SCCP Programmer's Manual

Version 1.0.2

Table of Contents

S. No.	Version	Date	Comments
1	0.0.1	02-Feb-2011	Initial Draft
2	1.0.1	04-Mar-2011	First customer delivery
3	1.0.2	17-May-2011	More clarification for SCCP Connectionless API, GTT Description
4	1.0.3	02/25/12	SSN added as a selection parameter for GTT rule set selection [Customer Requirement MDOO27012012]

1 Introduction

The basis of SCCP protocol implementation is ITU-T Specification Series Q.711-Q.719. Presently SCCP software implementation supports all the connection-less services and notifications. The SCCP implementation also has support of all the mandatory ANSI T1.112 connection less services.

This programmer's manual explains various API Primitives (Application Programming Interface Primitives) provided by SCCP protocol implementation (referred as just SCCP 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 SCCP sample application to understand the overall functioning of the SCCP library. The aim of this manual is to explain all the API services of the SCCP library. The sample application puts them into use and thus serves as a live working example for SCCP 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 "`g_sccp_errno`" is set to the respective error code.

The SCCP library uses C Programming macro "`SP_ERRNO`" 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 SCCP layer.

The macro "`SP_ERRNO`" is written in source file "sperr.h". Presently, this macro generates an error trace with file name and line number information.

3 SCCP Library Installation

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

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

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

3.1 Building the SCCP library

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

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

2. Type `` `make' `` to build the library libsccp.a. This library is present in '`./sccp/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, then please install CL-SCCP source code in the same directory where M3UA 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]`.

3.2 Building the sample application

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

1. Change to parent installation directory [./sccp] containing the source code for SCCP. Now change to directory `` `./sccp/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 various SCCP functionalities Global Title Translation support, XUDT & XUDTS support, segmentation and reassembly support etc.

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 SCCP to compile using any standard C/C++ compiler on any Operation System with very minimal porting effort.

4 Concept of MTP-SAP-ID

MTP-SAP-ID [MTP Service Access Point Identifier] is a logical interface between MTP/SIGTRAN layer and SCCP using which they communicate with each other. An MTP-SAP is a collection of various parameters like OPC, Network Standard, Network Indicator, Maximum PDU size allowed etc. This allows multiple point codes to be present on the same physical network node. For example a gateway STP node is part of two networks, national network and international network; then this gateway STP node would be having two OPC and accordingly two MTP-SAP needs to be created at SCCP layer.

Every point code is part of a particular MTP network. So, a point code resource is identified with a combination of PC and MTP-SAP-ID. A SCCP local subsystem or a remote subsystem is logically part of a particular MTP network node. Every MTP network node is uniquely identified with a combination of PC and MTP-SAP-ID. So, a SCCP subsystem is identified using a combination of SSN, PC and MTP-SAP-ID.

5 Concept of SCCP-SAP-ID

An SCCP SAP [SCCP Service Access Point] is a logical interface between SCCP layer and a local subsystem that would be taking service from the SCCP layer. In case a local subsystem needs to communicate with remote subsystems present in more that one MTP networks, then more than one SCCP-SAP need to be created for that particular local subsystem.

For example an HLR subsystem [SSN 6] needs to communicate with several operator networks in national and MTP networks in other countries. So, we need to create two SCCP SAP for HLR subsystem. One of the SCCP SAP would be used for communication within the national network and other SCCP SAP would be used for communication through international MTP network. [Please note this is just an example and it is possible to have just one SCCP SAP and with the help of GTT routing, appropriate MTP-SAP may be selected.]

6 Global Title Translation

Sure Speed SCCP is equipped with a very powerful and easy to use Global Title Translation engine. Global Title Translation is generally used whenever messages are to be routed across more than 1 MTP networks. However, it may also be used within a MTP network. A Global Title is a string of numeric digits with a length of up to 15 digits. There are various properties associated with these digits such as Nature of Address Indicator, Numbering Plan, Encoding Scheme etc. For more details on various Global Title formats, please refer ITU-T Q.713 specification.

A GTT can be easily configured by creating a GTT-Rule-Set and subsequently adding various GTT-Rules to it. Figure 1 shows the flow chart of configuring a GTT-Rule-Set.

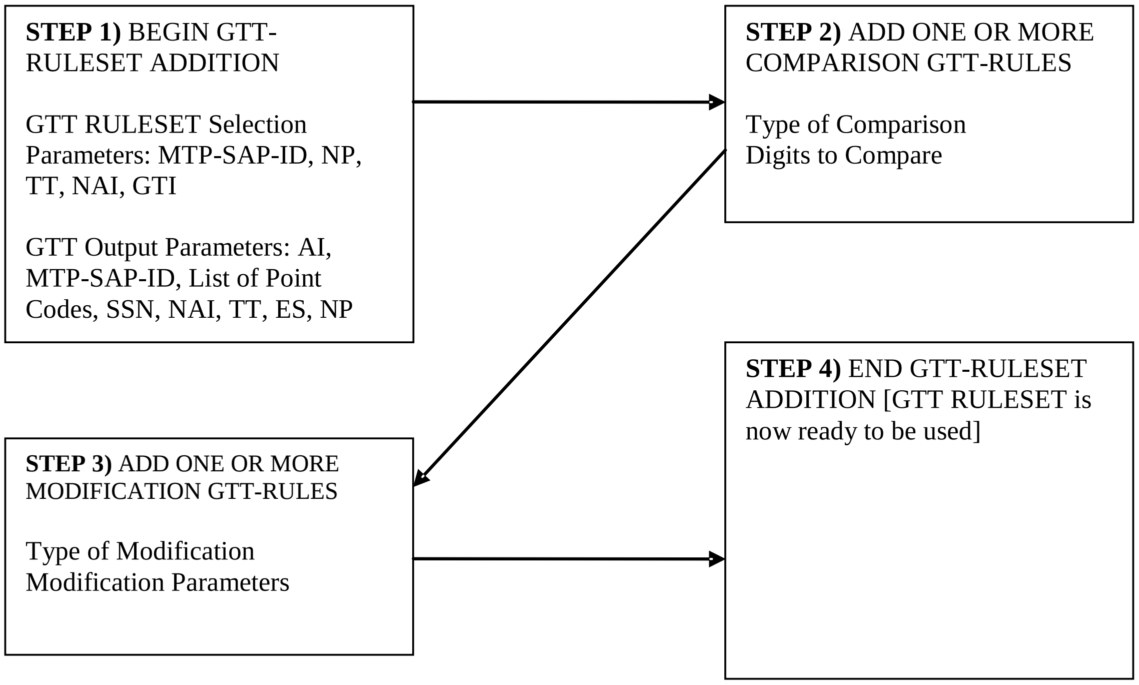

STEP 1) BEGIN GTT-RULESET ADDITION
GTT RULESET Selection Parameters: MTP-SAP-ID, NP, TT, NAI, GTI
GTT Output Parameters: AI, MTP-SAP-ID, List of Point Codes, SSN, NAI, TT, ES, NP
STEP 2) ADD ONE OR MORE COMPARISON GTT-RULES
Type of Comparison
Digits to Compare
STEP 3) ADD ONE OR MORE MODIFICATION GTT-RULES
Type of Modification
Modification Parameters
STEP 4) END GTT-RULESET ADDITION [GTT RULESET is now ready to be used]

Figure 1: GTT Rule-Set Configuration Steps

In the following sub-sections we will have a look at each of the steps mentioned in Figure 1.

6.1 Begin GTT-Rule-Set Addition

The first step to create a Global Title Translation logic is to begin the GTT-Rule-Set addition. During this step, we input two kinds of parameters:

1. **GTT Rule-Set Selection Parameters:** For any “Route on GTT” based called party address, a GTT rule-set is selected based on following criteria:

a. **MTP-SAP-ID** [In case message is received from a SCCP-SAP, then associated MTP-SAP-ID is used; if message is received from MTP/SIGTRAN layer, then the corresponding MTP-SAP-ID is used]
b. **Global Title Indicator [GTI]** and associated parameters like NAI, NP and TT. For example: with GTI type 1, only NAI is used as a selection criteria; for GTI type 4, NAI, NP and TT are used as selection criteria.
c. **SSN** may also be used as a selection criteria. The SSN in selection parameter may be configured with valid value or invalid value (SP_INV_SSN).
    - When a valid SSN is configured while adding a GTT Rule-Set, then this SSN value would always be used to decide if this GTT Rule-Set can be used to perform SCCP called party address GTT. This means that SCCP called party SSN MUST be equal to the SSN in the Rule-Set selection parameters.
    - If a SCCP Called Party address does not contain SSN, then it cannot use GTT rule-sets with valid SSN in selection parameters.
    - GTT Rule-Sets having SSN set to SP_INV_SSN in selection parameters may be used to perform GTT of SCCP called party address containing valid SSN.
    - GTT rule-sets with valid and invalid SSN may coexist together. This means that it is possible to configure some GTT rule-sets with selection SSN set to SP_INV_SSN and some GTT rule-sets with selection SSN set to valid value (6, 7, 8 etc.).
    - When similar GTT rule-sets with different selection SSN are configured, then selection of GTT rule-set would be based on SSN value. Following is an example of GTT selection:

GTT Rule-Set ID	**Selection Parameters**
1	GTI:4, TT:0, NAI:4, NP:1, MTP-SAP-ID:1, SSN=8, GTAI=1234567XXX
2	GTI:4, TT:0, NAI:4, NP:1, MTP-SAP-ID:1, SSN=7, GTAI=1234567XXX
3	GTI:4, TT:0, NAI:4, NP:1, MTP-SAP-ID:1, SSN=SP_INV_SSN, GTAI=1234567XXX
In this case if SCCP Called Party address consists of SSN=7 and GTAI=1234567890, then rule-set 2 would be selected.	

- It is responsibility of the SCCP provisioning entity to take care that conflicting GTT Rule-Sets are not configured simultaneously. Following is an example of conflicting GTT Rule-Sets:

GTT Rule-Set ID	Selection Parameters
1	GTI:4, TT:0, NAI:4, NP:1, MTP-SAP-ID:1, SSN=SP_INV_SSN, GTAI=1234567XXX
2	GTI:4, TT:0, NAI:4, NP:1, MTP-SAP-ID:1, SSN=7, GTAI=12345XXXXX
In this case if SCCP Called Party address consists of SSN=7 and GTAI=1234567890, then rule-set 1 would be selected, however, the desired rule-set may be 2. So, it is responsibility of provisioning entity to avoid such a configuration.	

There may be more than one GTT-Rule-Set with same GTT-Rule-Set selection parameters. The selection of actual GTT-Rule-Set depends on the Comparison Rules configured within the GTT-Rule-Set. We will discuss the GTT-Comparison-Rules in the subsequent section.

6.2 Configuring GTT-Comparison-Rules

Within a GTT-Rule-Set, there is a sequence of one or more GTT-Comparison-Rules. The sequence of GTT-Comparison-Rules is used to compare the input GT digits to see if present Rule-Set is suitable for it. Once the input GT digits match with the digits specified in the GTT-Comparison-Rules, the GTT-Rule-Set is considered suitable for the Global Title Translation.

There are various comparison criteria available while configuring a GTT-Comparison-Rule as following:

1. Match First N Digits: Check if first N digits of input called party GT are same as comparison digits
2. Match Last N Digits: Check if last N digits of input called party GT are same as comparison digits
3. Match Middle N Digits: Check if middle N digits of input called party GT are same as comparison digits
4. Match All Digits: Check if all the digits of input called party GT are same as comparison digits

5. Match Number of Digits: Check if number of digits in input called party GT is same as number of comparison digits
6. Match No Digits: This rule always returns true as no comparison is required. This comparison rule may be used when all the “Route-On-GT” based called party addresses need to be routed to same destination node. This destination node may be the actual GTT performing node.

Let’s take some example to understand the logic of comparison rules better:

Input GT Digits	Type of Comparison	Number of Comparison Digits	Comparison Digits	Comparison Result
12345XXXXXX	Compare First N Digits	5	12345	TRUE
543210XXXXX	Compare First N Digits	6	12345	FALSE
XXXXX543210	Compare Last N Digits	6	543210	TRUE
XXXXX756443	Compare Last N Digits	6	756433	FALSE
XXXX123345X…X	Compare Middle N Digits	10	FFFF123345 NOTE: “FFFF” in beginning means that we need to skip first 4 digits and then compare next 6 Digits.	TRUE
X….X83274XXX	Compare Middle N Digits	8	83274FFF NOTE: “FFF” in end means that we need to skip last 3 digits and then compare next 5 Digits.	TRUE
123456789	Compare All Digits	9	123456789	TRUE
1234567890	Compare All Digits	9	123456789	FALSE
1234567890	Match Number of Digits	10	X	TRUE
XXXXXXXXXX	Match No Digits	X	X	TRUE

Table 1: Examples of GTT Comparison Rules

Following example explains how a sequence of GTT-Comparison-Rules works:

Input GTT Digits	Rule 1	Rule 2	Rule 3	Result
12345678909	Rule Type: Match Number of Digits	Rule Type: Match First N Digits	Rule Type: Match Last N Digits	TRUE
	Number of Comparison Digits: 11	Number of Comparison Digits: 6	Number of Comparison Digits: 3	
	X	123456	909	
12345678909	Rule Type: Match Number of Digits	Rule Type: Match First N Digits	Rule Type: Match Last N Digits	FALSE
	Number of Comparison Digits: 11	Number of Comparison Digits: 6	Number of Comparison Digits: 4	
	X	123456	1327	

Table 2: Example of Sequence of GTT Comparison Rules

6.2.1 GTT Comparison Score

While selecting an appropriate GTT-Rule-Set, the rule-set with best possible comparison is selected. Let's take an example to understand this better:

Input GTT Digits	Rule 1	Rule 2	Result
123456789091	**Rule-Set 1**		
	Rule Type: Match Number of Digits	Rule Type: Match First N Digits	This Rule-set is not selected as comparison score is 6 [A score of 1 is given for Matching No. of Digits + A score of 5 is given for matching of 5 comparison digits]
	Number of Comparison Digits: 11	Number of Comparison Digits: 5	
	X	12345	
	Rule-Set 2		
	Rule Type: Match Number of Digits	Rule Type: Match First N Digits	This Rule-set is selected as comparison score is 8 [A score of 1 is given for Matching No. of Digits + A score of 7 is given for matching of 7 comparison digits]
	Number of Comparison Digits: 11	Number of Comparison Digits: 6	
	X	1234567	

So,

Rule-Set 2 is selected for Global Title Translation as it has higher comparison score.

