|  |  |
| --- | --- |
|  |  |

SCCP Protocol Emulation for TTCN-3 Toolset with TITAN, Description

# Abstract

The SCCP PE is developed for testing implementations of SCCP Users using TTCN-3 and it uses the services of underlying level MTP3. It is considered that the SCCP layer of the peer conforms to the same specifications as SCCP PE does.

Contents

[1 About this Document 2](#_Toc350931036)

[1.1 How to Read this Document 2](#_Toc350931037)

[1.2 Presumed Knowledge 2](#_Toc350931038)

[2 Functionality 2](#_Toc350931039)

[2.1 Implemented protocols 3](#_Toc350931040)

[2.2 Supported Standards 4](#_Toc350931041)

[2.3 Modified and non-implemented Protocol Elements 4](#_Toc350931042)

[2.3.1 Missing primitives 4](#_Toc350931043)

[2.3.2 Routing Functionality and Global Title Translation omitted 4](#_Toc350931044)

[2.3.3 Management messages are only partially implemented 4](#_Toc350931045)

[2.3.4 Flow Control not supported 5](#_Toc350931046)

[2.4 Ericsson-specific changes 5](#_Toc350931047)

[2.5 Backward incompatibilities 5](#_Toc350931048)

[2.6 System Requirements 5](#_Toc350931049)

[3 Feature list 5](#_Toc350931050)

[3.1 Service Classes 5](#_Toc350931051)

[3.2 MTP3 Service Primitives 5](#_Toc350931052)

[3.3 SCCP Messages 7](#_Toc350931053)

[3.4 SCCP Primitives of the Connectionless Service 9](#_Toc350931054)

[3.5 SCCP Primitives for Connection-oriented Services 10](#_Toc350931055)

[3.6 SCCP Management Functionality 11](#_Toc350931056)

[3.7 Inactivity Control 11](#_Toc350931057)

[3.8 Message Sequence Control 11](#_Toc350931058)

[3.9 Segmentation and Reassembly 11](#_Toc350931059)

[3.10 State Machine 11](#_Toc350931060)

[3.11 Global Title Translation 11](#_Toc350931061)

[3.12 Capacity and Limitation 12](#_Toc350931062)

[3.13 Differences between ITU, ANSI, MPT and TTC 12](#_Toc350931063)

[4 Test Port Usage 13](#_Toc350931064)

[4.1 The User Interface: the N-Service Primitives 14](#_Toc350931065)

[4.1.1 SCCP Primitives of Connectionless Service 14](#_Toc350931066)

[4.1.2 SCCP Primitives of Connection Oriented Service 15](#_Toc350931067)

[4.2 The MTP3-Service Primitives 16](#_Toc350931068)

[4.2.1 Choosing between protocol standards resp. versions 17](#_Toc350931069)

[4.2.2 Forced sending of XUDT messages 17](#_Toc350931070)

[4.3 Installation 17](#_Toc350931071)

[4.3.1 Description of files implementing the SCCP PE 17](#_Toc350931072)

[4.4 Configuration 18](#_Toc350931073)

[4.4.1 SCCP PE parameters in the Protocol Emulation configuration file 18](#_Toc350931074)

[4.4.2 Arguments of function SCCPStart 19](#_Toc350931075)

[4.5 Makefile 20](#_Toc350931076)

[4.6 Error messages 20](#_Toc350931077)

[4.7 Warning messages 20](#_Toc350931078)

[4.8 Examples 20](#_Toc350931079)

[4.8.1 Example 1 20](#_Toc350931080)

1. [4.8.1.1 Configuration file 1 21](#_Toc350931081)
2. [4.8.1.2 Test Suite 1 22](#_Toc350931082)

[4.8.2 Example 2 30](#_Toc350931083)

1. [4.8.2.1 Configuration file 2 31](#_Toc350931084)
2. [4.8.2.2 Test Suite 2 31](#_Toc350931085)

[5 Terminology 38](#_Toc350931086)

[5.1 Abbreviations 38](#_Toc350931087)

[6 References 39](#_Toc350931088)

[7 Change information 40](#_Toc350931089)

# About this Document

## How to Read this Document

This document is for the SCCP Protocol Emulation (SCCP PE). The SCCP PE is developed for the TTCN-3 Toolset with TITAN according to the Requirement Specification [4]. This document should to be read together with Product Revision Information [5].

## Presumed Knowledge

The knowledge of the TITAN TTCN-3 Test Executor [3] and the TTCN-3 language [1] is essential.

# Functionality

“The Signalling Connection Control Part (SCCP) provides additional functions to the Message Transfer Part (MTP) to cater for both connectionless as well as connection-oriented services to transfer … signalling information” [7].

The SCCP protocol emulation (PE) implements the SCCP protocol specified by ITU-T (see [7]- [10]), ANSI ([11]), MPT and TTC ([12]-[15]). The emulation is performed in language TTCN-3 [2] and it supposes the TTCN-3 Test Tool with TITAN test executor [2] as environment.

The SCCP PE is developed for testing implementations of SCCP Users using TTCN and it uses the services of underlying level MTP3 (see in *Figure 1*). It is considered that the SCCP layer of the peer conforms to the same specifications as SCCP PE does.

SCCP-User

Instance 2

SCCP-User

Instance 1

N-Service Primitives

Service access points (SAPs)

SCCP PE

Instance 1

SCCP PE

Instance 2

MTP3-Service Primitives

Service access points (SAPs)

MTP3 or M3UA (and lower layers)

Instance 1

MTP3 or M3UA (and lower layers)

Instance 2

*Figure 1. Service primitives in SS7*

The SCCP (and therefore SCCP PE itself) is situated between the MTP3 and the SCCP-User (see Figure 1). SCCP communicates with them via service primitives.

## Implemented protocols

The SCCP protocol uses the MTP for basic routing and error detection.

Signalling System 7 network:

* Network layer:
  + MTP Level 3 (M3UA – RFC 3332)
  + SCCP (**Q.711 (03/01)**
* Data link
  + MTP Level 2
* Physical layer
  + MTP Level 1

The ITU-T recommendations for the Message Transfer Part are:

* Functional description of the message transfer part (MTP) of Signalling System No. 7 - Q.701 (03/93)
* Signalling data link - Q.702 (11/88)
* Signalling link - Q.703 (07/96)
* Signalling network functions and messages - Q.704 (07/96)
* Signalling network structure - Q.705 (03/93)

The SCCP application can operate directly over the SCCP User Adaption protocol (SUA).

