tcap_u8 ssnInd:1; tcap_u8 pcInd:1; } tcap_sccp_addr_ind_t;	This parameter specifies SCCP address indicator parameter. 1. **natResv** – National reserve bit; this is a 1 bit parameter. 2. routeInd – Routing Indicator 3. gtInd – Global Title Indicator 4. ssnInd – SSN Indicator 5. pcInd – PC Indicator These parameters need to be given values according to the values prescribed in SCCP specifications.

19.4 Global Title

TCAP Parameter	Parameter Description
typedef struct { tcap_u8 nai; tcap_u8 es; tcap_u8 np; tcap_u8 tt; tcap_u8 numDigits; tcap_u8 digits[TC_MAX_GT_DIGITS]; } tcap_gt_t;	This parameter specifies Global Title portion of the SCCP Called/Calling address. 1. **nai** – Nature of address Indicator 2. **es** – Encoding Scheme 3. **np** – Numbering Plan 4. **tt** – Translation Type 5. **numDigits** – Number of GT Digits 6. **digits** – GT Digits These parameters need to be given values according to the values prescribed in SCCP specifications.

19.5 Address

TCAP Parameter	Parameter Description
typedef struct { tcap_sccp_addr_ind_t ai; tcap_u8 ssn; tcap_u32 pc; tcap_gt_t gt; } tcap_addr_t;	This parameter specifies the SCCP Called/Calling address. 1. **ai** – SCCP address Indicator 2. **ssn** – Sub-System Number 3. **pc –** Point Code 4. **gt** – Global Title These parameters need to be given values

	according to the values prescribed in SCCP specifications.

19.6 Protocol Version

TCAP Parameter	Parameter Description
typedef struct { tcap_bool avbl; tcap_u8 ver[TC_PRT_VER_LEN]; } tcap_prot_ver_t;	This parameter specifies the TCAP protocol versions supported. 1. **avbl** – TCAP protocol version parameter is available and should be encoded in the dialogue portion of the TCAP message 2. **ver** – TCAP protocol version encoded as bit string parameter as specified in the TCAP specification.

19.7 Application Context Name

TCAP Parameter	Parameter Description
typedef struct { tcap_bool avbl; tcap_u8 len; tcap_u8 acn[TC_ACN_SIZE]; } tcap_acn_t;	This parameter specifies the application context name to be encoded or received in the TCAP message. 1. **avbl** – TCAP ACN parameter is available and should be encoded in the dialogue portion of the TCAP message 2. **len** – Length of the ACN parameter. 3. **ver** – TCAP ACN parameter to be encoded or received in the message.

19.8 Object Identifier

TCAP Parameter	Parameter Description
typedef struct { tcap_bool avbl; tcap_u32 len; tcap_u8 objId[TC_MAX_OBJ_ID_SIZE]; } tcap_obj_id_t;	This parameter is used for object identifer parameter type 1. **avbl** – Object Identifier parameter is available 2. **len** – Length of object identifier

	parameter 3. **objId** – Object Identifier

19.9 Integer

TCAP Parameter	Parameter Description
typedef struct { tcap_bool avbl; tcap_u32 val; } tcap_int_val_t;	This parameter is used for integer parameter type 1. **avbl** – Integer parameter is available 2. **val** – integer value

19.10 Result

TCAP Parameter	Parameter Description
typedef struct { tcap_bool avbl; tcap_u32 result; } tcap_rslt_t;	This parameter specifies TCAP result parameter 1. **avbl** – Result parameter is available 2. **result** – value of result parameter

19.11 Result Source Diagnostic

TCAP Parameter	Parameter Description

typedef struct
{
tcap_bool avbl;
tcap_u32 rsltSrcDgnType;
union
{
tcap_u32 dlgSvcUsrDgnVal;
tcap_u32 dlgSvcPrvDgnVal;
};
} tcap_rslt_src_dgn_t;
	This parameter specifies TCAP result source diagnostic parameter
	
	1. **avbl** – Result source diagnostic parameter is available
	2. **rsltSrcDgnType** – type of result source diagnostic value
	1. **TC_RSL_SRC_DLG_SVC_USR** (0x01) – Dialogue Service User
	2. **TC_RSL_SRC_DLG_SVC_PRV** (0x02) – Dialogue Service Provider
	3. **dlgSvcUsrDgnVal** – Diagnostic value when **rsltSrcDgnType** is set to **TC_RSL_SRC_DLG_SVC_USR**.
	4. **dlgSvcPrvDgnVal** – Diagnostic value when

	rsltSrcDgnType is set to **TC_RSL_SRC_DLG_SVC_PRV**.

19.12 Object Descriptor

TCAP Parameter	Parameter Description
typedef struct { tcap_bool avbl; tcap_u32 len; tcap_u8 objDesc[TC_MAX_OBJ_DESC_SIZE]; } tcap_obj_desc_t;	This parameter is used for object descriptor parameter type 1. **avbl** – Object descriptor parameter is available 2. **len** – Length of object descriptor parameter 3. **objDesc** – Value of object descriptor parameter.