6.3 Configuring GTT Modification Commands

Once a GTT-Rule-Set has been configured with sequence of GTT comparison rules, we need to add a sequence of GTT-Modification-Commands containing one or more modification rules. These rules state the modification that would be done over the input GT digits. The modification of input GT digits takes place only when comparison rules are successful.

There are a number of modifications that may be carried out over the input GT digits [called party GT digits]:

1. Apply New GT: replace the input GT digits with entirely new GT digits
2. Replace First N Digits: replace first N digits of input GT
3. Replace Last N Digits: replace last N digits of input GT
4. Replace Middle N Digits: replace middle N digits of input GT
5. Delete First N Digits: remove first N digits of input GT
6. Delete Last N Digits: remove last N digits of input GT
7. Delete Middle N Digits: remove middle N digits of input GT
8. Add Digits in Beginning: add digits to beginning of input GT
9. Add digits to End: add digits to end of input GT
10. Add digits in Middle: add digits in middle of input GT
11. Do Nothing: Do not make any changes to input GT

Let's take some examples in the following table to understand how modification commands work:

Input GT Digits	Rule Type	Number of GT Digits	Number of Modification Digits	Modification Digits	Output GT Digits
1234567890	Apply New GT	X	12	987654321021	987654321021
1234567890	Replace First N digits	4	5	32713	32713567890
1234567890	Replace Last N digits	5	4	9384	123459384
1234567890	Replace Middle N digits	4	8	FFFF3274 NOTE: "FFFF" in the beginning means that we need to skip first 4 digits of input GT and then replace next 4 digits with "3274".	1234327490
1234567890	Replace Middle N digits	3	7	4363FFF NOTE: "FFF" in the end means that we need to	12344363890

				skip last 3 digits of input GT and then replace next 3 digits with "4363".	
1234567890	Delete First N Digits	2	X	X	34567890
1234567890	Delete Last N Digits	3	X	X	1234567
1234567890	Delete Middle N Digits	2	3	FF0 NOTE: "FF" in the beginning means that we need to skip first 2 digits of input GT and then delete next 2 digits	12567890
1234567890	Delete Middle N Digits	3	3	0FF NOTE: "FF" in the end means that we need to skip last 2 digits of input GT and then delete next 3 digits	1234590
1234567890	Add Digits in Beginning	X	3	932	9321234567890
1234567890	Add Digits in End	X	4	3138	12345678903138
1234567890	Add Digits in Middle	X	5	FF547 NOTE: "FF" in the beginning means that we need to skip first 2 digits of input GT and then add "547"	1254734567890
1234567890	Add Digits in Middle	X	4	0FFF NOTE: "FFF" in the end means that we need to skip last 3 digits of input GT and then add "0"	12345670890
1234567890	Do Nothing	X	X	X	1234567890

Table 3: Examples of GTT Modification Commands

Following table shows how a sequence of modification commands within a GTT-Rule-Set would work:

Input GTT Digits	**Rule 1**	**Rule 2**	**Result**
1234567890	Rule Type: Replace First N Digits	Rule Type: Replace Last N Digits	4387456039
	Number of GT Digits: 3	Number of GT Digits: 4	

	Number of Modification Digits: 4	Number of Modification Digits: 3	
	Modification Digits: 4387	Modification Digits: 039	

Table 4: Example of GTT Modification Commands Sequence

6.4 Finalizing GTT Rule-Set Configuration

This step is performed after all the required comparison and modification rules have been configured. Once this step has been performed, we cannot add any more rules to the GTT-Rule-Set. Also, this step makes GTT-Rule-Set available for use during future global title translations.

6.5 GTT Generic Rule-Sets

We have created a concept of generic GTT Rule-Sets. You may configure only one generic GTT rule-set per MTP-SAP. However, configuring a generic GTT rule-set is also not necessary. A generic GTT rule set is selected only when no suitable GTT-rule-set is found for a “Route-On-GT” based called party address. A generic GTT rule-set may also be used to forward all the “Route-On-GT” based messages to another network node that performs actual Global Title Translations.

For configuring a GTT rule-set, selection criteria is based on just the MTP-SAP-ID and only the destination MTP-SAP and Point Codes need to be mentioned. The input message [with “Route-on-GT” based Called Party Address] is forwarded as it is to the network node identified with Point codes mentioned in the configured generic GTT rule-set.

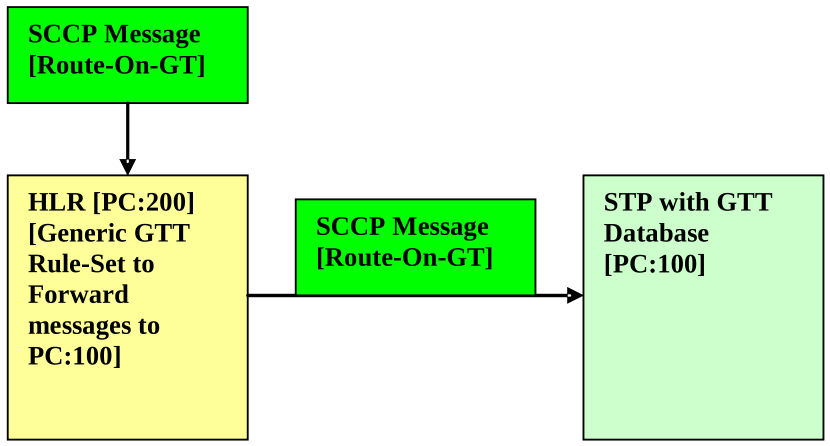

SCCP Message [Route-On-GT]
HLR [PC:200] [Generic GTT Rule-Set to Forward messages to PC:100]
SCCP Message [Route-On-GT]
STP with GTT Database [PC:100]

Figure 2: Example of Generic GTT Rule-Set

7 Global Title Translation API

This section lists the API that enables application developers to configure global title translation rules within the SCCP layer.

7.1 SCCP_GTT_BEGIN_RULESET_ADDITION

Description	This API is used to begin addition of a GTT-rule-set in the SCCP library. The name rule-set has been used as it is collection of more than one GTT-rule. A GTT-rule is used to either compare the input GT digits or perform modification over the input GT digits. Apart from GTT-rules, GTT-rule-set also has a group of parameters which are used for its selection whenever a GTT has to be performed. For example a message with global title indicator 0x04 and translation type 10 is received, then a suitable GTT-rule-set configured with global title indicator 0x04 and translation type 10 would be selected. Please refer section “Global Title Translation” on details about GTT rule-set. Please note that a rule-set must contain at least one comparison rule and one modification rule. Also, SCCP_GTT_END_RULESET_ADDITION must be invoked to make the GTT-rule-set usable.	

Prototype	`sccp_s32 sccpGTT_beginRuleSetAddition`
	`(`
	`sccp_u32 ruleSetId,`
	`spGttRuleSet_t *ruleSet_p`
	`);`	

	Associated C Data Structures	`typedef struct`
		`{`
		`spaddrInd_t ai;`
		`sccp_u16 mtp_sap_id;`
		`sccp_u16 num_pc;`
		`sccp_u32 pc[SP_MAX_GTT_PC];`

		`sccp_u8 ssn;`
		`sccp_u8 nai;`
		`sccp_u8 es;`
		`sccp_u8 np;`
		`sccp_u8 tt;`
		`} spGttOutParams_t;`
		
		`typedef struct`
		`{`
		`sccp_u8 gti;`
		`sccp_u8 nai;`
		`sccp_u8 tt;`
		`sccp_u8 np;`
		`sccp_u16 mtp_sap_id;`
		`sccp_u8 ssn;`
		`spGttOutParams_t outParams;`
		`} spGttRuleSet_t;`

	`ruleSetId`	GTT-Rule-Set identifier, this value needs to be between `SP_MIN_GTT_RULESETID` and `SP_MAX_GTT_RULESETID`.
	`gti`	Global Title Indicator that would be used for selection of this GTT-rule-set
	`nai`	Nature of Address Indicator that would be used for selection of this GTT-rule-set
	`tt`	Translation Type that would be used for selection of this GTT-rule-set
	`np`	Numbering Plan that would be used for selection of this GTT-rule-set
	`mtp_sap_id`	MTP-SAP identifier that would be used for selection of this GTT-rule-set
	`ssn`	SSN that would be used for selection of this GTT-rule-set. The SSN value may be set to invalid SSN (SP_INV_SSN or 0x00) value also.
	`outParams`	Once an SCCP called party address goes through successful global title translation, the parameters specified in this structure are used to build the new SCCP called party address.
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT	
Code Sample	Please refer source file demo/main.c	

7.2 SCCP_GTT_ADD_RULE

Description	This API is used to add a GTT rule to an existing GTT-rule-set. A GTT-rule cannot be added to a closed GTT-rule-set. A GTT-rule is used for two purposes; 1. **Comparison of input GT digits** – The GTT rule specifies the criteria based on which input GT digits are compared with the

	configured comparison digits. Once the comparison is successful, the GTT-rule-set may be considered for performing final global title translation of the input called party address. 2. **Modification of input GT digits** – The GTT rule specifies the modification that is to be performed over the input GT digits. The modification commands are executed only when the GTT-rule-set has been selected. Please refer section “Global Title Translation” on details about GTT rules.	

Prototype	`sccp_s32 sccpGTT_addRule`
	`      (`
	`            sccp_u32 ruleId,`
	`            spGttRule_t *rule_p`
	`      );`	

	Associated C Data Structures	`typedef enum`
		`{`
		`      SP_GTT_RULETYPE_COMP,`
		`      SP_GTT_RULETYPE_MOD,`
		`      SP_GTT_RULETYPE_INV`
		`} spGttRuleType;`
		
		`typedef enum`
		`{`
		`      SP_MATCH_FIRST_N_DIG,`
		`      SP_MATCH_LAST_N_DIG,`
		`      SP_MATCH_MID_N_DIG,`
		`      SP_MATCH_ALL_DIG,`
		`      SP_MATCH_NUM_DIG,`
		`      SP_MATCH_NO_DIG`
		`} spGttCompRuleType;`
		
		`typedef enum`
		`{`
		`      SP_DO_NOTHING,`
		`      SP_APPLY_NEW_GT,`
		`      SP_REP_FIRST_N_DIG,`
		`      SP_REP_LAST_N_DIG,`
		`      SP_REP_MID_N_DIG,`
		`      SP_DEL_FIRST_N_DIG,`
		`      SP_DEL_LAST_N_DIG,`
		`      SP_DEL_MID_N_DIG,`
		`      SP_ADD_TO_BEG,`
		`      SP_ADD_TO_END,`
		`      SP_ADD_IN_MID`
		`} spGttModCmdType;`
		
		`typedef struct`
		`{`
		`      spGttCompRuleType type;`
		`      sccp_u32 Ncd;`
		`      sccp_u8`

		compDigits[SP_MAX_COMP_DIG];
		} spGttCompRule_t;
		
		typedef struct
		{
		        spGttModCmdType type;
		        sccp_u32 Ngtd;
		        sccp_u32 Nmd;
		        sccp_u8
		modDigits[SP_MAX_MOD_DIG];
		} spGttModCmd_t;
		
		typedef struct
		{
		        spGttRuleType type;
		        sccp_u32 ruleSetId;
		        union
		        {
		                spGttCompRule_t compRule;
		                spGttModCmd_t modCmd;
		        };
		} spGttRule_t;

	ruleId	Rule identifier, this value needs to be between 0 and less than SP_MAX_GTT_RULES.
	type	Type of GTT-rule. Valid values are: SP_GTT_RULETYPE_COMP – Comparison Rule SP_GTT_RULETYPE_MOD – Modification Rule
	ruleSetId	GTT-Rule-Set Identifier to which this GTT-Rule has to be added.
	Case I – Configuring a Comparison Rule. In this case parameters of structure “compRule” need to be filled.	
	type	Type of comparison rule. Valid values for this parameter are: SP_MATCH_FIRST_N_DIG – Match first N digits of input GT with the comparison digits SP_MATCH_LAST_N_DIG – Match last N digits of input GT with the comparison digits SP_MATCH_MID_N_DIG – Match middle N digits of input GT with the comparison digits SP_MATCH_ALL_DIG – Match all digits of input GT with the comparison digits SP_MATCH_NUM_DIG – Check that number of digits of input GT is same as number of comparison digits SP_MATCH_NO_DIG – Don’t match input GT digits
	Ncd	Number of comparison digits
	compDigits	Comparison Digits; each of this digits must be in range of 0 to 9. It can be 0x0F in case comparison of middle digits is to be done. Please refer Table 1 for more details.
	Case II – Configuring a Modification Rule. In this case parameters of	

	structure “`modCmd`” need to be filled.	
	`type`	Type of modification to be performed over the input GT digits. Valid values for this parameter are: `SP_DO_NOTHING` – Don’t perform any modification to the input GT digits `SP_APPLY_NEW_GT` – Replace input GT digits with new set of GT digits as mentioned in the modification digits `SP_REP_FIRST_N_DIG` – Replace first `Ngtd` number of input GT digits with the modification digits `SP_REP_LAST_N_DIG` – Replace last `Ngtd` number of input GT digits with the modification digits `SP_REP_MID_N_DIG` – Replace middle `Ngtd` number of input GT digits with the modification digits `SP_DEL_FIRST_N_DIG` – Delete first `Ngtd` number of input GT digits `SP_DEL_LAST_N_DIG` – Delete last `Ngtd` number of input GT digits `SP_DEL_MID_N_DIG` – Delete middle `Ngtd` number of input GT digits `SP_ADD_TO_BEG` – Add modification digits to beginning of input GT digits `SP_ADD_TO_END` – Add modification digits to end of input GT digits `SP_ADD_IN_MID` – Add modification digits in middle of input GT digits Please refer Table 3 for examples on how modification rules work.
	`Ngtd`	Number of Input Global Title digits to be removed or replaced
	`Nmd`	Number of modification digits
	`modDigits`	Modification digits
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT, SCCP_GTT_BEGIN_RULESET_ADDITION	
Code Sample	Please refer source file demo/main.c	

7.3 SCCP_GTT_END_RULESET_ADDITION

Description	This API is used to end addition of a GTT-rule-set. The GTT-rule-set is now ready to use. Please note that at least one comparison rule and one modification rule must be present in a GTT rule-set.

Prototype	`sccp_s32 sccpGTT_endRuleSetAddition`
	`(`
	`sccp_u32 ruleSetId`
	`);`

	Associated C Data Structures	
	`ruleSetId`	GTT-Rule-Set identifier that has already been added to SCCP layer using API `SCCP_GTT_BEGIN_RULESET_ADDITION`.
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT, SCCP_GTT_BEGIN_RULESET_ADDITION, SCCP_GTT_ADD_RULE	
Code Sample	Please refer source file demo/sccp.c	

7.4 SCCP_GTT_DELETE_RULESET

Description	This API is used to delete an already configured GTT-rule-set.	