## Supported Standards

SCCP PE implements SCCP specification of ITU, ANSI, MPT and TTC in one module.

Several component instances of the SCCP PE behavior can be used in a TTCN-3 test configuration. Each of the test component instances can behave according to one of the above versions of SCCP (ITU or ANSI or MPT or TTC).

## Modified and non-implemented Protocol Elements

For detailed information see section 3. In section 3.12, you can find a summary of capacity and limitations (including the not implemented parts).

### Missing primitives

The following primitives are not implemented: LUDT, LUDTS, MTP3-PAUSE, MTP3-RESUME and MTP3-STATUS (chapters 3.12 and 3.13).

### Routing Functionality and Global Title Translation omitted

Routing functionality is not implemented: SCCP PE emulates a signaling endpoint. Accordingly, Global Title Translation is not implemented either. (Chapters 3.11 and 3.12)

### Management messages are only partially implemented

The following management messages are not implemented:

**SOG**: Subsystem-out-of-service-grant (1.16/Q.712)

**SOR**: Subsystem-out-of-service-request (1.17/Q.712)

For further details see section 3.6.

### Flow Control not supported

Not implemented by SCCP PE as Service Class 3 is not supported.

## Ericsson-specific changes

Change Request MTTSM00016068 was implemented to support MPT (china) version.

## Backward incompatibilities

-

## System Requirements

In order to operate the SCCP PE the following system requirements must be satisfied:

* TITAN TTCN-3 Test Executor version R7B (1.7.pl1) or higher installed. For installation guide see [2]. Please note: This version of the protocol module is not compatible with TITAN releases earlier than R7B.

# Feature list

## Service Classes

Service primitives are implemented as messages in the test ports.

There are four service classes in SCCP (see 6/Q.711 and 2/T1.112.1-2001) as follows:

0 Basic connectionless class

1 In-sequence delivery connectionless class

2 Basic connection-oriented class

3 Flow control connection-oriented class.

The SCCP PE supports classes 0,1,2 but doesn’t support class 3.

There is no difference between class 0 and 1 because only one test port used by MTP.

## MTP3 Service Primitives

MTP 3 Abstract Service Primitives are received and sent by SCCP across service access points (*Figure 1*) can be found in *Table 1*.

“Not implemented” primitives are discarded by SCCP PE.

*Table1 MTP3 primitives handled by SCCP PE*

The fields are the same for ITU, ANSI, MPT and TTC but their lengths are different as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Length in bits** | | | |
| **Field** | **ITU-T** | **ANSI** | **MPT national\*\*** | **TTC national\*** |
| SIO | 8 | 8 | 8 | 8 |
| DPC | 14 | 24 | 24 | 16 |
| OPC | 14 | 24 | 24 | 16 |
| SLS | 4 | 8 | 4 | 4 |

\*:If SIO sub-service field=0. Otherwise TTC international is the same as ITU-T

\*\*:MPT international is the same as ITU-T

*Table 2. Size of fields in different specifications.*

## SCCP Messages

User data fields of MTP3 primitives received by SCCP from MTP3 (or from M3UA) are mapped to N-service primitives that will be sent to the SCCP User(s).

The User data field of an MTP3 signal unit contains the SCCP message as an octetstream (i.e. an octetstring) in order LSB (lowest bit sent/received first).

The structure and fields of an SCCP message are coded and decoded according to ITU Q.713 [9], ANSI T1.112-2001[11] or TTC JT-Q713 [14].

The first octet of the SCCP message is the message type. Its value determines the decoding of the octetstring further handling. An SCCP message received may be mapped to an N-service primitive or may invoke an exception handling procedure based on the state of the SCCP PE. The supported message types and the related mappings are summarized in *Table 3*.

(Compare it with Table 1/Q.713, Table1/T1.112.3 and Table1/JT-Q713.)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Message type (in MTP-TRANSFER  req)** | **Protocol Classes** | | | **Message type code** | **Handling (depending on SCCP state)** | **Remark** |
| **0** | **1** | **2** |  |  |  |
| CR  Connection request |  |  | X | 0000 0001 | N-CONNECT ind | Not supported by TTC |
| CC Connection confirm |  |  | X | 0000 0010 | N-CONNECT conf  / or Back: ERR / | Not supported by TTC |
| CREF Connection refused |  |  | X | 0000 0011 | N-DISCONNECT (see Q.713.A.1) | Not supported by TTC |
| RLSD  Released |  |  | X | 0000 0100 | Active=>N-DISCONNECT indication Idle OR wait for CC=>Back RLC Otherwise=>discard, log only see Q.714/B.2 | Not supported by TTC |
| RLC  Release complete |  |  | X | 0000 0101 | N-DISCONNECT ind | Not supported by TTC |
| DT1 Data form1 |  |  | X | 0000 0110 | Active=>N-DATA ind OR Conn pending OG=>N-DISCONNECT | Not supported by TTC |
| UDT Unitdata | X | X |  | 0000 1001 | N-UNITDATA indication OR UDT with SSA |  |
| UDTS Unitdata Service | X | X |  | 0000 1010 | N-UNITDATA indication |  |
| ERR Protocol data unit error |  |  | X | 0000 1111 | Idle=>ERR OR  Active=>N-DISCONNECT ind | Not supported by TTC |
| IT  Inactivity Test |  |  | X | 0001 0000 |  | Not supported by TTC |
| XUDT Extended Unitdata | X | X |  | 0001 0001 | N-UNITDATA indication |  |
| XUDTS Extended Unitdata Service | X | X |  | 00010010 | N-UNITDATA indication |  |

*Table 3. Message types implemented by SCCP PE*

This table describes what kind of messages SCCP PE accepts from MTP3 and how they are translated. The translation rule is more complicated than it is described in table 3. Details can be found in Q.714.