19.13 User Information

TCAP Parameter	Parameter Description

typedef struct
{
tcap_bool avbl;
tcap_obj_id_t objId;
tcap_int_val_t idrRef;
tcap_obj_desc_t objDesc;
tcap_u32 encLen;
tcap_u8 encType;
tcap_u8 tagForm;
tcap_u8 encInfo[TC_MAX_USR_INFO_SIZE];
} tcap_usr_info_t;
	This parameter is used for User Information parameter type
	
	1. **avbl** – User information parameter is available
	2. **objId** – Object Identifier portion of user information parameter
	3. **idrRef** – Indirect reference
	4. **objDesc** – Object Descriptor parameter of user information
	5. **encLen** – Length of encoding
	6. **encType** – Type of encoding
	7. **tagForm** – Tag form for encoding; it is set to either of the following values
	    1. **TC_TAG_FORM_PRIMITIVE (0x00)** – Primitive Tag Form
	    2. **TC_TAG_FORM_CONSTRUCTOR (0x01)** – Constructor Tag Form
	8. **encInfo** – Encoding/Encoded information

19.14 Operation Code

TCAP Parameter	Parameter Description
typedef struct { tcap_u8 avbl; tcap_u8 type; tcap_u32 opCode; } tcap_opcode_t;	This parameter is used for operation code parameter type 1. **avbl** – Operation Code parameter is available 2. **type** – Type of operation code; this would contain either of the following value 1. TC_OPCODE_TYPE_LCL (0x00) – Local Operation Code 2. TC_OPCODE_TYPE_GBL (0x01) – Global Operation Code 3. **opCode** – value of operation code

19.15 Parameters Portion of TCAP Components

TCAP Parameter	Parameter Description
typedef struct { tcap_u8 avbl; tcap_u32 len; tcap_u8 params[TC_MAX_PARAMS_LEN]; } tcap_params_t;	This data structure contains parameters portion of the TCAP components. This includes whole body of the parameters portion, including tag, length and value. 1. **avbl** – parameter portion of TCAP component is available 2. **len** – Length of parameter portion 3. **params** – Entire parameters portion including Tag, Length & Value. This is passed transparently between TCAP & TCAP user.

19.16 Problem Code

TCAP Parameter	Parameter Description
typedef struct { tcap_u8 avbl; tcap_u8 type; tcap_u32 code; } tcap_problem_code_t;	This parameter is used for problem code parameter type 1. **avbl** – Problem Code parameter is available 2. **type** – Type of problem code; this would contain either of the following value 1. TC_PROB_TYPE_GEN (0x00) – General Problem Type

	2. TC_PROB_TYPE_INVK (0x01) – Invoke Problem Type 3. TC_PROB_TYPE_RR (0x02) – Return-Result Problem Type 4. TC_PROB_TYPE_RE (0x03) – Return-Error Problem Type 5. TC_PROB_TYPE_LCL (0x08) – Local Reject Problem Type 2. **code** – Problem Code; for values of this parameter, please refer to TCAP specification. When Local Reject problem type is reported, then either of following values is used: 1. TC_LCL_RJCT_INV_RJCT_CMP – Invalid reject component 2. TC_LCL_RJCT_UNRCG_INVKID – Unrecognized invoke Identifier

19.17 Error Code

TCAP Parameter	Parameter Description

`typedef struct`
`{`
`  tcap_u8 avbl;`
`  tcap_u8 type;`
`  tcap_u32 code;`
`} tcap_problem_code_t;`
	This parameter is used for error code parameter type
	
	1. **avbl** – Error Code parameter is available
	2. **type** – Type of error code; this would contain either of the following value
	1. TC_ERCD_TYPE_LCL (0x00) – Local Error Type
	2. TC_ERCD_TYPE_GBL (0x01) – Global Error Type
	3. **code** – Value of Error Code

19.18 Data

TCAP Parameter	Parameter Description

`typedef struct`
`{`
`  tcap_u32 len;`
`  tcap_u8 *data_p;`
`} tcap_data_t;`
	This parameter is used for TCAP data parameter type
	
	1. **len** – Length of TCAP data
	2. **data_p** – Pointer to memory containing TCAP data

20 Integrating TCAP with MTP/M3UA, SCTP and TCAP User

The application developer would need to integrate the TCAP layer with the layer management interface to configure TCAP layer. First, MTP/M3UA and SCTP would be configured & made active, followed by SCCP layer followed by TCAP and then other TCAP based applications like MAP or CAP. Layer management entity would also handle various notifications from TCAP layer and translate them into relevant information for TCAP users and operator.

A wrapper code (light weight application) around TCAP layer shall be written to adapt TCAP primitives & notifications into format understood by the TCAP users. This wrapper code would pass primitives between TCAP users [example MAP or CAP] and TCAP. Similarly, a thin wrapper layer between TCAP and SCCP would be required to facilitate passing of messages between SCCP and TCAP layers.

This page is intentionally left blank.