Prototype	`sccp_s32 sccpGTT_Del_RuleSet`
	`(`
	`sccp_u32 ruleSetId`
	`);`	

	Associated C Data Structures	
	`ruleSetId`	GTT-Rule-Set identifier that has already been added to SCCP layer using API `SCCP_GTT_BEGIN_RULESET_ADDITION`.
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT SCCP_ADD_MTP_SAP SCCP_GTT_BEGIN_RULESET_ADDITION SCCP_GTT_ADD_RULE SCCP_GTT_END_RULESET_ADDITION	
Code Sample	Please refer source file demo/main.c	

7.5 SCCP_GTT_ADD_GENERIC_RULESET

Description	This API is used to add a generic GTT rule-set for a particular MTP-SAP. If no suitable GTT rule-set is found for a called party address with “GT based routing”, then generic rule-set is used to forward that particular SCCP message to another network node where actual GTT would take place. A generic GTT rule-set is only used when no other suitable GTT rule-set is found. Also, it is not necessary to configure a generic GTT rule-set. Please refer section “Global Title Translation” on details about generic GTT rule-set.

Prototype	`sccp_s32 sccpGTT_addGenericRuleSet`
	`(`
	`spGttRuleSet_t *ruleSet_p`
	`);`

	Associated C Data Structures	`typedef struct`
		`{`
		`        spaddrInd_t ai;`
		`        sccp_u16 mtp_sap_id;`
		`        sccp_u16 num_pc;`
		`        sccp_u32 pc[SP_MAX_GTT_PC];`
		`        sccp_u8 ssn;`
		`        sccp_u8 nai;`
		`        sccp_u8 es;`
		`        sccp_u8 np;`
		`        sccp_u8 tt;`
		`} spGttOutParams_t;`
		
		`typedef struct`
		`{`
		`        sccp_u8 gti;`
		`        sccp_u8 nai;`
		`        sccp_u8 tt;`
		`        sccp_u8 np;`
		`        sccp_u16 mtp_sap_id;`
		`        spGttOutParams_t outParams;`
		`} spGttRuleSet_t;`

	`mtp_sap_id`	MTP-SAP identifier for which this generic GTT rule-set is applicable. Please note that only this parameter is used as selection criteria for a generic GTT-rule-set. Other parameters like gti, nai, tt and np are ignored and thus, application developer does not need to fill them.
	`outParams`	Out of all the output parameters only number of point codes (`num_pc`) and point codes list (`pc[]`) are used for the generic GTT-rule-set. All other parameters of this structure are ignored.
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT, SCCP_ADD_MTP_SAP	
Code Sample	Please refer source file demo/main.c	

7.6 SCCP_GTT_DELETE_RULESET

Description	This API is used to delete an already configured GTT-rule-set.	

Prototype	`sccp_s32 sccpGTT_delGenericRuleSet`
	`      (`
	`            sccp_u16 mtp_sap_id`
	`      );`	

	Associated C Data Structures	
	`mtp_sap_id`	MTP-SAP identifier for which generic GTT rule-set has to be deleted.
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT	

	SCCP_ADD_MTP_SAP SCCP_GTT_ADD_GENERIC_RULESET
Code Sample	Please refer source file demo/main.c

8 Example Scenarios & Sequence of API calls

This section lists example scenarios where SCCP protocol may be used. The SCCP protocol supports almost all the possible connection-less scenarios and this section lists a sub-set of those scenarios.

8.1 Communication between Remote SCCP Subsystem using SSN based Routing

Following figure shows the case where SCCP would be used for connection-less SS7 signaling between two or more remote SCCP subsystems using SSN based Routing. In present example, all the three nodes are part of the same national MTP network. HLR has point code 30, two MSC's have point codes of 10 and 20. The SCCP Sub System Numbers (SSN) used by these nodes are standard SSN as per the GSM specifications.

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

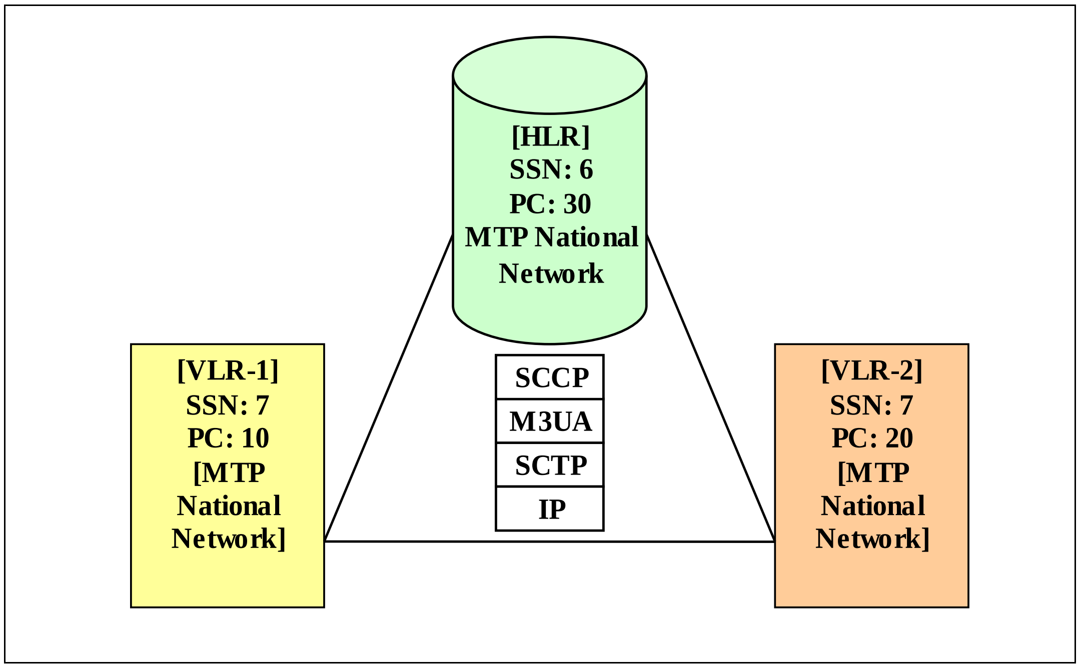


Figure 3: Example of SSN based Routing

8.1.1 Configuring SCCP at HLR

SCCP APIs need to be invoked in a specific sequence to initialize SCCP layer, configure various network resources [point codes, subsystems etc.] and then bringing SCCP subsystems into service. Following is the sequence of APIs that need to be invoked to configure SCCP at HLR node.

Note:- LM is Layer Management, LSS is Local Sub System

1. **SCCP_INIT (LM→SCCP)** – Initialize the SCCP protocol layer. [Please note that all the SCCP resources are configurable and proper scaling should be done before preparing SCCP protocol library.]
2. **SCCP_MGMT_STATUS (LM→SCCP)** – This API is used to put SCCP layer into ALLOWED state. This means that SCCP layer is now ready to respond to SST [SSN=1] with SSA [SSN=1] message.
3. **SCCP_ADD_MTP_SAP (LM→SCCP)** – This API would be invoked one or more times to configure service access points provided by the underlying MTP/SIGTRAN layer. This is primarily used for configuring network standard, maximum network PDU size allowed, network indicator value etc. In this example only one MTP-SAP is required as HLR is communicating only within the National MTP network.
4. **SCCP_ADD_PC_RESOURCES (LM→SCCP)** – This API would be invoked one or more times to configure all the concerned Point Code resources for the SCCP Layer. This means that any point code that would be communicating with the SCCP layer needs to be configured using this API. At HLR, two PC resources one for each VLR would be configured. So, PC resources 10 and 20 would be configured on the HLR.
5. **SCCP_ADD_RMT_RESOURCES (LM→SCCP)** – This API would be invoked one or more times to configure all the remote subsystems that would be communicating with the local subsystems or local SCCP layer. At HLR, there would be two remote subsystems configured. Each of them would be having SSN as 7 and point code 10 and 20 respectively.
6. **SCCP_ADD_SAP (LM→SCCP)** – This API would be invoked one or more times to configure all the local subsystems using services from the SCCP layer. Only one local SCCP SAP is required with SSN as 6 for providing SCCP connection-less service interface to the HLR application.
7. **SCCP_N_STATE_REQ (LM→SCCP)** – This API would be invoked one or more times to put a local SCCP sub system in to either ALLOWED or PROHIBITED state. On HLR, this API would be used to put local SSN 6 into service or taking it out of service.
8. **SCCP_USER_NTFY (SCCP → LSS**) – This API is used by the SCCP Layer to provide notifications to the local subsystems. UNITDATA received for a particular local subsystem is also passed to it using this API.
9. **SCCP_N_UDT_REQ (LSS→SCCP)** – This API is used for transferring connection-less message to a remote or a local subsystem.

8.1.2 Configuring SCCP at VLR-1 or VLR-2

Following is the sequence of APIs that need to be invoked to configure SCCP layer at either VLR-1 or VLR-2 network node.

Note:- LM is Layer Management, LSS is Local Sub System

1. **SCCP_INIT (LM→SCCP)** – Initialize the SCCP protocol layer. [Please note that all the SCCP resources are configurable and proper scaling should be done before preparing SCCP protocol library.]
2. **SCCP_MGMT_STATUS (LM→SCCP)** – This API is used to put SCCP layer into ALLOWED state. This means that SCCP layer is now ready to respond to SST [SSN=1] with SSA [SSN=1] message.
3. **SCCP_ADD_MTP_SAP (LM→SCCP)** – This API would be invoked one or more times to configure service access points provided by the underlying MTP/SIGTRAN layer. This is primarily used for configuring network standard, maximum network PDU size allowed, network indicator value etc. In this example only one MTP-SAP is required as VLR-1/VLR-2 is communicating only within the National MTP network.
4. **SCCP_ADD_PC_RESOURCES (LM→SCCP)** – This API would be invoked one or more times to configure all the concerned Point Code resources for the SCCP Layer. This means that any point code that would be communicating with the SCCP layer needs to be configured using this API. At VLR-1/VLR-2, two PC resources would be configured. On VLR-1 PC resources for VLR-2 [20] and HLR [30] would be configured. On VLR-2 PC resources for VLR-1 [10] and HLR [30] would be configured.
5. **SCCP_ADD_RMT_RESOURCES (LM→SCCP)** – This API would be invoked one or more times to configure all the remote subsystems that would be communicating with the local subsystems or local SCCP layer. At VLR-1 remote resources with SSN 7 over VLR-2 [PC:20] and SSN 6 over HLR [PC:30] are configured. On VLR-2 remote resources with SSN 7 over VLR-1 [PC:10] and SSN 6 over HLR [PC:30] are configured.
6. **SCCP_ADD_SAP (LM→SCCP)** – This API would be invoked one or more times to configure all the local subsystems using services from the SCCP layer. Only one local SCCP SAP is required with SSN as 7 for providing SCCP connection-less service interface to the HLR application.
7. **SCCP_N_STATE_REQ (LM→SCCP)** – This API would be invoked one or more times to put a local SCCP sub system in to either ALLOWED or PROHIBITED state. On VLR-1/VLR-2, this API would be used to put local SSN 7 into service or taking it out of service.
8. **SCCP_USER_NTFY (SCCP → LSS**) – This API is used by the SCCP Layer to provide notifications to the local subsystems. UNITDATA received for a particular local subsystem is also passed to it using this API.
9. **SCCP_N_UDT_REQ (LSS→SCCP)** – This API is used for transferring connection-less message to a remote or a local subsystem.

8.2 GTT Based Communication Scenario with Multiple MTP Networks

Figure 4 shows the case where Global Title based routing needs to be used to route connection-less messages between two different national networks. The messages would originate from a national MTP network, go through international MTP network and then enter the destination national MTP network. This is a typical case where two mobile operators of different countries have a roaming agreement.

In the following sub-sections, we will explain configuration sequence from SCCP point of view for VLR-1, HLR-1 and STP-1 network nodes as shown in Figure 4. The configuration at VLR-2, HLR-2 and STP-2 would be similar in nature.

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

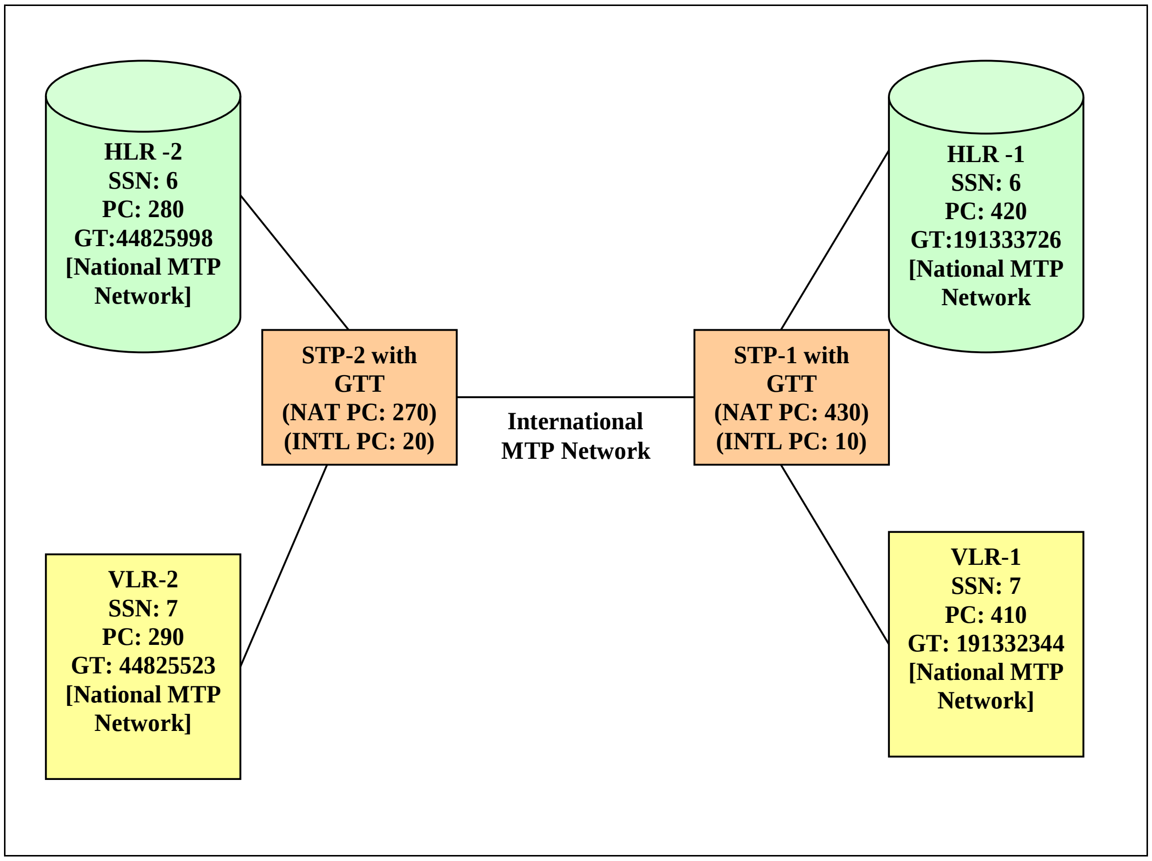


Figure 4: GT Based Routing with Multiple MTP Networks

8.2.1 Configuring SCCP at HLR-1

As we see in Figure 4 HLR-1 is connected to all other nodes through STP-1. We will assume that GTT based routing is being used by HLR-1 to communicate with other network nodes. Following is the sequence of APIs that need to be invoked to configure SCCP at HLR-1 node.