## SCCP Primitives of the Connectionless Service

SCCP can provide 2 classes of connectionless services (0 and 1) but there is no difference between them in this implementation (see 6/Q.711 and 2/T1.112.1-2001).

The primitives to the upper layers and the corresponding parameters for connectionless service are implemented as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ITU-T, ANSI and TTC names** | | | **Protocol implementation info** | |
| **Generic name** | **Specific name** | **Parameters** | **ASP name** | **Msg type mapped to OR next msg to be sent back** |
| N-UNITDATA | Request or indication | Called Address  Calling Address Sequence Control Return Option Importance User data | N\_UNITDATA\_req  N\_UNITDATA\_ind | Req=>UDT  From UDT=> Ind |
| N-NOTICE | Indication | Called Address Calling Address Reason for return User Data Importance | N\_NOTICE\_ind | From UDTS |

*Table 4. Primitives and their Mappings for Connectionless Service*

## SCCP Primitives for Connection-oriented Services

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ITU-T, ANSI and TTC** | | | **Protocol implementation info** | |
| **Generic Name** | **Specific name** | **Parameters** | **ASP name** | **Msg type mapped to** |
| N-CONNECT | Request  Indication | Called Address  Calling Address  Responding Address  Expedited selection  Quality of services parameter set  User data  Importance  Connection identification | N\_CONNECT\_req  N\_CONNECT\_ind | Connection request (CR) |
| Response  Confirm | N-CONNECT\_res | Connection confirm (CC) |
| N-DATA | Request  Indication | Importance  User data  Connection identification |  | Data form 1 (DT1) |
| N-DISCONNECT | Request  Indication | Originator  Reason  User data  Responding address  Importance  Connection identification |  | Released (RLSD) OR Connection refusal (CREF) see Q.714/3.3 |

*Table 5 Network service primitives and mapping for connection-oriented services*

## SCCP Management Functionality

There is no interworking between MTP3/M3UA and SCCP management.

SCCP management messages (see 1.15-1.19/Q.712/):

**SSA**: Subsystem-allowed (1.15/ Q.712)

**SOG**: Subsystem-out-of-service-grant (1.16/Q.712)

**SOR**: Subsystem-out-of-service-request (1.17/Q.712)

**SSP**: Subsystem-prohibited (1.18/Q.712)

**SST**: Subsystem-status-test (Q.712/1.19)

These messages are not supported by TTC.

The SCCP management is restricted to the following:

|  |  |
| --- | --- |
| **Received** | **Returned** |
| SST | SSA |
| SSP | SST |
| SSA | SSA |

*Table 6. Management message handling*

## Inactivity Control

It is implemented.

## Message Sequence Control

SCCP PE maintains the order of messages between of upper and lower layer interfaces.

## Segmentation and Reassembly

It is a feature in service class 0 and 1. SCCP PE supports it.

## State Machine

SCCP PE maintains a state machine behavior for each connection-oriented services according to Figure 8/Q.711.

## Global Title Translation

Not supported.

## Capacity and Limitation

For Your convenience here is a short summary about limitations explained in previous chapters:

| **Feature** | **Restriction** | **Remark** |
| --- | --- | --- |
| Handling different length of signalling point codes thus addresses | ITU ANSI and TCC are implemented. | Specification dependent, See Table 2 |
| Management | Partially implemented (see Table 6) | Not supported by TCC |
| Service class 1 | Supported |  |
| Service class 2 | Supported | Not supported by TTC |
| Service class 3 | NOT IMPLEMENTED | Not supported by TTC |
| Routing | NOT IMPLEMENTED | SCCP PE is a signalling endpoint |
| Message sequence control | NOT IMPLEMENTED | Indifferent |
| Flow control | NOT IMPLEMENTED | Because class 3 not supported |
| Reassembly | NOT IMPLEMENTED |  |
| LUDT, LUDTS transfer | NOT IMPLEMENTED | Because ATM carrier not considered |
| Global title translation | NOT IMPLEMENTED |  |
| MTP3-PAUSE, MTP3-RESUME, MTP3-STATUS sending and processing after receiving | NOT IMPLEMENTED |  |

*Table 7. Features with restriction in SCCP PE*

There shall be exactly one SCCP User test component instance for each SCCP PE instance. An SCCP PE instance is able to handle up to 16 SCCP connections and 16 segmented N-UNITDATA messages at the same time.

## Differences between ITU, ANSI, MPT and TTC

1. Address length (see Table 2).
2. TTC doesn’t support connection-oriented services and management functionality.
3. TTC doesn’t support management functionality.
4. ANSI has different Address Indicator structure (Order of PC and SSN is changed, see Figure 4/Q.713 (07/96) and Figure 4/T1.112.3)
5. ANSI has different Address Elements structure (Ordering of PC and SSN is changed, see Figure 5/Q.713 (07/96) and Figure 4A/T1.112.3)
6. ANSI has different gti0001 structure (see Figure 7/Q.713 (07/96) and Figure 6/T1.112.3)
7. ANSI doesn’t support gti0011 and gti0100.   
   More exactly ANSI gti0001 = ITU gti0011
8. ANSI doesn’t support optional field “importance”.
9. TTC doesn’t support LUDT, LUDTS

# Test Port Usage

The SCCP PE is developed for testing implementations of SCCP Users using TTCN-3 and it uses the services of underlying level MTP3 (see in *Figure 2*). It is considered that the SCCP layer of the peer conforms to the same specifications as SCCP PE does.

##### SUT

##### Test Suite

SCCP-User

Instance 2

SCCP-User

Instance 1

N-Service Primitives

SCCP

Instance 2

###### SCCP PE Instance 1

Component SCCP\_CT

MTP3-Service Primitives

MTP3 or M3UA (and lower layers)

Instance 2

MTP3 or M3UA (and lower layers)

Instance 1

**System**

*Figure 2 Service primitives in SS7*

## The User Interface: the N-Service Primitives

SCCP PE communicates with its user by means of N-Service primitives.

These primitives are implemented as TTCN-3 records. Any SCCP User inserts its message in the field “User Data”. Their implementation can be found in file SCCPasp\_Types.ttcn.

### SCCP Primitives of Connectionless Service

SCCP PE can receive N\_UNITDATA\_req messages and can send N\_UNITDATA\_ind and N\_NOTICE\_ind in case of connectionless communication.Their implementation is the following ( for details see the file SCCPasp\_Types.ttcn itself):

type record N\_UNITDATA\_req

{

SCCP\_PAR\_Address calledAddress ,

SCCP\_PAR\_Address callingAddress ,

SCCP\_PAR\_Sequence\_Control sequenceControl optional ,

SCCP\_PAR\_Return\_Option returnOption optional ,

SCCP\_PAR\_UserData userData ,

SCCP\_PAR\_Importance importance optional

}

type record N\_UNITDATA\_ind

{

SCCP\_PAR\_Address calledAddress ,

SCCP\_PAR\_Address callingAddress ,

SCCP\_PAR\_Sequence\_Control sequenceControl optional ,

SCCP\_PAR\_Return\_Option returnOption optional ,

SCCP\_PAR\_UserData userData ,

SCCP\_PAR\_Importance importance optional

}

type record N\_NOTICE\_ind

{

SCCP\_PAR\_Address calledAddress ,

SCCP\_PAR\_Address callingAddress ,

SCCP\_PAR\_Reason\_For\_Return reasonForReturn ,

SCCP\_PAR\_UserData userData ,

SCCP\_PAR\_Importance importance optional

}

### SCCP Primitives of Connection Oriented Service

SCCP PE can receive N\_CONNECT\_req, N\_CONNECT\_res, N\_DATA\_req and N\_DISCONNECT\_req and send them as N\_CONNECT\_ind, N\_CONNECT\_cfm,N\_DATA\_ind and N\_DISCONNECT\_ind, respectively.

Their implementation is the following:

type record N\_CONNECT\_req

{

SCCP\_PAR\_Address calledAddress,

SCCP\_PAR\_Address callingAddress optional,

SCCP\_PAR\_Expedited\_Data\_Sel expeditedDataSel optional,

SCCP\_PAR\_Quality\_Of\_Service qualityOfService optional,

SCCP\_PAR\_UserData userData optional,

SCCP\_PAR\_Connection\_Id connectionId optional,

SCCP\_PAR\_Importance importance optional

}

type record N\_CONNECT\_ind

{

SCCP\_PAR\_Address calledAddress,

SCCP\_PAR\_Address callingAddress optional,

SCCP\_PAR\_Quality\_Of\_Service qualityOfService optional,

SCCP\_PAR\_UserData userData optional,

SCCP\_PAR\_Connection\_Id connectionId optional,

SCCP\_PAR\_Importance importance optional

}

type record N\_CONNECT\_res

{

SCCP\_PAR\_Address respondingAddress optional,

SCCP\_PAR\_Expedited\_Data\_Sel expeditedDataSel optional,

SCCP\_PAR\_Quality\_Of\_Service qualityOfService optional,

SCCP\_PAR\_UserData userData optional,

SCCP\_PAR\_Connection\_Id connectionId optional,

SCCP\_PAR\_Importance importance optional

}

type record N\_CONNECT\_cfm

{

SCCP\_PAR\_Address respondingAddress optional,

SCCP\_PAR\_Quality\_Of\_Service qualityOfService optional,

SCCP\_PAR\_UserData userData optional,

SCCP\_PAR\_Connection\_Id connectionId optional,

SCCP\_PAR\_Importance importance optional

}

type record N\_DATA\_req

{

SCCP\_PAR\_UserData userData ,

SCCP\_PAR\_Connection\_Id connectionId optional ,

SCCP\_PAR\_Importance importance optional

}

type record N\_DATA\_ind

{

SCCP\_PAR\_UserData userData ,

SCCP\_PAR\_Connection\_Id connectionId optional ,

SCCP\_PAR\_Importance importance optional

}

type record N\_DISCONNECT\_req

{

SCCP\_PAR\_Address respondingAddress optional,

SCCP\_PAR\_Reason reason ,

SCCP\_PAR\_UserData userData optional ,

SCCP\_PAR\_Connection\_Id connectionId optional ,

SCCP\_PAR\_Importance importance optional

}

type record N\_DISCONNECT\_ind

{

SCCP\_PAR\_Originator originator ,

SCCP\_PAR\_Address respondingAddress optional ,

SCCP\_PAR\_Reason reason ,

SCCP\_PAR\_UserData userData optional ,

SCCP\_PAR\_Connection\_Id connectionId optional ,

SCCP\_PAR\_Importance importance optional

}

## The MTP3-Service Primitives

For implementation details see file MTP3asp\_Types.ttcn of product CNL 113 337.Here only the implementation of the two MTP-TRANSFER primitives listed:

type record MTP3\_Field\_sio

{

bitstring ni length(2),

bitstring prio length(2),

bitstring si length(4)

}

type record ASP\_MTP3\_TRANSFERind

{

MTP3\_Field\_sio sio,

integer opc,

integer dpc,

integer sls,

octetstring data

}

type record ASP\_MTP3\_TRANSFERreq

{

MTP3\_Field\_sio sio,

integer opc,

integer dpc,

integer sls,

octetstring data

}

### Choosing between protocol standards resp. versions

The service type or “flavor” of the SCCP PE defines which specification should be followed.

These types are: MTP3 ITU, MTP3b ITU, MTP3 ANSI, MTP3 MPT, MTP3 TCC (Japanese)

Remark:  
M3UA is not an option. M3UA is not a standalone service type because if M3UA serves on level 3 instead of MTP3 it can receive and send messages of any length according to any required upper specification mentioned above.

### Forced sending of XUDT messages

SCCP transfers the received information from N\_UNITDATA\_req in udt messages or in xudt messages (if the User data is long). The mapping into xudt can be forced.

## Installation

Since the SCCP PE is used as a part of the TTCN-3 test environment this requires TTCN-3 Test Executor to be installed before any operation of the SCCP PE. For more details on the installation of TTCN-3 Test Executor see the relevant section of [2].

An implementation of the MTP3 protocol or an MTP3 test port is also assumed.