Note:- LM is Layer Management, LSS is Local Sub System

1. **SCCP_INIT (LM→SCCP)** – Initialize the SCCP protocol layer. [Please note that all the SCCP resources are configurable and proper scaling should be done before preparing SCCP protocol library.]
2. **SCCP_MGMT_STATUS (LM→SCCP)** – This API is used to put SCCP layer into ALLOWED state. This means that SCCP layer is now ready to respond to SST [SSN=1] with SSA [SSN=1] message.
3. **SCCP_ADD_MTP_SAP (LM→SCCP)** – This API would be invoked one or more times to configure service access points provided by the underlying MTP/SIGTRAN layer. This is primarily used for configuring network standard, maximum network PDU size allowed, network indicator value etc. In this example only one MTP-SAP is required as HLR is communicating only within the National MTP network.
4. **SCCP_ADD_PC_RESOURCES (LM→SCCP)** – This API would be invoked one or more times to configure all the concerned Point Code resources for the SCCP Layer. This means that any point code that would be communicating with the SCCP layer needs to be configured using this API. At HLR, one PC resource for STP-1 [PC: 430] would be configured.
5. **SCCP_ADD_GENERIC_GTT_RULESET** – As communication is based on GTT based routing and actual GTT routing takes place on STP-1, we will configure a generic GTT rule-set to forward all "Route-on-GT" based messages to STP-1.
6. **SCCP_ADD_SAP (LM→SCCP)** – This API would be invoked one or more times to configure all the local subsystems using services from the SCCP layer. Only one local SCCP SAP is required with SSN as 6 for providing SCCP connection-less service interface to the HLR application.
7. **SCCP_N_STATE_REQ (LM→SCCP)** – This API would be invoked one or more times to put a local SCCP sub system in to either ALLOWED or PROHIBITED state. On HLR, this API would be used to put local SSN 6 into service or taking it out of service.
8. **SCCP_USER_NTFY (SCCP → LSS**) – This API is used by the SCCP Layer to provide notifications to the local subsystems. UNITDATA received for a particular local subsystem is also passed to it using this API.
9. **SCCP_N_UDT_REQ (LSS→SCCP)** – This API is used for transferring connection-less message to a remote or a local subsystem.

8.2.2 Configuring SCCP at STP-1

Following is the sequence of APIs that need to be invoked to configure SCCP layer at STP-1 network node.

Note:- LM is Layer Management, LSS is Local Sub System

1. **SCCP_INIT (LM→SCCP)** – Initialize the SCCP protocol layer. [Please note that all the SCCP resources are configurable and proper scaling should be done before preparing SCCP protocol library.]
2. **SCCP_MGMT_STATUS (LM→SCCP)** – This API is used to put SCCP layer into ALLOWED state. This means that SCCP layer is now ready to respond to SST [SSN=1] with SSA [SSN=1] message.
3. **SCCP_ADD_MTP_SAP (LM→SCCP)** – This API would be invoked one or more times to configure service access points provided by the underlying MTP/SIGTRAN layer. This is primarily used for configuring network standard, maximum network PDU size allowed, network indicator value etc. In this example two MTP-SAP are required, one for national MTP network and other for international MTP network.
4. **SCCP_ADD_PC_RESOURCES (LM→SCCP)** – This API would be invoked one or more times to configure all the concerned Point Code resources for the SCCP Layer. This means that any point code that would be communicating with the SCCP layer needs to be configured using this API. At STP-1, three PC resources would be configured for HLR-1, STP-2 and VLR-1.
5. **Configuring GTT Rule-Sets** – Following set of APIs are invoked one or more times to configure GTT for VLR-1, HLR-1 and STP-2.
    a. **SCCP_BEGIN_GTT_RULESET_ADDITION**
    b. **SCCP_ADD_RULES** [Invoked 1 or more times for adding comparison rules]
    c. **SCCP_ADD_RULES** [Invoked 1 or more times for adding modification rules]
    d. **SCCP_END_GTT_RULESET_ADDITION**
6. **SCCP_USER_NTFY (SCCP → LSS**) – This API is used by the SCCP Layer to provide notifications to the local subsystems. UNITDATA received for a particular local subsystem is also passed to it using this API.
7. **SCCP_N_UDT_REQ (LSS→SCCP)** – This API is used for transferring connection-less message to a remote or a local subsystem.

8.2.3 Configuring SCCP at VLR-1

As we see in Figure 4 VLR-1 is connected to all other nodes through STP-1. The configuration sequence of VLR-1 would be similar to HLR-1 with appropriate SSN values and point codes.

9 Timing Management for SCCP Library

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

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

Supplying a timer tick to SCCP library is similar to passing an external event like a message or invoking any API. All the SCCP-MG, management notifications and timer events must be passed to the SCCP library using a single thread. However, user data may be passed to SCCP library using multi-threaded design.

10 Collecting Logs from SCCP Library

SCCP library logs messages using following two C macros:

1. `SP_TRACE` – For logging messages
2. `SP_HEX` – 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 "`spdef.h`". Presently, these macros are using standard C IO function (`printf`) for logging messages directly to the screen.

11 API between SCCP and MTP/M3UA

This section describes API interface between SCCP and M3UA layers. Complete details of these APIs may be found in document later.

11.1 Handling Notifications from M3UA

It is assumed that before SCCP is configured, M3UA ASP/SGP/IPSP entities are already provisioned and ready to use. A change in M3UA entities [ASP/IPSP/Remote-AS] state may lead to change in overall availability state of a point code. This availability state of point code needs to be informed to the SCCP layer using standard notifications as following:

- SCCP_MTP_PAUSE_IND

- SCCP_MTP_RESUME_IND
- SCCP_MTP_STATUS_IND

11.2 Sending Data to M3UA/MTP

SCCP layer would send any messages to M3UA layer using the API SCCP_SENDMSG. It is responsibility of SCCP application developer to write this API using services provided by the underlying MTP/SIGTRAN layer [M3UA in this case].

11.3 Receiving Data from M3UA/MTP

Whenever any data is received by M3UA for SCCP layer, it needs to be passed to the SCCP layer. This can be done through SCCP_MTP_TRANSFER_IND API. This API takes routing label, MTP-SAP-ID and message contents as input.

12 API [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 SCCP API's. **We strongly recommend you to have a look in the sample application source code that has been supplied along with the Sure Speed SCCP package. This would give you in-depth idea about usage of SCCP API's.**

```
#include <spdef.h>
#include <sptyp.h>
#include <sperr.h>
#include <spprt.h>
#include <spapi.h>
```

Above *include files* contain structures, enumerations, compile time defined values, error numbers and various other data structures that would help in writing application based on Sure Speed SCCP library.

This section of the document describes API's in detail, and services provided by each of the API implemented as part of Sure Speed SCCP implementation. Please see section "***Data Structures Associated with API Primitives"*** for source code of structures used along with API primitives.

13 Scaling SCCP 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
SCCP_MAX_TIMERS	spdef.h	0x0200	Maximum no. of timers supported within SCCP layer
SCCP_MAX_USR_DAT_SZ	spdef.h	1024	Maximum user data size accepted by SCCP layer
SP_MAX_PENDING_XUDT	spdef.h	0x0100	Maximum no. of XUDT messages that may remain pending for reassembly as last segment is not yet received
SP_MAX_GT_DIGITS	spdef.h	0x20	Maximum no. of GT digits, generally this is not more than 15
SP_MAX_RESRC_PER_PC	spdef.h	0x10	Maximum no. of local or remote resource per PC resource
SP_MAX_CNRD_PC	spdef.h	0x10	Maximum no. of concerned point codes for a local or remote subsystem
SP_MAX_SCCP_SAP	spdef.h	0x0100	Maximum no. of Service Access Points supported by the SCCP layer
SP_MAX_CNRD_RMT_RESRC	spdef.h	0x10	Maximum no. of concerned remote resources for a local subsystem
SP_MAX_PC_RESRC	spdef.h	0x0100	Maximum no. of PC resources supported by the SCCP layer
SP_MAX_RMT_RESRC	spdef.h	0x0200	Maximum no. of remote resources supported by the SCCP layer
SP_MAX_MTP_SAP	spdef.h	0x0010	Maximum no. of MTP SAP supported by the SCCP layer
SP_MAX_MGMT_NTFY	spdef.h	64	Maximum no. of MGMT related notifications that may remain pending within the SCCP layer
SP_MAX_USR_NTFY	spdef.h	64	Maximum no. of USER notifications that may remain pending within the

			SCCP layer
SP_MAX_GTT_RULES	sptyp.h	1024	Maximum no. of GTT rules allowed
SP_MAX_GTT_RULESETS	sptyp.h	256	Maximum no. of GTT rule-sets allowed
SP_MAX_GTT_PC	sptyp.h	0x04	Maximum no. of output point codes after GTT is performed

For example; if an SCCP application requires connecting with only two remote subsystems, then the value of SP_MAX_RMT_RESRC may be reduced to 2. Similarly any other value above may be fine tuned as per the requirements. This would accordingly increase or decrease the overall memory requirement for SCCP layer.

14 Layer Management API (LM→SCCP)

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

14.1 SCCP_INIT

Description	This API is used by layer management to initialize the SCCP layer. The SCCP 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 “spdef.h” and “sptyp.h”.	
Prototype	sccp_s32 sccp_init (void);	
Parameters	Associated C Data Structures	None
Restrictions/Bugs	None	
Pre-requisites	None	
Code Sample	Please refer source file demo/sccp.c	

14.2 SCCP_MGMT_STATUS

Description	This API is used to change the status of SCCP management subsystem [SSN=1]
Prototype	sccp_s32 sccp_MGMT_status

	(sccp_u32 status);	
	Associated C Data Structures	None
	status	This parameter is of type “unsigned 32 bits”. It is used to set the overall status of the SCCP layer or management Subsystem [SSN=1]. The values that this parameter can take are: • SP_SS_PRHBTD – For marking subsystem status as prohibited • SP_SS_ALLWD – For marking subsystem status as allowed
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT	
Code Sample	Please refer source file demo/sccp.c	

14.3 SCCP_CONFIG

Description	This API is used to set values for various SCCP wide parameters as per the choice of application developer.	
Prototype	sccp_s32 sccp_config (sccp_gbl_cfg_t *gbl_cfg_p);	

	Associated C Data Structures	`typedef struct`
		`{`
		`sccp_u32 N_Rst_Lvl;`
		`sccp_u32 N_Rst_Sub_Lvl;`
		`sccp_u8 hopCtr;`
		`sccp_u32 opt;`
		`} sccp_gbl_cfg_t;`

	N_Rst_Lvl	Number of restriction levels [RL_M], that would be used for managing signaling point congestion status
	N_Rst_Sub_Lvl	Number of restriction sub levels [RSL_N] present in a restriction level that would be used for managing signaling point congestion status
	hopCtr	Hop Counter value to be used while sending an outgoing XUDT or LUDT messages. This value can be overridden by the application developer by supplying a valid hop counter value through SCCP API SCCP_N_UDT_REQ.
	opt	This parameter specifies the SCCP wide options that needs to enable or disabled. Presently only one option is allowed as following: SP_OPT_SEND_SSA_RESTART – Set this option

		when SCCP is required to send SSA messages to all the concerned remote point codes
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT	
Code Sample	Please refer source file demo/sccp.c	

14.4 SCCP_RECONFIG

Description	This API is used to set values for various SCCP wide parameters individually as per the choice of application developer.	
Prototype	bool sccp_reconfig (sccp_gbl_cfg_name name, sccp_gbl_cfg_t *gbl_cfg_p);	

Parameters	Associated C Data Structures	`typedef struct`
		`{`
		`sccp_u32 N_Rst_Lvl;`
		`sccp_u32 N_Rst_Sub_Lvl;`
		`sccp_u8 hopCtr;`
		`sccp_u32 opt;`
		`} sccp_gbl_cfg_t;`
		
		`typedef enum`
		`{`
		`SCCP_GBLCFG_RST_LVL,`
		`SCCP_GBLCFG_RST_SUBLVL,`
		`SCCP_GBLCFG_HOP_CTR`
		`} sccp_gbl_cfg_name;`

	name	“name” is an enumerated parameter of type “sccp_gbl_cfg_name”. This parameter specifies the name of the individual configuration parameter whose value is to be modified. The “name” parameter can take following values: • SCCP_GBLCFG_RST_LVL – This value specifies that restriction level parameter is to be reconfigured as per the value specified in N_Rst_Lvl variable. • SCCP_GBLCFG_RST_SUBLVL – This value specifies that restriction sub level parameter is to be reconfigured as per the value specified in the N_Rst_Sub_Lvl variable • SCCP_GBLCFG_HOP_CTR – This value specifies that default hop counter value is to be reconfigured as per the value specified in the hopCtr variable.
Restrictions/Bugs	None	

Pre-requisites	SCCP_INIT
Code Sample	Please refer source file demo/sccp.c

14.5 SCCP_SET_OPT

Description	This API is used to set the SCCP wide options. This is the same option that may also be configured using the SCCP_CONFIG API. It is recommended to use this API only when SCCP wide options are to be modified.	
Prototype	sccp_s32 sccp_set_opt (sccp_u32 opt);	
Parameters	Associated C Data Structures	none
	opt	This parameter specifies the SCCP wide options that needs to enable or disabled. Presently only one option is allowed as following: SP_OPT_SEND_SSA_RESTART – Set this option when SCCP is required to send SSA messages to all the concerned remote point codes
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT	
Code Sample	Please refer source file demo/sccp.c	