### Description of files implementing the SCCP PE

The SCCP PE consists of the following files:

SCCPasp\_Types.ttcn  
SCCP\_Mapping.ttcnpp  
SCCP\_Types.ttcn  
SCCP\_Emulation.ttcn

Their functionality is the following:

SCCPasp\_Types.ttcn

This file contains the interface between the SCCP PE and the SCCP User. It contains the abstract service primitives implemented as TTCN-3 messages, templates and it contains the port definitions between the SCCP User and SCCP.

SCCP\_Mapping.ttcnpp

This file contains the dual face port definition for the lower port including the encoding-decoding functions used in the dual face port.

SCCP\_Types.ttcn

This file contains all other definitions used in SCCP PE. It contains definitions of types, templates, ports.

SCCP\_Emulation.ttcn

This file contains PDU templates and the dynamical part.

## Configuration

The SCCP protocol behaviour can be influenced in two way. The first one is to set module parameters in the configuration file. The second one is to set the arguments of the function “SCCPStart”

### SCCP PE parameters in the Protocol Emulation configuration file

Some properties of the SCCP PE can be set in the [MODULE\_PARAMETERS] section of the configuration file. These are the following:

SCCP.tsp\_maxLocalReference:

-type: float

-meaning: Max value of the field Local Reference .Local Reference = 0 …(SCCP.tsp\_maxLocalReference-1). Local Reference For details see 3.3/[9]

-possible values: 0-16777216

-default value: 16777216.0

-OPTIONAL

SCCP.tsp\_maxConnectionId:

-type:float

-meaning: max value of ASP field Connection Identification. More exactly Connection Identification = 0… (SCCP.tsp\_maxConnectionId –1). For details see [7].

-possible values: 0-16777216

-default value: 16777216.0

-OPTIONAL

SCCP.tsp\_force\_xudt:

-type: integer

-meaning: If it is 1, the N\_UNITDATA\_req always will be mapped into xudt, regardless of the size of the ASP (forced mapping).

-possible values: 0 –forcing is off (NO)

1 –forcing is on (YES)

-default value: 0

-OPTIONAL

SCCP.tsp\_SIF\_MaxLength :

-type: integer

-meaning: The maximum size of SIFin bytes

-possible values: 8..1532

-default value: 272 (MTP3)

-OPTIONAL

### Arguments of function SCCPStart

SCCPStart is the function containing the behaviour of the SCCP test component. It should receive some initial parameters in the argument pl\_Boot with type MSC\_SCCP\_MTP3\_parameters. This way is introduced to give the possibility to apply more than one SS7 protocol stack in the same test suite.

The type definition:

type record MSC\_SCCP\_MTP3\_parameters

{

MTP3\_Field\_sio sio,

integer opc,

integer dpc,

integer sls,

SCCP\_ServiceType sccp\_serviceType,

integer ssn optional

}

Definition of the fields:

sio: Service information octet, see 14.2/Q.704.

opc: SPC of the node containing the SCCP (“this node”)

dpc: SPC of the peer node (SPC of the SUT). If it is set to 0, then it is not included into the messages sent from TTCN to SUT.

sls: Signalling Link Selection field of the routing label, see 2.2/Q.704.

sccp\_serviceType: It defines which specification should be follow. Its possible values are:

-“mtp3\_itu”

-“mtp3b\_itu”

-“mtp3\_ansi”

-“mtp3b\_ansi”

-“mtp3\_mpt”

-“mtp3\_ttc”

ssn: Subsystem Number. It identifies the SCCP User, see 3.4.2.2/[9]. If it is set, then the subsystem test message will be approved only for this subsystem. If it is omitted, then every subsystem test message will be approved.

## Makefile

If the lower (dual faced) port is connected to an MTP3 distributor component, then flag for TTCN-3 files should be set on the following way:

CPPFLAGS\_TTCN3 = -DNoMTPMsgDistribution

If this flag is set, then the lower port will be an external port otherwise it will be an internal port.

## Error messages

-

## Warning messages

-

## Examples

Two examples are attached here.

The first one demonstrates how to make a so called “self test” without real SUT.

The second one is a very simple test where the MTP3 level is implemented as a test port and the SUT is implemented by SEA.

### Example 1

This example demonstrates how to make a self test.

It consists of two “tower” i.e. quasi SS7 protocol stack - A and B.

MTC plays the role of SCCP User A and SCCP User 2. It sends a message across its port A and waits for answer in its port B.

The two MTP3 protocol replaced by an MTP3sim component with two ports, A and B. If a MTP3\_TRANSFERreq primitive is received in any of its ports, it will be “renamed” for MTP3\_TRANSFERind and will be send out on the other port.

The scheme is the following:

**MTC**

SCCPasp\_PT PORT\_A SCCPasp\_PT PORT\_B

SCCPasp\_SP\_PT SCCP\_SP\_PORT

**SCCP “A”**

MTP3asp\_SCCP\_PT MTP3SCCP\_PORT

SCCPasp\_SP\_PT SCCP\_SP\_PORT

**SCCP “B”**

MTP3asp\_SCCP\_PT MTP3SCCP\_PORT

MTP3asp\_SP\_PT MTP\_PORT\_A MTP3asp\_SP\_PT MTP\_PORT\_B

**MTP3sim**

*Figure 3 Scheme in Example 1*

#### Configuration file 1

[LOGGING]

#FileName := "SCCP\_selftest.cfg"

FileMask := LOG\_ALL | DEBUG | MATCHING\_TIMEOUT | MATCHING\_PROBLEM

#ConsoleMask := LOG\_ALL

#ConsoleMask := WARNING | ERROR | TESTCASE | STATISTICS | PORTEVENT

ConsoleMask := LOG\_ALL | DEBUG | MATCHING\_TIMEOUT | MATCHING\_PROBLEM

#LogFile := "My.log"

#TimeStampFormat := Time

LogSourceInfo := Yes

[EXECUTE]

SCCP\_selftest.tc\_ConnlessSendingShortASP

#SCCP\_selftest.tc\_ConnlessSendingLongASP

[TESTPORT\_PARAMETERS]

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// \* DO NOT FORGET TO SET THE FOLLOWING TWO LINE TO YOUR SEA \*

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

system.\*.Hostname := "balisea" //sea server name

system.\*.HttpPort := "5000" //sea http port

system.\*.IID\_String := "b303d76a-266c-11d4-b8f5-08002090d3da"