14.6 SCCP_GET_OPT

Description	This API is used to get the SCCP wide options.	
Prototype	sccp_s32 sccp_get_opt (sccp_u32 *opt);	
Parameters	Associated C Data Structures	none
	opt	This parameter is a pointer to a unsigned 32 bit parameter. The SCCP layer sets this parameter to the value of SCCP wide options.
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT	
Code Sample	Please refer source file demo/sccp.c	

14.7 SCCP_TIMER_CONFIG

Description	This API is used to set values for various SCCP specific timers. The values are in milliseconds.	
Prototype	bool sccp_timer_config (sccp_timer_cfg_t *cfg_p);	

Parameters	Associated C Data Structures	`typedef struct`
		`{`
		`sccp_u32 Tattack;`
		`sccp_u32 Tdecay;`
		`sccp_u32 Tstat_info;`
		`sccp_u32 Tstat_info_step;`
		`sccp_u32 Tstat_info_max;`
		`sccp_u32 Txudt_ras;`
		`sccp_u32 Tcoord_chg;`
		`sccp_u32 Tignore_sst;`
		`} sccp_timer_cfg_t;`

	Tattack	SCCP Attack timer duration in milliseconds. Default value of this timer is 400 ms.
	Tdecay	SCCP Decay timer duration in milliseconds. Default value of this timer is 5000 ms (5 seconds).
	Tstat_info	SCCP Timer(stat.info) duration in milliseconds. Default value of this timer is 30000 ms (30 seconds).
	Tstat_info_step	This timer specifies the step size in milliseconds that would be added to SCCP timer(stat.info) before every SST retransmission is done. Default value of step size is 30000 ms (30 seconds).
	Tstat_info_max	This timer specifies the maximum duration between two SST transmissions in milliseconds. Default value of this timer is 600000 ms (10 minutes).
	Txudt_ras	This timer specifies the duration for which SCCP would wait for all segments of a particular message to arrive. In case all the segments do not arrive within this duration, then all the segments are discarded. Default value for this timer is 15000 ms (15 seconds).
	Tcoord_chg	SCCP Timer(coord.chg) duration in milliseconds. Default value of this timer is 90000 milliseconds (90 seconds).
	Tignore_sst	SCCP Timer(ignore SST) duration in milliseconds. Default value of this timer is 120000 milliseconds (2 minutes).
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT	
Code Sample	Please refer source file demo/sccp.c	

14.8 SCCP_TIMER_RECONFIG

Description	This API is used to set value of a specific SCCP timer. The value is in milliseconds.	
Prototype	bool sccp_timer_reconfig (sccp_timer_name name, sccp_timer_cfg_t *cfg_p);	

Parameters	Associated C Data Structures	`typedef enum`
		`{`
		`SCCP_TIMER_STAT_INFO = 0,`
		`SCCP_TIMER_ATTACK = 1,`
		`SCCP_TIMER_DECAY = 2,`
		`SCCP_TIMER_XUDT_RAS = 3,`
		`SCCP_TIMER_STAT_INFO_STEP = 4,`
		`SCCP_TIMER_STAT_INFO_MAX = 5,`
		`SCCP_TIMER_COORD_CHG = 6,`
		`SCCP_TIMER_IGNORE_SST = 7,`
		`SCCP_TIMER_UNDEF = 8`
		`} sccp_timer_name;`
		
		`typedef struct`
		`{`
		`sccp_u32 Tattack;`
		`sccp_u32 Tdecay;`
		`sccp_u32 Tstat_info;`
		`sccp_u32 Tstat_info_step;`
		`sccp_u32 Tstat_info_max;`
		`sccp_u32 Txudt_ras;`
		`sccp_u32 Tcoord_chg;`
		`sccp_u32 Tignore_sst;`
		`} sccp_timer_cfg_t;`

	name	This parameter specifies the name of the timer whose value is to be configured in the SCCP layer.
	Tattack	SCCP Attack timer duration in milliseconds. To set this timer value, “name” parameter must be set to “`SCCP_TIMER_ATTACK`”.
	Tdecay	SCCP Decay timer duration in milliseconds. To set this timer value, “name” parameter must be set to “`SCCP_TIMER_DECAY`”.
	Tstat_info	SCCP Timer(stat.info) duration in milliseconds. To set this timer value, “name” parameter must be set to “`SCCP_TIMER_STAT_INFO`”.
	Tstat_info_step	This timer specifies the step size in milliseconds that would be added to SCCP timer(stat.info) before every SST retransmission is done. To set this timer value, “name” parameter must be set to “`SCCP_TIMER_STAT_INFO_STEP`”.
	Tstat_info_max	This timer specifies the maximum duration between two SST transmissions in milliseconds. To set this timer value, “name” parameter must be set to

		“SCCP_TIMER_STAT_MAX”.
	Txudt_ras	This timer specifies the duration for which SCCP would wait for all segments of a particular message to arrive. In case all the segments do not arrive within this duration, then all the segments are discarded. To set this timer value, “name” parameter must be set to “SCCP_TIMER_XUDT_RAS”.
	Tcoord_chg	SCCP Timer(coord.chg) duration in milliseconds. To set this timer value, “name” parameter must be set to “SCCP_TIMER_COORD_CHG”.
	Tignore_sst	SCCP Timer(ignore SST) duration in milliseconds. To set this timer value, “name” parameter must be set to “SCCP_TIMER_IGNORE_SST”.
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT	
Code Sample	Please refer source file demo/sccp.c	

14.9 SCCP_ADD_MTP_SAP

Description	This API is used for configuring an MTP Service Access Point. All the communication between SCCP and MTP/M3UA takes place through a MTP-SAP. There may be more than one MTP-SAP providing service to the SCCP layer. All these MTP-SAP may be configured one-by-one using this API.	

Prototype	`sccp_s32 sccp_Add_MTP_SAP`
	`      (`
	`            sccp_u16 mtp_sap_id,`
	`            sccp_mtpsap_cfg_t *cfg_p`
	`      );`	

Parameters	Associated C Data Structures	`typedef struct`
		`{`
		`      sccp_u32 pc;`
		`      sccp_u32 std;`
		`      sccp_u32 maxDataSz;`
		`      sccp_u32 opt;`
		`      sccp_u8 ni;`
		`      sccp_u8 natResvBit;`
		`      sccp_u32 ss7NtwID;`
		`} sccp_mtpsap_cfg_t;`

	mtp_sap_id	This parameter is an unsigned 16 bit variable se t to identifier of MTP-SAP. This parameter may take values from 0 to less than SP_MAX_MTP_SAP.
	pc	Originating Point Code associated with the MTP-SAP.
	std	MTP standard associated with the MTP-SAP. The values for this parameter may be one of the following:

		• `SCCP_STD_ITU` • `SCCP_STD_ANSI`
	`maxDataSz`	Maximum size of data that may be passed to the MTP/M3UA by the SCCP layer. This value must be set according to the maximum data size supported by the underlying transport layer. This size includes size of MTP3 routing label also.
	`opt`	The options associated with the MTP-SAP. Presently only one option is associated with the MTP-SAP: • `SCCP_MSOPT_LUDT_ALLWD` – If this option is set, then LUDT is allowed by the MTP-SAP.
	`ni`	Network Indicator value associated with the MTP-SAP.
	`natResvBit`	National Reserve Bit to be used when sending any SCCP management messages through this MTP-SAP.
	`ss7NtwID`	This is a logical identifier associated with the MTP network. SCCP layer does not use this parameter internally. [Example: this parameter may be set to the M3UA network appearance parameter.]
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT	
Code Sample	Please refer source file demo/sccp.c	

14.10 SCCP_DEL_MTP_SAP

Description	This API is used to delete an already configured MTP-SAP. Please note that before deleting an MTP-SAP, associated point code resources, remote-resources and local SCCP SAP using the MTP-SAP must be deleted.	

Prototype	`sccp_s32 sccp_Del_MtpSap`
	`      (`
	`            sccp_u16 mtp_sap_id`
	`      );`	

Parameters	Associated C Data Structures	
	`mtp_sap_id`	This parameter is an unsigned 16 bit variable se t to identifier of MTP-SAP. This parameter may take values from 0 to less than `SP_MAX_MTP_SAP`. Please note that this MTP-SAP must have been configured earlier using API SCCP_ADD_MTP_SAP.
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT, SCCP_ADD_MTP_SAP	

Code Sample	Please refer source file demo/sccp.c

14.11 SCCP_GET_MTP_SAP_CFG

Description	This API is used to get configuration of an already configured MTP Service Access Point.	

Prototype	`sccp_s32 sccp_Get_MtpSap_Cfg`
	`(`
	`sccp_u16 mtp_sap_id,`
	`sccp_mtpsap_cfg_t *cfg_p`
	`);`	

Parameters	Associated C Data Structures	`typedef struct`
		`{`
		`sccp_u32 pc;`
		`sccp_u32 std;`
		`sccp_u32 maxDataSz;`
		`sccp_u32 opt;`
		`sccp_u8 ni;`
		`sccp_u32 ss7NtwID;`
		`} sccp_mtpsap_cfg_t;`

	`mtp_sap_id`	Identifier of an already configured MTP-SAP
	`cfg_p`	This is pointer to the structure in which SCCP layer returns configuration of the MTP-SAP. Please refer API SCCP_ADD_MTP_SAP for more details on each parameter of this structure.
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT SCCP_ADD_MTP_SAP	
Code Sample	Please refer source file demo/sccp.c	

14.12 SCCP_ADD_PC_RESRC

Description	This API is used for configuring a remote point code resource.	

Prototype	`sccp_s32 sccp_Add_PcResrc`
	`(`
	`sccp_u16 pc_resrc_id,`
	`sccp_pc_resrc_cfg_t *cfg_p`
	`);`	

Parameters	Associated C Data Structures	`typedef struct`
		`{`
		`sccp_u32 pc;`
		`sccp_u32 mtp_sap_id;`
		`sccp_u32 opt;`
		`sccp_u32 status;`
		`} sccp_pc_resrc_cfg_t;`

	`pc_resrc_id`	This parameter is an unsigned 16 bit variable set to identifier of remote point code resource. This

		parameter may take values from 0 to less than SP_MAX_PC_RESRC.
	pc	Point Code of the remote point code resource
	mtp_sap_id	Identifier of MTP-SAP through which point code may be accessed
	opt	Options associated with the remote point code resource. Presently there are no options associated with a remote point code resource. This parameter must be set to 0.
	status	Initial status of the point code resource. The values that this parameter may take are: • SP_PC_PRHBTD – Point code resource is prohibited. This is recommended initial value. • SP_PC_SCCP_PRHBTD – SCCP at point code resource is prohibited • SP_PC_CNGSTD – Point code is congested • SP_PC_ALLWD – Point code resource is allowed
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT SCCP_ADD_MTP_SAP	
Code Sample	Please refer source file demo/sccp.c	

14.13 SCCP_DEL_PC_RESRC

Description	This API is used to delete an already configured remote point code resource. Please note that before deleting a remote PC resource, associated remote-subsystem-resources residing on the point code resource must be deleted.	
Prototype	sccp_s32 sccp_Del_PcResrc (sccp_u16 pc_resrc_id);	
Parameters	Associated C Data Structures	
	pc_resrc_id	This parameter is an unsigned 16 bit variable set to identifier of remote PC resource. This parameter may take values from 0 to less than SP_MAX_PC_RESRC. Please note that this PC-resource must have been configured earlier using API SCCP_ADD_PC_RESRC.
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT, SCCP_ADD_MTP_SAP, SCCP_ADD_PC_RESRC	
Code Sample	Please refer source file demo/sccp.c	

14.14 SCCP_GET_PC_RESRC_CFG

Description	This API is used to get configuration of an already configured remote point code resource.	

Prototype	`sccp_s32 sccp_Get_PcResrc_Cfg`
	`(`
	`sccp_u16 pc_resrc_id,`
	`sccp_pc_resrc_cfg_t *cfg_p`
	`);`	

Parameters	Associated C Data Structures	`typedef struct`
		`{`
		`sccp_u32 pc;`
		`sccp_u32 mtp_sap_id;`
		`sccp_u32 opt;`
		`sccp_u32 status;`
		`} sccp_pc_resrc_cfg_t;`

	`pc_resrc_id`	Identifier of an already configured remote PC resource
	`cfg_p`	This is pointer to the structure in which SCCP layer returns configuration of the PC-resource. Please refer API SCCP_ADD_PC_RESRC for more details on each parameter of this structure.
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT SCCP_ADD_MTP_SAP SCCP_ADD_PC_RESRC	
Code Sample	Please refer source file demo/sccp.c	

14.15 SCCP_GET_PC_RESRC_STATUS

Description	This API is used to get present status of an already configured remote point code resource.	

Prototype	`sccp_s32 sccp_Get_PcResrc_Status`
	`(`
	`sccp_u16 pc_resrc_id,`
	`sccp_u32 *status_p`
	`);`	

Parameters	Associated C Data Structures	
	`pc_resrc_id`	Identifier of an already configured remote point code resource
	`status_p`	This is pointer to the unsigned 32 bits variable in which SCCP layer would return present status of the remote point code. Please refer API SCCP_ADD_PC_RESRC for valid values of status of a point code resource.
Restrictions/Bugs	None	

Pre-requisites	SCCP_INIT, SCCP_ADD_MTP_SAP, SCCP_ADD_PC_RESRC
Code Sample	Please refer source file demo/sccp.c

14.16 SCCP_ADD_RMT_RESRC

Description	This API is used for configuring a remote subsystem.	

Prototype	`sccp_s32 sccp_Add_RmtResrc`
	`(`
	`sccp_u16 rmt_resrc_id,`
	`sccp_rmt_resrc_cfg_t *cfg_p`
	`);`	

Parameters	Associated C Data Structures	`typedef struct`
		`{`
		`sccp_u32 ssn;`
		`sccp_u32 pc;`
		`sccp_u32 opt;`
		`sccp_u32 mtp_sap_id;`
		`sccp_u8 natResvBit;`
		`sccp_u16 nCrndPc;`
		`sccp_u16`
		`crndPcList[SP_MAX_CNRD_PC];`
		`} sccp_rmt_resrc_cfg_t;`

	`rmt_resrc_id`	This parameter is an unsigned 16 bit variable se t to identifier of remote resource. This parameter may take values from 0 to less than SP_MAX_RMT_RESRC.
	`ssn`	Subsystem number of the remote subsystem
	`pc`	Point Code at which remote subsystem resides
	`opt`	Options associated with this remote subsystem. Presently only one option is supported as following: • SP_RROPT_SND_SST_PC_AVBL – SCCP layer would send SST for this remote subsystem as soon as point code on which this subsystem resides becomes available.
	`mtp_sap_id`	Identifier of MTP-SAP through which point code on which remote subsystem resides may be accessed
	`natResvBit`	National Reserved Bit to be set in SCCP called party address when sending a message to this remote subsystem. This parameter can take a value of 1 or 0.
	`nCrndPc`	Number of PC resources concerned with the status of this remote subsystem
	`crndPcList`	List of point codes concerned with the status of this remote subsystem. These point codes would be informed of any status change of the remote

		subsystem.
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT SCCP_ADD_MTP_SAP SCCP_ADD_PC_RESRC	
Code Sample	Please refer source file demo/sccp.c	

14.17 SCCP_DEL_RMT_RESRC

Description	This API is used to delete an already configured remote-subsystem.	