//Device Type ID

system.\*.Loop:= "ON"

system.\*.Filter:= "OFF"

system.\*.MTP3ServiceType := "MTP3ttc" //"MTP3itu" ["MTP3itu" (default)|"MTP3ansi" | "M3UA" | "MTP3ttc" ]

// CMGW6 -> SCTP\_ASSOC\_10.2.110.102

// CMGW3 data: SCTP\_ASSOC\_10.2.110.2

system.CMGW6.EntityName := "S7ST-0" //device name to connect

system.CMGW6.Sio:= "83’O" //or "H’83" SCCP

system.CMGW6.SUT\_Pc:= "461086" // 07-09-30 =0x07091E see command: s7stp:st=s7stg-0&&-32;

system.CMGW6.TESTER\_Pc:= "461087" //07-09-31=0x07091F

system.CMGW6.M3UA\_version:= "1"

[MODULE\_PARAMETERS]

//for sccp:

tsp\_own\_GT := '14377760'H

tsp\_remote\_GT := '14375760'H

tsp\_SSN := 2 //8:MSC 5:MAP see 3.4.2.2/Q.713

tsp\_SIO := '03'O //SCCP national

tsp\_own\_SPC := 2351 //16382

tsp\_remote\_SPC := 2300 //16383 // max value on 14 bits

#tsp\_own\_SPC := 461087 // =0x07091E

#tsp\_remote\_SPC := 461086 // =0x07091D

tsp\_SLS := 0

#tsp\_sccp\_serviceType := "mtp3\_itu"

#tsp\_sccp\_serviceType := "mtp3\_ansi"

tsp\_sccp\_serviceType := "mtp3\_ttc"

#for mtp3\_itu/gti0011 or mtp3\_ansi/gti0001 :

#tsp\_translationType := 7

tsp\_SIF\_MaxLength := 272

tsp\_force\_xudt := 1 // 1:yes, 0:no

#### Test Suite 1

// File: SCCP\_selftest.ttcn

// Description: SS7 SCCP basic test

// according to specification ITU-T SS7 SCCP, ANSI ..., TCC ...

// References: ITU-T: Recommendation Q.711-Q.714,

// ANSI ,

//

module SCCP\_selftest

{//startmodule

modulepar

{

hexstring tsp\_own\_GT := '0614377760'H;

hexstring tsp\_remote\_GT := '0614375760'H;

integer tsp\_SSN := 2; //8:MSC 5:MAP see 3.4.2.2/Q.713

octetstring tsp\_SIO := '83'O;//SCCP national

integer tsp\_own\_SPC := 461087; // =0x07091E

integer tsp\_remote\_SPC := 461086; // =0x07091D

integer tsp\_SLS := 0;

charstring tsp\_sccp\_serviceType := "mtp3\_itu"

}//modulepar

import from General\_Types all;

import from MTP3asp\_Types all;

import from MTP3asp\_PortType all;

import from SCCPasp\_Types all;

import from SCCP\_Types all;

import from SCCP\_Emulation all;

//==================================================================

// MTPsim component

// Description: Simulates two MTP stacks for two MTP3-User

// to implement this configuration:

// MTPsim includes MTP3/1 and MTP3/2

// MTPsim only receives TRANSFER\_req and sends TRANSFER\_ind

// with the same content

// +----------+ +----------+

// |SCCP-userA| <--->|SCCP-userB| = MTC

// +----------+ +----------+

// | A | B

// +----------+ +----------+

// | SCCP A | <--->| SCCP B |

// +----------+ +----------+

// | A | B

// +-----------------------------+

// | MTP3 1. | MTP3 2.| = MTP3sim

// +----------------------------+

//

//==================================================================

group MTPsim

{

type component MTPsim\_CT {

port MTP3asp\_SP\_PT MTP\_A\_PORT

port MTP3asp\_SP\_PT MTP\_B\_PORT

}

function MTPsim\_EventHandler( ) runs on MTPsim\_CT

{

var ASP\_MTP3\_TRANSFERreq vl\_MTP3\_TRANSFERreq;

var ASP\_MTP3\_TRANSFERind vl\_MTP3\_TRANSFERind

alt{

[] MTP\_A\_PORT.receive( ASP\_MTP3\_TRANSFERreq:? ) -> value vl\_MTP3\_TRANSFERreq

{

MTP\_B\_PORT.send( t\_ASP\_MTP3\_TRANSFERind(

vl\_MTP3\_TRANSFERreq.sio,

vl\_MTP3\_TRANSFERreq.opc,

vl\_MTP3\_TRANSFERreq.dpc,

vl\_MTP3\_TRANSFERreq.sls,

vl\_MTP3\_TRANSFERreq.data ) ) ;

repeat;

}//A.receive

[] MTP\_B\_PORT.receive( ASP\_MTP3\_TRANSFERreq:? ) -> value vl\_MTP3\_TRANSFERreq

{

MTP\_A\_PORT.send( t\_ASP\_MTP3\_TRANSFERind (

vl\_MTP3\_TRANSFERreq.sio,

vl\_MTP3\_TRANSFERreq.opc,

vl\_MTP3\_TRANSFERreq.dpc,

vl\_MTP3\_TRANSFERreq.sls,

vl\_MTP3\_TRANSFERreq.data ));

repeat;

}//B.receive

}//alt

} //MTPsim\_EventHandler

}//group MTPsim

// Main test component with behaviour SCCPuserA andSCCPuserB

type component MTC\_CT {

var SCCP\_PAR\_Address v\_CalledAddress, v\_CallingAddress;

var integer v\_testvalue;

var MTPsim\_CT vc\_MTPsim;

var SCCP\_CT vc\_SCCP\_A, vc\_SCCP\_B;

var MSC\_SCCP\_MTP3\_parameters v\_BootA;

var MSC\_SCCP\_MTP3\_parameters v\_BootB;

var SCCP\_PAR\_Connection\_Id v\_cid\_A, v\_cid\_B;

port SCCPasp\_PT A\_PORT; //SCCPuserA

port SCCPasp\_PT B\_PORT //SCCPuserB

}

function initBootParams() runs on MTC\_CT

{

v\_BootA :=

{ sio:=

{ ni:= substr(oct2bit(tsp\_SIO),0,2),

prio:= substr(oct2bit(tsp\_SIO),2,2),

si:= substr(oct2bit(tsp\_SIO),4,4)

},

opc:=tsp\_own\_SPC,

dpc:=tsp\_remote\_SPC,

sls:=tsp\_SLS,

sccp\_serviceType:=tsp\_sccp\_serviceType,

ssn:= tsp\_SSN

};

v\_BootB :=

{ sio:=

{ ni:= substr(oct2bit(tsp\_SIO),0,2),

prio:= substr(oct2bit(tsp\_SIO),2,2),

si:= substr(oct2bit(tsp\_SIO),4,4)

},

opc:=tsp\_remote\_SPC,

dpc:=tsp\_own\_SPC,

sls:=tsp\_SLS,

sccp\_serviceType:=tsp\_sccp\_serviceType,

ssn:= tsp\_SSN

};

return;

} //initBootParams

function init() runs on MTC\_CT

{

initBootParams();

log("v\_BootA:",v\_BootA);

log("v\_BootB: ",v\_BootB);

vc\_MTPsim:= MTPsim\_CT.create;

// Protocol Stack A creation & connections:

vc\_SCCP\_A:=SCCP\_CT.create;

connect(vc\_SCCP\_A:MTP3sccp\_PORT,vc\_MTPsim:MTP\_A\_PORT);

connect(self:A\_PORT,vc\_SCCP\_A:SCCP\_PORT);

// Protocol Stack B creation & connections:

vc\_SCCP\_B:=SCCP\_CT.create;

connect(vc\_SCCP\_B:MTP3sccp\_PORT,vc\_MTPsim:MTP\_B\_PORT);

connect(self:B\_PORT,vc\_SCCP\_B:SCCP\_PORT);

// Start stacks:

vc\_MTPsim.start( MTPsim\_EventHandler() );

vc\_SCCP\_A.start( SCCPStart( v\_BootA ) ); // Bootparameters !!! cont here!!!

vc\_SCCP\_B.start( SCCPStart(v\_BootB));

log( "init() is done");

}// init

function terminate( ) runs on MTC\_CT

{

log( "termitate() started");

/\*while( all component.running != true )

{

//waits

}\*/

all component.stop;

disconnect(vc\_SCCP\_A:MTP3sccp\_PORT,vc\_MTPsim:MTP\_A\_PORT);

disconnect(self:A\_PORT,vc\_SCCP\_A:SCCP\_PORT);

disconnect(vc\_SCCP\_B:MTP3sccp\_PORT,vc\_MTPsim:MTP\_B\_PORT);

disconnect(self:B\_PORT,vc\_SCCP\_B:SCCP\_PORT);

log(" all components stopped");

self.stop;

log( "termitate() finished");

} //terminate

// function getOddEven returns '0'

// if even number of dec digit can be found in GT see Q.713

function getOddEven( in hexstring pl\_GT) return bitstring

{

return int2bit( (lengthof(pl\_GT) mod 2) ,1);

}

function getOddEvenEnc( in hexstring pl\_GT) return bitstring

{

if( (lengthof(pl\_GT) mod 2) == 0 ) { return '0010'B;} //even

else { return '0001'B;} //odd

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//function setAddresses\_gti0001() runs on MTC\_CT

// Sets CalledAddress and CallingAddress as a gti001-type address

// according to the cfg file.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function setAddresses\_gti0001() runs on MTC\_CT

{

if( (tsp\_sccp\_serviceType == "mtp3\_itu") or

(tsp\_sccp\_serviceType == "mtp3b\_itu") or

(tsp\_sccp\_serviceType == "mtp3\_ttc") or

(tsp\_sccp\_serviceType == "mtp3b\_ttc")

) {

v\_CalledAddress :={

addressIndicator := {

pointCodeIndic := '1'B,

ssnIndicator := '1'B,

globalTitleIndic := '0001'B,

routingIndicator := '0'B

},//addressIndicator

signPointCode := SCCP\_SPC\_int2bit(tsp\_remote\_SPC, tsp\_sccp\_serviceType, tsp\_SIO), // see SCCP\_Emulation.ttcn

subsystemNumber := tsp\_SSN,

globalTitle := {

gti0001:= {

natureOfAddress := '0000011'B,

oddeven := getOddEven( tsp\_remote\_GT ),

globalTitleAddress := tsp\_remote\_GT

}

}//globalTitle

} // v\_CalledAddress

v\_CallingAddress :={

addressIndicator := {

pointCodeIndic := '1'B,

ssnIndicator := '1'B,

globalTitleIndic := '0001'B,

routingIndicator := '0'B

},//addressIndicator

signPointCode := SCCP\_SPC\_int2bit(tsp\_own\_SPC, tsp\_sccp\_serviceType, tsp\_SIO), // see SCCP\_Emulation.ttcn

subsystemNumber := tsp\_SSN,

globalTitle:= {

gti0001 := {

natureOfAddress := '0000011'B,

oddeven := getOddEven( tsp\_own\_GT ),

globalTitleAddress := tsp\_own\_GT

}

}//globalTitle

} // v\_CallingAddress

} else if(

(tsp\_sccp\_serviceType == "mtp3\_ansi") or

(tsp\_sccp\_serviceType == "mtp3b\_ansi") )