Prototype	`sccp_s32 sccp_Del_RmtResrc`
	`      (`
	`            sccp_u32 rmt_resrc_id`
	`      );`	

Parameters	Associated C Data Structures	
	`rmt_resrc_id`	This parameter is an unsigned 16 bit variable set to identifier of remote-subsystem-resource. This parameter may take values from 0 to less than SP_MAX_RMT_RESRC. Please note that this remote-subsystem must have been configured earlier using API SCCP_ADD_RMT_RESRC.
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT, SCCP_ADD_MTP_SAP, SCCP_ADD_PC_RESRC, SCCP_ADD_RMT_RESRC	
Code Sample	Please refer source file demo/sccp.c	

14.18 SCCP_GET_RMT_RESRC_CFG

Description	This API is used to get configuration of an already configured remote subsystem resource.	

Prototype	`sccp_s32 sccp_Get_RmtResrc_Cfg`
	`      (`
	`            sccp_u32 rmt_resrc_id,`
	`            sccp_rmt_resrc_cfg_t *cfg_p`
	`      );`	

Parameters	Associated C Data Structures	`typedef struct`
		`{`
		`      sccp_u32 ssn;`
		`      sccp_u32 pc;`
		`      sccp_u32 opt;`
		`      sccp_u32 mtp_sap_id;`
		`      sccp_u8 natResvBit;`
		`      sccp_u16 nCrndPc;`
		`      sccp_u16`
		`crndPcList[SP_MAX_CNRD_PC];`

		`} sccp_rmt_resrc_cfg_t;`
	`rmt_resrc_id`	Identifier of an already configured remote-resource
	`cfg_p`	This is pointer to the structure in which SCCP layer returns configuration of the remote-subsystem. Please refer API SCCP_ADD_RMT_RESRC for more details on each parameter of this structure.
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT SCCP_ADD_MTP_SAP SCCP_ADD_PC_RESRC SCCP_ADD_RMT_RESRC	
Code Sample	Please refer source file demo/sccp.c	

14.19 SCCP_GET_RMT_RESRC_STATUS

Description	This API is used to get status of an already configured remote subsystem resource.	

Prototype	`sccp_s32 sccp_Get_RmtResrc_Status`
	`       (`
	`              sccp_u32 rmt_resrc_id,`
	`              sccp_u32 *status_p`
	`       );`	

Parameters	Associated C Data Structures	
	`rmt_resrc_id`	Identifier of an already configured remote-resource
	`status_p`	This is pointer to the unsigned 32 bits variable in which SCCP layer would return present status of the remote-subsystem. The value of remote-subsystem may be one of the following: • `SP_SS_PRHBTD` • `SP_SS_ALLWD`
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT, SCCP_ADD_MTP_SAP, SCCP_ADD_PC_RESRC, SCCP_ADD_RMT_RESRC	
Code Sample	Please refer source file demo/sccp.c	

14.20 SCCP_ADD_SAP

Description	This API is used for configuring a service access point through which a local subsystem would be taking services from the SCCP layer.

Prototype	`sccp_s32 sccp_Add_SAP`
	`       (`
	`              sccp_u16 sccp_sap_id,`
	`              sccp_sap_cfg_t *param_p`
	`       );`

Parameters	Associated C Data Structures	`typedef struct`
		`{`
		`        sccp_u8 ssn;`
		`        sccp_u32 mtp_sap_id;`
		`        sccp_u32 app_id;`
		`        sccp_u32 opt;`
		`        sccp_u8 natResvBit;`
		`        sccp_u16 nCrndRmtResrc;`
		`        sccp_u16`
		`crndRmtResrc[SP_MAX_CNRD_RMT_RESRC];`
		`} sccp_sap_cfg_t;`

	`sccp_sap_id`	This parameter is an unsigned 16 bit variable set to identifier of SCCP service access point. This parameter may take values from 0 to less than `SP_MAX_SCCP_SAP`.
	`ssn`	SSN of the local subsystem that would take services from SCCP through this SAP.
	`mtp_sap_id`	Identifier of MTP-SAP through which messages originated from this local subsystem would be delivered to MTP/M3UA.
	`app_id`	Application identifier of the local subsystem. This number is not used by SCCP layer internally. But it would be provided back to the local subsystem in the notifications from the SCCP layer.
	`opt`	Options associated with this local subsystem. Presently only one option is supported as following: • `SP_SOPT_MAP_USER` – This local subsystem is a MAP user.
	`natResvBit`	National Reserved Bit to be set in SCCP calling party address when sending a message originated from the local subsystem. This parameter can take a value of 1 or 0.
	`nCrndRmtResrc`	Number of remote resources concerned with the status of this local subsystem. In other words, this local subsystem would be communicating with this number of remote subsystems.
	`crndRmtResrc`	List of concerned remote resources that would be communicating with this local subsystem.
Restrictions/ Bugs	None	
Pre-requisites	SCCP_INIT SCCP_ADD_MTP_SAP SCCP_ADD_PC_RESRC SCCP_ADD_RMT_RESRC	
Code Sample	Please refer source file demo/sccp.c	

14.21 SCCP_DEL_SAP

Description	This API is used to delete an already configured SCCP SAP	
Prototype	`sccp_s32 sccp_Del_SAP ( sccp_u32 sccp_sap_id );`	
Parameters	Associated C Data Structures	
	`sccp_sap_id`	This parameter is an unsigned 32 bit variable set to identifier of SCCP Service Access Point. This parameter may take values from 0 to less than SP_MAX_SCCP_SAP. Please note that this SCCP-SAP must have been configured earlier using API SCCP_ADD_SAP.
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT, SCCP_ADD_MTP_SAP, SCCP_ADD_PC_RESRC, SCCP_ADD_RMT_RESRC, SCCP_ADD_SAP	
Code Sample	Please refer source file demo/sccp.c	

14.22 SCCP_GET_SAP_CFG

Description	This API is used to get configuration of an already configured SCCP service access point.	
Prototype	`sccp_s32 sccp_Get_SAP_Cfg ( sccp_u32 sccp_sap_id, sccp_sap_cfg_t *cfg_p );`	
Parameters	Associated C Data Structures	`typedef struct { sccp_u8 ssn; sccp_u32 mtp_sap_id; sccp_u32 app_id; sccp_u32 opt; sccp_u8 natResvBit; sccp_u16 nCrndRmtResrc; sccp_u16 crndRmtResrc[SP_MAX_CNRD_RMT_RESRC]; } sccp_sap_cfg_t;`
	`sccp_sap_id`	Identifier of an already configured SCCP Service Access Point.
	`cfg_p`	This is pointer to the structure in which SCCP layer returns configuration of the SCCP SAP. Please refer API SCCP_ADD_SAP for more details on each parameter of this structure.
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT, SCCP_ADD_MTP_SAP, SCCP_ADD_PC_RESRC, SCCP_ADD_RMT_RESRC, SCCP_ADD_SAP	

Code Sample	Please refer source file demo/sccp.c

14.23 SCCP_GET_SAP_STATUS

Description	This API is used to get status of a local subsystem that is taking services through the specified service access point	

Prototype	`sccp_s32 sccp_Get_SAP_Status`
	`      (`
	`            sccp_u32 sccp_sap_id,`
	`            sccp_u32 *status_p`
	`      );`	

Parameters	Associated C Data Structures	
	`sccp_sap_id`	Identifier of an already configured SCCP service access point through which local subsystem is taking services from the SCCP layer
	`status_p`	This is pointer to the unsigned 32 bits variable in which SCCP layer would return present status of the local-subsystem. The value of remote-subsystem may be one of the following: • `SP_SS_PRHBTD` • `SP_SS_ALLWD`
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT, SCCP_ADD_MTP_SAP, SCCP_ADD_PC_RESRC, SCCP_ADD_RMT_RESRC, SCCP_ADD_SAP	
Code Sample	Please refer source file demo/sccp.c	

14.24 SCCP_ADD_CRND_RMT_RESRC

Description	This API is used for informing SCCP layer that there is a new concerned remote resource for an already configured local subsystem. Please note that this API must not be used if remote-subsystem-resource has already been configured as a concerned remote resource while configuring the SCCP SAP. This API must be used if remote-subsystem-resource has been added after addition of the SCCP SAP and remote-subsystem-resource would be communicating with the SCCP SAP.	

Prototype	`sccp_s32 sccp_Add_Concerned_RmtResrc`
	`      (`
	`            sccp_u16 rmt_resrc_id,`
	`            sccp_u32 sccp_sap_id`
	`      );`	

Parameters	Associated C Data Structures	
	`rmt_resrc_id`	This parameter is an unsigned 16 bit variable set to identifier of an already configured remote-subsystem-

		resource.
	`sccp_sap_id`	This parameter is an unsigned 32 bit variable set to identifier of an already configured SCCP service access point.
Restrictions/ Bugs	None	
Pre-requisites	SCCP_INIT, SCCP_ADD_MTP_SAP, SCCP_ADD_PC_RESRC, SCCP_ADD_RMT_RESRC	
Code Sample	Please refer source file demo/sccp.c	

14.25 SCCP_ADD_CRND_PC_RESRC

Description	This API is used for informing SCCP layer that there is a new concerned remote point code resource for an already configured remote subsystem. Please note that this API must not be used if remote-PC-resource has already been configured as a concerned PC resource while configuring the remote subsystem. This API must be used if remote-PC-resource has been added after addition of the remote subsystem and remote-PC-resource needs to be communicated about status changes of the remote-subsystem.	

Prototype	`sccp_s32 sccp_Add_Concerned_PcResrc`
	`      (`
	`            sccp_u16 pc_resrc_id,`
	`            sccp_u16 rmt_resrc_id`
	`      );`	

Parameters	Associated C Data Structures	
	`pc_resrc_id`	This parameter is an unsigned 16 bit variable set to identifier of an already configured remote-PC-resource.
	`rmt_resrc_id`	This parameter is an unsigned 16 bit variable set to identifier of an already configured remote-subsystem-resource.
Restrictions/ Bugs	None	
Pre-requisites	SCCP_INIT, SCCP_ADD_MTP_SAP, SCCP_ADD_PC_RESRC, SCCP_ADD_RMT_RESRC	
Code Sample	Please refer source file demo/sccp.c	

15 Tracing Management API (LM→SCCP)

The main purpose of SCCP traces is to provide help in offline debugging. There are various trace levels as specified in Table 5 for logging different types of conditions.

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

Table 5: SCCP Trace Levels

15.1 SCCP_SET_TRACEMAP

Description	This API is used to enable one or more trace levels together for SCCP layer. For example to enable warning and error traces, trace level must be set to value `SP_WARN	SP_ERROR`.	
Prototype	`void sccp_set_trace_map ( sccp_u32 traceMap );`	
Parameters	`traceMap`	Trace levels that are to be enabled in the SCCP 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 = SP_ERROR	SP_MSG;`
Restrictions/Bugs	Presently, tracing support is not complete. Logs are still being built in the SCCP library.	
Pre-requisites		

15.2 SCCP_GET_TRACEMAP

Description	This API is used to query the trace levels presently enabled in the SCCP layer.
Prototype	`sccp_u32 sccp_get_trace_map (`

	`void`
	`);`

Return Value	The trace map of SCCP layer is return value of this API. The return value is an unsigned 32 bit parameter.
Restrictions/Bugs	Presently, tracing support is not complete. Logs are still being built in the SCCP library.
Pre-requisites	

15.3 SCCP_SET_TRACE_LEVEL

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

Prototype	`void sccp_set_trace_level`
	`(`
	`sccp_u32 traceLvl`
	`);`	

Parameters	`traceLvl`	Trace level that is to be enabled in the SCCP library. It can take any value as specified in Table 5.
Restrictions/Bugs	Presently, tracing support is not complete. Logs are still being built in the SCCP library.	
Pre-requisites		

16 Layer Management API (SCCP→LM)

16.1 SCCP_MGMT_NTFY

Description	This API is used to collect management notifications from SCCP. This API must be invoked every time after an external event has been passed to the SCCP layer. For example, this API must be invoked after invoking any other SCCP API that is used to pass an MTP/M3UA event or after passing a timer tick to the SCCP layer. The list of APIs after which SCCP_MGMT_NTFY API must be invoked is as following: • SCCP_MTP_TRANSFER_IND • SCCP_MTP_PAUSE_IND • SCCP_MTP_RESUME_IND • SCCP_MTP_STATUS_IND • SCCP_CHECK_TIMER_EXPIRY A timer tick can result in an internal timer expiry that can be a notification to the layer management. This API must be repeatedly invoked until it returns failure. This API is used to notify following conditions:

	1. Hop Counter Violation 2. Upper Layer Message Returned for Prohibited Subsystem 3. Lower Layer Message Returned for Prohibited Subsystem	

Prototype	`sccp_s32 sccp_mgmt_ntfy`
	`       (`
	`              sccp_mgmt_ntf_t *ntf_p`
	`       );`	

Parameters	Associated C Data Structures	`typedef struct`
		`{`
		`        sccp_u16 mtp_sap_id;`
		`        spaddr_t cgAdd;`
		`        spaddr_t cdAdd;`
		`        spMtpRtLbl_t rtLbl;`
		`} sccp_mgntf_ll_hpctr_vltd_t;`
		
		`typedef struct`
		`{`
		`        sccp_u16 mtp_sap_id;`
		`        spaddr_t cgAdd;`
		`        spaddr_t cdAdd;`
		`        spMtpRtLbl_t rtLbl;`
		`} sccp_mgntf_ll_ssp_t;`
		
		`typedef struct`
		`{`
		`        sccp_u16 sccp_sap_id;`
		`        spaddr_t cgAdd;`
		`        spaddr_t cdAdd;`
		`} sccp_mgntf_ul_ssp_t;`
		
		`typedef struct`
		`{`
		`        sccp_u32 ntf_type;`
		`        union`
		`        {`
		`                sccp_mgntf_ll_hpctr_vltd_t ll_hpctr_vldtd;`
		`                sccp_mgntf_ll_ssp_t ll_ssp;`
		`                sccp_mgntf_ul_ssp_t ul_ssp;`
		`        };`
		`} sccp_mgmt_ntf_t;`

	`ntf_type`	Type of management notification reported by SCCP layer. This parameter is used to determine the valid parameters in the union. This is an unsigned 32 bit parameter with following values: ● `SP_MN_LL_MSGRET_HPCTR_VLTD` – Hop counter violation ● `SP_MN_LL_MSGRET_SSP` – Message received for prohibited subsystem from Lower layer [MTP/M3UA] ● `SP_MN_UL_MSGRET_SSP` – Message received for prohibited subsystem from Upper layer [local subsystem]
	Case I – If `ntf_type` equal to `SP_MN_LL_MSGRET_HPCTR_VLTD` then only parameter structure "`ll_hpctr_vldtd`" is to be interpreted.	
	`mtp_sap_id`	MTP-SAP identifier from which message is received causing hop counter violation
	`cgAdd`	Calling address present in the message
	`cdAdd`	Called address present in the message
	`rtLbl`	MTP routing label present in the message
	Case II – If `ntf_type` equal to `SP_MN_LL_MSGRET_SSP` then only parameter	

	structure “ll_ssp” is to be interpreted.	
	mtp_sap_id	MTP-SAP identifier from which message is received for prohibited subsystem
	cgAdd	Calling address present in the message
	cdAdd	Called address present in the message
	rtLbl	MTP routing label present in the message
	Case III – If ntf_type equal to SP_MN_UL_MSGRET_SSP then only parameter structure “ul_ssp” is to be interpreted.	
	sccp_sap_id	SCCP service access point from which message is received for the prohibited subsystem
	cgAdd	Calling address supplied by local subsystem
	cdAdd	Called address supplied by local subsystem
Restrictions/ Bugs		
Pre-requisites	SCCP_INIT	
Code Sample	Please refer source file demo/main.c	

17 User APIs (USER→SCCP)

Following APIs are used by SCCP Subsystems to take services provided by the SCCP Layer.

17.1 SCCP_N_UDT_REQ

Description	This API is used by SCCP subsystems to transfer a connectionless message to the appropriate destination subsystem. The message may be sent using a UDT or XUDT or LUDT message.	

Prototype	`sccp_s32 sccp_N_UDT_REQ`
	`(`
	`sccp_u32 sccp_sap_id,`
	`spaddr_t *cdAdd_p,`
	`spaddr_t *cgAdd_p,`
	`sccp_u16 seqCtl,`
	`bool retOpt,`
	`spdata_t *pUsrDat,`
	`sccp_u8 imp,`
	`sccp_u8 hopCtr`
	`);`	

Parameters	Associated C Data Structures	`typedef struct`
		`{`
		`spaddrInd_t ai;`
		`sccp_u8 ssn;`
		`sccp_u32 pc;`
		`spgt_t gt;`

		`} spaddr_t;`
		
		`typedef enum`
		`{`
		`    SP_AI_RTE_GT = 0,`
		`    SP_AI_RTE_SSN = 1`
		`} sprouteInd_t;`
		
		`typedef struct`
		`{`
		`    sccp_u32 len;`
		`    sccp_u8 *data_p;`
		`} spdata_t;`
		
		`typedef struct`
		`{`
		`    sccp_u8 natResv:1;`
		`    sccp_u8 routeInd:1;`
		`    sccp_u8 gtInd:4;`
		`    sccp_u8 ssnInd:1;`
		`    sccp_u8 pcInd:1;`
		`} spaddrInd_t;`
		
		`typedef struct`
		`{`
		`    sccp_u8 nai;`
		`    sccp_u8 es;`
		`    sccp_u8 np;`
		`    sccp_u8 tt;`
		`    sccp_u8 numDigits;`
		`    sccp_u8 digits[SP_MAX_GT_DIGITS];`
		`} spgt_t;`

	`sccp_sap_id`	SCCP service access point through which connectionless message is being passed to the SCCP layer.
	`cdAdd_p`	Pointer to Called party 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 "`SP_INV_PC`" [numeric value 0xFFFFFFFF] if point code is not available.
	`cgAdd_p`	Pointer to Calling party address. This address is encoded as it is in the outgoing connectionless SCCP message.
	`seqCtl`	Sequence control parameter associated with this message transfer. To maintain sequence of messages, all the messages which are part of the sequence must have same value for this parameter. In case sequencing is not important, this parameter must be set to value "`SP_NO_SEQCTL`" [numeric value 256].
	`retOpt`	This parameter specifies if message is to be returned back in case of an error. This parameter takes two values as following: • `true` – Return option is set, so message would be returned back in case of error

		• `false` – return option is not set, so message would not be returned back on error
	`pUsrDat`	User data that is to be sent in the SCCP connectionless message. In case the size of user data is too large for a single UDT message, then segmentation would take place and more than one XUDT messages would be sent out.
	`imp`	Importance parameter for the XUDT message. Valid values for this parameter are 0 to 7. If this parameter is set to "`SP_INV_IMP`" [numeric value 255], then SCCP first tries to deliver message using UDT. In case this parameter is set to a value other than "`SP_INV_IMP`", then XUDT message with optional parameter "Importance" is used.
	`hopCtr`	Hop Counter parameter for the XUDT message. Valid values for this parameter are 0 to 15. If this parameter is set to "`SP_INV_HOPCTR`", then SCCP first tries to deliver message using UDT. In case this parameter is set to a value other than "`SP_INV_HOPCTR`", then XUDT message is used.
Restrictions/ Bugs	None	
Pre-requisites	SCCP_INIT, SCCP_ADD_MTP_SAP, SCCP_ADD_PC_RESRC, SCCP_ADD_RMT_RESRC, SCCP_ADD_SAP, SCCP_N_STATE_REQ	
Code Sample	Please refer source file demo/sccp.c	

17.2 SCCP_N_STATE_REQ

Description	This API is used by local SCCP subsystems to inform there own state change to the SCCP layer.	

Prototype	`sccp_s32 sccp_N_State_Req`
	`       (`
	`              sccp_u16 sccp_sap_id,`
	`              sccp_u32 status`
	`       );`	

Parameters	Associated C Data Structures	
	`sccp_sap_id`	SCCP service access point through which local subsystem is taking services.
	`status`	Status of the local subsystem: • `SP_SS_PRHBTD` • `SP_SS_ALLWD`
Restrictions/ Bugs	None	
Pre-requisites	SCCP_INIT, SCCP_ADD_MTP_SAP, SCCP_ADD_PC_RESRC, SCCP_ADD_RMT_RESRC, SCCP_ADD_SAP	
Code Sample	Please refer source file demo/sccp.c	

17.3 SCCP_N_COORD_SOR

Description	This API is used by local SCCP subsystem to initiate a coordinated subsystem out-of-service request	

Prototype	`sccp_s32 sccp_N_COORD_SOR`
	`(`
	`sccp_u32 sccp_sap_id,`
	`sccp_u16 pc_resrc_id`
	`);`	

Parameters	Associated C Data Structures	
	`sccp_sap_id`	SCCP service access point through which local subsystem is taking services. This is the local subsystem that wishes to go out of service and wants to take permission from the replicated subsystem.
	`pc_resrc_id`	PC-resource where the replicated subsystem resides.
Restrictions/ Bugs	None	
Pre-requisites	SCCP_INIT, SCCP_ADD_MTP_SAP, SCCP_ADD_PC_RESRC, SCCP_ADD_RMT_RESRC, SCCP_ADD_SAP	
Code Sample	Please refer source file demo/sccp.c	

17.4 SCCP_N_COORD_SOG

Description	This API is used by local SCCP subsystem to grant its replicated subsystem to go out of service. This API must be invoked only when SCCP layer notifies that replicated subsystem is asking for permission to go out of service.	

Prototype	`sccp_s32 sccp_N_COORD_SOG`
	`(`
	`sccp_u32 sccp_sap_id,`
	`sccp_u16 pc_resrc_id`
	`);`	

Parameters	Associated C Data Structures	
	`sccp_sap_id`	SCCP service access point through which local subsystem is taking services. This is the local subsystem that has received notification from its replicated subsystem for permission to go out of service.
	`pc_resrc_id`	PC-resource where the replicated subsystem requesting for going out of service resides.
Restrictions/ Bugs	None	
Pre-	`SCCP_INIT`	

requisites	`SCCP_ADD_MTP_SAP`
	`SCCP_ADD_PC_RESRC`
	`SCCP_ADD_RMT_RESRC`
	`SCCP_ADD_SAP`
	`SCCP_MGMT_NTFY(SP_UN_COORD_IND)`

Code Sample	Please refer source file demo/sccp.c

18 User APIs (SCCP→User)

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

18.1 SCCP_USER_NTFY

Description	This API is used to collect notifications for local subsystems from SCCP. This API must be invoked every time after an external event has been passed to the SCCP layer. For example, this API must be invoked after invoking any other SCCP API that is used to pass an MTP/M3UA event or after passing a timer tick to the SCCP layer. The list of APIs after which SCCP_USER_NTFY API must be invoked is as following: • SCCP_MTP_TRANSFER_IND • SCCP_MTP_PAUSE_IND • SCCP_MTP_RESUME_IND • SCCP_MTP_STATUS_IND • SCCP_CHECK_TIMER_EXPIRY A timer tick can result in an internal timer expiry that can be a notification to the layer management. This API must be repeatedly invoked until it returns failure. This API is used to notify following conditions: 1. Receipt of connectionless data 2. Error condition using notice indication 3. PC state change 4. Remote subsystem state change 5. Remote SCCP state change 6. Local subsystem state change 7. Receipt of connectionless data after successful re-assembly of all XUDT segments 8. Receipt of request from replicated subsystem for going out of service 9. Outcome of coordinated subsystem out-of-service request

Prototype	`sccp_s32 sccp_user_ntfy`
	`      (`
	`            sccp_user_ntf_t *ntf_p`

	);	
Parameters	Associated C Data Structures	

```
typedef struct
{
        spaddr_t cdAdd;
        spaddr_t cgAdd;
        spdata_t data;
        sccp_u8 imp;
        sccp_u8 prtCls;
        spMtpRtLbl_t rtLbl;
} sccp_unitdata_ind_t;

typedef struct
{
        spaddr_t cdAdd;
        spaddr_t cgAdd;
        sccp_u32 len;
        sccp_u8 data[SCCP_MAX_USR_DAT_SZ];
        sccp_u8 imp;
        sccp_u8 prtCls;
        spMtpRtLbl_t rtLbl;
} sccp_ras_unitdata_ind_t;

typedef struct
{
        sccp_pc_status_name status;
        sccp_u16 pc_resrc_id;
        sccp_u16 mtp_sap_id;
        sccp_u32 rstImpLvl;
        sccp_u32 pc;
} sccp_pc_status_ind_t;

typedef struct
{
        sccp_ss_status_name status;
        sccp_u16 rmt_resrc_id;
        sccp_u32 pc;
        sccp_u8 ssn;
        sccp_u32 mtp_sap_id;
} sccp_rmt_resrc_status_ind_t;

typedef struct
{
        sccp_ss_status_name status;
        sccp_u8 ssn;
        sccp_u32 mtp_sap_id;
} sccp_lcl_ss_status_ind_t;

typedef struct
{
                sccp_status_name status;
                sccp_u16 pc_resrc_id;
                sccp_u32 pc;
                sccp_u16 mtp_sap_id;
} sccp_status_ind_t;


typedef struct
{
        sccp_u8 retCause;
        spaddr_t cdAdd;
        spaddr_t cgAdd;
        sccp_u16 datSz;
        sccp_u8 usrDat[SCCP_MAX_USR_DAT_SZ];
} sccp_notice_ind_t;

typedef struct
{
        sccp_u32 pc;
        sccp_u16 mtp_sap_id;
        sccp_u16 pc_resrc_id;
} sccp_coord_ind_t;

typedef struct
{
        sccp_u32 status;
} sccp_coord_status_ind_t;

typedef struct
{
        sccp_u32 ntf_type;
        sccp_u16 sccp_sap_id;
        sccp_u16 app_id;
        union
        {
                sccp_unitdata_ind_t udt;
                sccp_pc_status_ind_t pc_stat;
                sccp_rmt_resrc_status_ind_t rmt_rsrc_stat;
                sccp_status_ind_t sccp_stat;
                sccp_lcl_ss_status_ind_t lcl_ss_stat;
                sccp_notice_ind_t ntc;
                sccp_ras_unitdata_ind_t ras_udt;
                sccp_coord_status_ind_t coord_status;
                sccp_coord_ind_t coord;
        };
} sccp_user_ntf_t;
```

	`ntf_type`	Type of user notification reported by SCCP layer. This parameter is used to determine the valid parameters in the union. This is an unsigned 32 bit parameter with following values: • `SP_UN_UDT_IND` – Receipt of connectionless data • `SP_UN_NOT_IND` – Error condition • `SP_UN_PC_STATE` – PC State change • `SP_UN_RMT_SS_STATE` – Remote subsystem state change • `SP_UN_RMT_SCCP_STATE` – Remote SCCP state change • `SP_UN_LCL_SS_STATE` – Local subsystem state change • `SP_UN_RAS_UDT_IND` – Reassembled connectionless data • `SP_UN_COORD_IND` – Coordinated out-of-service request received • `SP_UN_COORD_STATUS_IND` – Outcome of coordinated out-of-service request
	`sccp_sap_id`	SCCP service access point identifier for which this notification has been reported by the SCCP layer
	`app_id`	Application identifier of the local subsystem
	Case I – If `ntf_type` equal to `SP_UN_UDT_IND` then parameter structure “udt” of the union is to be interpreted.	
	`cgAdd`	Calling address present in the message
	`cdAdd`	Called address present in the message
	`data`	Connectionless data received in the message
	`imp`	Importance parameter as received in XUDT or LUDT message
	`prtCls`	Protocol class of the connectionless message

	`rtLbl`	MTP routing label present in the message
	Case II – If `ntf_type` equal to `SP_UN_NOT_IND` then parameter structure "`ntc`" of the union is to be interpreted.	
	`retCause`	Return cause as per SCCP specification Q.713
	`cgAdd`	Calling address present in the message
	`cdAdd`	Called address present in the message
	`datSz`	Size of user data
	`usrDat`	User data returned in UDTS/XUDTS/LUDTS message
	Case III – If `ntf_type` equal to `SP_UN_PC_STATE` then parameter structure "`pc_stat`" of the union is to be interpreted.	
	`status`	Present status of the point code resource. The values for this parameter have been explained in API SCCP_ADD_PC_RESRC.
	`pc_resrc_id`	PC-resource identifier
	`mtp_sap_id`	MTP-SAP identifier through which PC-resource is accessed
	`rstImpLvl`	Restricted importance level associated with the PC-resource
	`pc`	Point code of the PC-resource
	Case IV – If `ntf_type` equal to `SP_UN_RMT_SS_STATE` then parameter structure "`rmt_rsrc_stat`" of the union is to be interpreted.	
	`status`	Present status of the remote subsystem. Valid values for this parameter are: • `SP_SS_PRHBTD` • `SP_SS_ALLWD`
	`rmt_resrc_id`	Remote-resource identifier
	Case V – If `ntf_type` equal to `SP_UN_RMT_SCCP_STATE` then parameter structure "`sccp_stat`" of the union is to be interpreted.	
	`status`	Present status of the SCCP at the remote-PC-resource: • `SP_SCCP_UNAV_UNK` – SCCP is unavailable due to unknown reason • `SP_SCCP_UNEQP` – SCCP is unequipped • `SP_SCCP_INACC` – SCCP is inaccessible • `SP_SCCP_CNGSTD` – SCCP is congested • `SP_SCCP_AVBL` – SCCP is available
	`pc_resrc_id`	PC-resource identifier where SCCP whose status is being reported resides
	Case VI – If `ntf_type` equal to `SP_UN_LCL_SS_STATE` then parameter structure "`lcl_ss_stat`" of the union is to be interpreted.	
	`status`	Status of the local subsystem. The valid values for this parameter are: • `SP_SS_PRHBTD` • `SP_SS_ALLWD`
	`ssn`	Subsystem number of the local subsystem

	`mtp_sap_id`	MTP-SAP identifier associated with the local subsystem
	Case VII – If `ntf_type` equal to `SP_UN_RAS_UDT_IND` then parameter structure "`ras_udt`" of the union is to be interpreted.	
	`cgAdd`	Calling address present in the message
	`cdAdd`	Called address present in the message
	`len`	number of bytes of user data
	`data`	Connectionless data received in the message
	`imp`	Importance parameter as received in XUDT message
	`prtCls`	Protocol class of the connectionless message
	`rtLbl`	MTP routing label present in the message
	Case VIII – If `ntf_type` equal to `SP_UN_COORD_IND` then parameter structure "`coord`" of the union is to be interpreted. This notification means that the replicated subsystem is asking for permission to go out of service.	
	`pc_resrc_id`	remote-PC-resource identifier where replicated subsystem resides
	Case IX – If `ntf_type` equal to `SP_UN_COORD_STATUS_IND` then parameter structure "`coord_status`" of the union is to be interpreted. This notification is used to inform the local subsystem if it can go out-of-service or not after it has invoked the API N_COORD_SOR.	
	`status`	This parameter notifies the final outcome of the coordinated out-of-service procedure. The values for this parameter are: • `SP_COORD_STATUS_GRANTED` • `SP_COORD_STATUS_DENIED`
Restrictions/ Bugs	None	
Pre-requisites	SCCP_INIT	
Code Sample	Please refer source file demo/main.c	

19 SCCP to MTP/M3UA API

19.1 SCCP_SENDMSG

Description	This API is used to pass SCCP messages to underlying MTP/M3UA 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 MTP/M3UA layer.

Prototype	`sccp_s32 sccp_sendmsg`
	`      (`
	`            sccp_u16 mtp_sap_id,`
	`            sccp_u32 dpc,`
	`            sccp_u32 opc,`
	`            sccp_u8 sls,`
	`            sccp_u8 mp,`

	`sccp_u8 ni,`
	`sccp_u8 si,`
	`sccp_u8 *msg,`
	`sccp_u32 msgLen`
	`);`	

Parameters	`mtp_sap_id`	MTP-SAP identifier through which SCCP message is to be transferred
	`dpc`	Destination Point Code
	`opc`	Originating Point Code
	`sls`	Signaling Link Selection
	`mp`	Message Priority
	`ni`	Network Indicator
	`si`	Service Indicator
	`msg`	Pointer to Message buffer
	`msgLen`	Number of user data bytes in message buffer
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT, SCCP_ADD_MTP_SAP, SCCP_ADD_PC_RESRC, SCCP_ADD_RMT_RESRC	
Code Sample	Please refer source file demo/sccp.c	

20 MTP/M3UA to SCCP API

This section lists APIs that need to be invoked when an event or notification is received from the underlying MTP/M3UA layer.

20.1 SCCP_MTP_PAUSE_IND

Description	This API is used to pass MTP-PAUSE notification to the SCCP layer. Whenever MTP-PAUSE notification from MTP/M3UA layer is received, it must be passed to the SCCP layer using this API.	

Prototype	`sccp_s32 sccp_MTP_PAUSE_IND`
	`(`
	`sccp_u16 mtp_sap_id,`
	`sccp_u32 *aPcList,`
	`sccp_u32 nPc`
	`);`	

Parameters	`mtp_sap_id`	MTP-SAP identifier that has reported MTP-PAUSE notification
	`aPcList`	List of point codes for which MTP-PAUSE has been received
	`nPc`	Number of point codes for which MTP-PAUSE has been received
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT, SCCP_ADD_MTP_SAP, SCCP_ADD_PC_RESRC	
Code Sample	Please refer source file demo/sccp.c	

20.2 SCCP_MTP_RESUME_IND

Description	This API is used to pass MTP-RESUME notification to the SCCP layer. Whenever MTP-RESUME notification from MTP/M3UA layer is received, it must be passed to the SCCP layer using this API.	
Prototype	`sccp_s32 sccp_MTP_RESUME_IND ( sccp_u16 mtp_sap_id, sccp_u32 *aPcList, sccp_u32 nPc );`	
Parameters	`mtp_sap_id`	MTP-SAP identifier that has reported MTP-RESUME notification
	`aPcList`	List of point codes for which MTP-RESUME has been received
	`nPc`	Number of point codes for which MTP-RESUME has been received
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT, SCCP_ADD_MTP_SAP, SCCP_ADD_PC_RESRC	
Code Sample	Please refer source file demo/sccp.c	

20.3 SCCP_MTP_STATUS_IND

Description	This API is used to pass MTP-STATUS notification to the SCCP layer. Whenever MTP-STATUS notification from MTP/M3UA layer is received, it must be passed to the SCCP layer using this API.	
Prototype	`sccp_s32 sccp_MTP_STATUS_IND ( sccp_u16 mtp_sap_id, sccp_u8 cause, sccp_u32 affPC, sccp_u32 addInfo );`	
Parameters	`mtp_sap_id`	MTP-SAP identifier that has reported MTP-RESUME notification
	`cause`	Cause value associated with the MTP-STATUS primitive. The values for this parameter are: • `SCCP_MTP_STATUS_CAUSE_UNK` – Status indication received for remote SCCP unavailable for unknown reason • `SCCP_MTP_STATUS_CAUSE_UNEQP` – Status indication received for remote SCCP unavailable as it is unequipped • `SCCP_MTP_STATUS_CAUSE_INACC` -- Status indication received for remote SCCP

		unavailable as it is inaccessible • SCCP_MTP_STATUS_CONG – remote point code is congested
	affPC	Affected PC where SCCP resides or where congestion has taken place
	addInfo	This parameter is only used when remote point code congestion indication is notified. This parameter indicates the congestion level.
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT, SCCP_ADD_MTP_SAP, SCCP_ADD_PC_RESRC	
Code Sample	Please refer source file demo/sccp.c	

20.4 SCCP_MTP_TRANSFER_IND

Description	This API is used to pass data received from MTP/M3UA layer for SCCP layer. This API must be invoked after MTP-TRANSFER-IND for SCCP layer is received to pass the data and associated information to the SCCP layer.	
Prototype	`sccp_s32 sccp_MTP_TRANSFER_IND ( sccp_u32 mtp_sap_id, spMtpRtLbl_t rtLbl, sccp_u8 *msg, sccp_u32 msgLen );`	
Parameters	mtp_sap_id	MTP-SAP identifier that has received data for SCCP layer
	rtLbl	MTP route label received with the SCCP message
	msg	Message buffer containing SCCP message
	msgLen	Length of SCCP message received
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT, SCCP_ADD_MTP_SAP, SCCP_ADD_PC_RESRC, SCCP_ADD_SAP	
Code Sample	Please refer source file demo/sccp.c	

21 Timing Management API

21.1 SCCP_ CHECK_TIMER_EXPIRY

Description	This API is used for supplying timer ticks to the SCCP library. This means that this API must be invoked at regular intervals. As SCCP timers are in range of 100ms, so it is recommended that this API is invoked at least once every 100ms.
Prototype	`sccp_s32 sccp_check_timer_expiry`

	`(`
	`        void`
	`);`	

Parameters		
Restrictions/Bugs	None	
Pre-requisites	SCCP_INIT	
Code Sample	Please refer source file demo/main.c	

22 Data Structures Associated with the SCCP APIs

All the data structures & values that would be used by application are present in following SCCP include files:

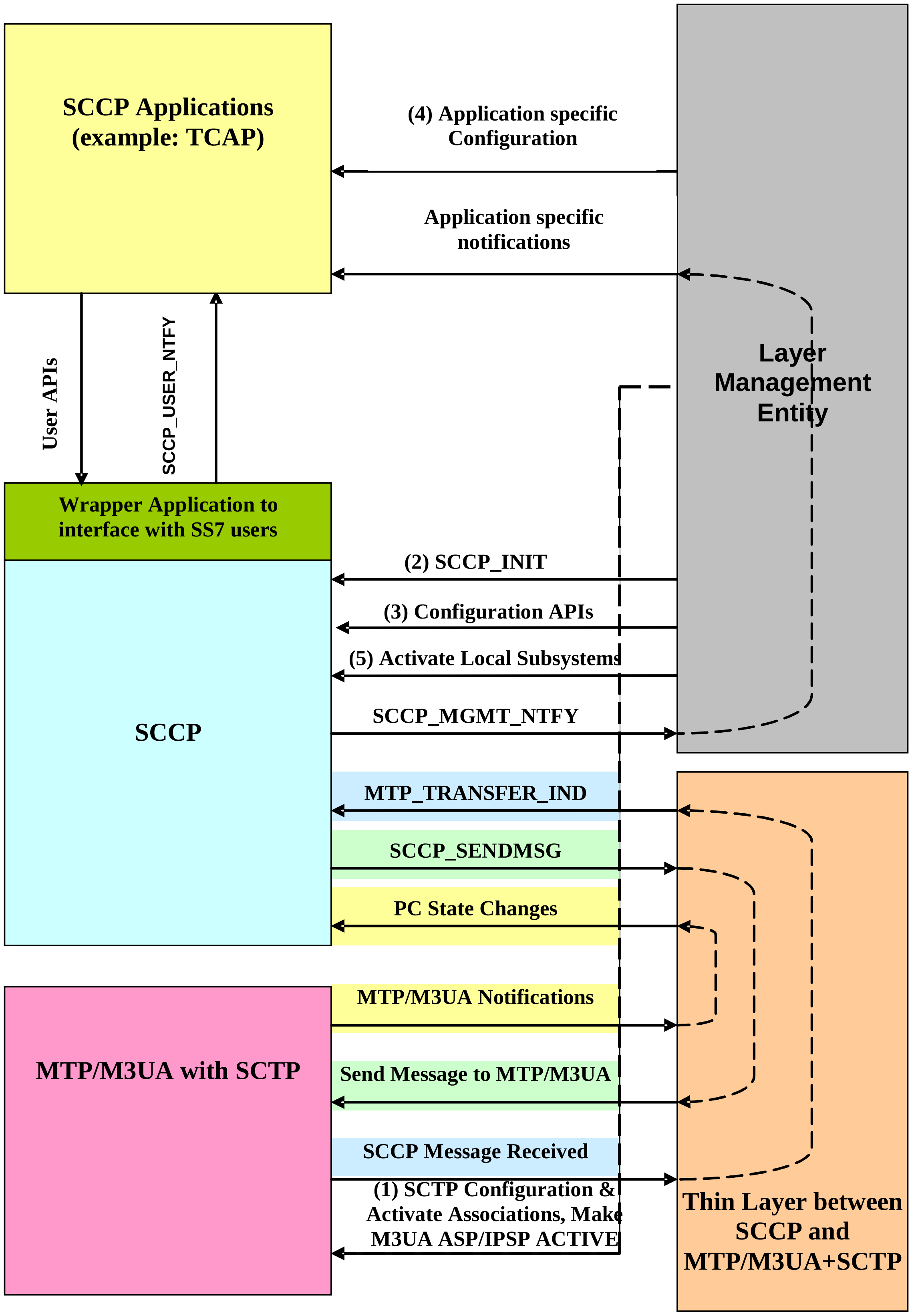
1. **`spdef.h`**
2. **`sptyp.h`**
3. **`sperr.h`**
4. **`spprt.h`**
5. **`spapi.h`**

23 Integrating SCCP with Third Party Components (MTP/M3UA, SCTP and TCAP)

The application developer would need to integrate the SCCP layer with its layer management interface to configure SCCP layer. First, MTP/M3UA and SCTP would be configured & made active, followed by SCCP layer followed by SS7 user parts like TCAP and other TCAP based applications like MAP or CAP. Layer management entity would also handle various notifications from SCCP layer and translate them into relevant information for SCCP users and operator.

A wrapper code (light weight application) around SCCP layer shall be written to adapt SCCP primitives & notifications into format understood by the SCCP users. This wrapper code would pass primitives between SCCP users [example TCAP] and SCCP. Similarly, a thin wrapper layer between SCCP and MTP/M3UA would be required to facilitate passing of messages between MTP/M3UA and SCCP layers.

Following illustration shows how SCCP may be integrated with MTP/M3UA and SCTP (transport layer) and SS7 user parts.


SCCP Applications (example: TCAP)
(4) Application specific Configuration
Application specific notifications
User APIs
SCCP_USER_NTFY
Layer Management Entity
Wrapper Application to interface with SS7 users
(2) SCCP_INIT
(3) Configuration APIs
(5) Activate Local Subsystems
SCCP
SCCP_MGMT_NTFY
MTP_TRANSFER_IND
SCCP_SENDMSG
PC State Changes
MTP/M3UA Notifications
MTP/M3UA with SCTP
Send Message to MTP/M3UA
SCCP Message Received
(1) SCTP Configuration & Activate Associations, Make M3UA ASP/IPSP ACTIVE
Thin Layer between SCCP and MTP/M3UA+SCTP

This page is intentionally left blank.