{

v\_CalledAddress :={

addressIndicator := {

pointCodeIndic := '1'B,

ssnIndicator := '1'B,

globalTitleIndic := '0001'B,

routingIndicator := '0'B

},//addressIndicator

signPointCode := SCCP\_SPC\_int2bit(tsp\_remote\_SPC, tsp\_sccp\_serviceType, tsp\_SIO), // see SCCP\_Emulation.ttcn

subsystemNumber := tsp\_SSN,

globalTitle := {

gti0011:= {

translationType := int2oct(7,1),

encodingScheme := getOddEvenEnc( tsp\_remote\_GT ),

numberingPlan := '0111'B, //ISDN/mobile numbering plan, see T1.112.3-2001/3.4.2.3.1

globalTitleAddress:= tsp\_remote\_GT

}

}//globalTitle

} // v\_CalledAddress

v\_CallingAddress :={

addressIndicator := {

pointCodeIndic := '1'B,

ssnIndicator := '1'B,

globalTitleIndic := '0001'B,

routingIndicator := '0'B

},//addressIndicator

signPointCode := SCCP\_SPC\_int2bit(tsp\_remote\_SPC, tsp\_sccp\_serviceType, tsp\_SIO), // see SCCP\_Emulation.ttcn

subsystemNumber := tsp\_SSN,

globalTitle := {

gti0011:= {

translationType := int2oct(7,1),

encodingScheme := getOddEvenEnc( tsp\_own\_GT ),

numberingPlan := '0111'B, //ISDN/mobile numbering plan, see T1.112.3-2001/3.4.2.3.1

globalTitleAddress:= tsp\_own\_GT

}

}//globalTitle

} // v\_CallingAddress

}//if

else

{

log( "wrong tsp\_sccp\_serviceType ->exit ");

setverdict( fail );

}

}//setAddresses\_gti001

function f\_SendAndReceive1N\_UNITDATA(in octetstring pl\_userdata) runs on MTC\_CT

{

var ASP\_SCCP\_N\_UNITDATA\_ind vl\_N\_UNITDATA\_ind;

timer TL\_timer:= 40.0;

TL\_timer.start;

log("A\_PORT.send follows");

log("Addresses:",v\_CalledAddress, v\_CallingAddress);

A\_PORT.send( t\_ASP\_N\_UNITDATA\_req( v\_CalledAddress,

v\_CallingAddress,

'00000001'B, //sequence control

'00000001'B, //return option

pl\_userdata,

omit ) );

log("A\_PORT.send executed");

alt {

[] B\_PORT.receive( tr\_ASP\_N\_UNITDATA\_ind ) -> value vl\_N\_UNITDATA\_ind

{

if( (vl\_N\_UNITDATA\_ind.calledAddress == v\_CalledAddress ) and

(vl\_N\_UNITDATA\_ind.callingAddress == v\_CallingAddress) and

(vl\_N\_UNITDATA\_ind.userData == pl\_userdata) )

{

log("Correct CalledAddress, CallingAddress and userData received, data are correct");

setverdict(pass);

}

else

{

log("Some data corrupted");

log("Original data:", v\_CalledAddress, v\_CallingAddress, pl\_userdata);

setverdict( fail );

}

};

[] TL\_timer.timeout

{

setverdict( fail );

log("Timeout....");

};

} //alt

TL\_timer.stop;

}//f\_SendAndReceive1N\_UNITDATA

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Connection Oriented Part

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function f\_connect

Establishes a connection

(Sends an ASP\_SCCP\_N\_CONNECT\_req on A\_PORT and waits for

N\_CONNECT\_ind on B\_PORT. If it is received,

it sends back an ASP\_SCCP\_N\_CONNECT\_res on B\_PORT and waits for

N\_CONNECT\_cfm on A\_PORT)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function f\_connect() runs on MTC\_CT return boolean

{

var ASP\_SCCP\_N\_CONNECT\_ind vl\_N\_CONNECT\_ind;

var ASP\_SCCP\_N\_CONNECT\_cfm vl\_N\_CONNECT\_cfm;

setverdict(none);

v\_cid\_A := 13;

timer TL\_timer:= 40.0;

TL\_timer.start;

// A Sends ASP\_SCCP\_N\_CONNECT\_req , receives

A\_PORT.send( t\_ASP\_N\_CONNECT\_req( v\_CalledAddress,

v\_CallingAddress,

omit, //expeditedDataSel

omit, //QoS

omit, //userData

v\_cid\_A,

omit //importance

) );

alt {

[] B\_PORT.receive( tr\_ASP\_N\_CONNECT\_ind ) -> value vl\_N\_CONNECT\_ind

{

v\_cid\_B := vl\_N\_CONNECT\_ind.connectionId;

B\_PORT.send( t\_ASP\_N\_CONNECT\_res( omit,// respondingAddress

omit,//expeditedDataSel

omit,//qualityOfService

omit, //userData

v\_cid\_B,

omit //importance

));

}

[] B\_PORT.receive

{

log( "unexpected asp received for ASP\_SCCP\_N\_CONNECT\_req, failed");

setverdict( fail );

return false;

}

[] TL\_timer.timeout

{

setverdict( pass );

log("Timeout....");

return false;

}

}

// receives ASP\_SCCP\_N\_CONNECT\_cfm

alt {

[] A\_PORT.receive( tr\_ASP\_N\_CONNECT\_cfm ) -> value vl\_N\_CONNECT\_cfm

{

setverdict( pass );

log("f\_connect finished successfully");

return true;

}

[] TL\_timer.timeout

{

setverdict( pass );

log("Timeout....");

return false;

}

}// alt

log("f\_connect finished");

return false;

}//f\_connect

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function f\_send

Sends an ASP\_SCCP\_N\_DATA\_req on A\_PORT and waits for answer in

B\_PORT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function f\_send(in octetstring pl\_userdata) runs on MTC\_CT

{

var ASP\_SCCP\_N\_DATA\_ind vl\_N\_DATA\_ind;

timer TL\_timer:= 120.0;

TL\_timer.start;

A\_PORT.send( t\_ASP\_N\_DATA\_req ( pl\_userdata, v\_cid\_A, omit) ) ;

alt {

[] B\_PORT.receive( tr\_ASP\_N\_DATA\_ind ) -> value vl\_N\_DATA\_ind

{

if( vl\_N\_DATA\_ind.userData == pl\_userdata )

{

log( "userData received correctly" );

setverdict( pass );

}

else

{

log("user data mismatch error in f\_send()")

setverdict(fail);

}

}//B\_PORT.receive( tr\_ASP\_N\_DATA\_ind )

[] B\_PORT.receive

{

log( "unexpected asp received for ASP\_SCCP\_N\_DATA\_req, failed");

setverdict( fail );

}

[] TL\_timer.timeout

{

setverdict( pass );

log("Timeout....");

}

} //alt

}//f\_send

//f\_disconnect with timeout

function f\_disconnect( ) runs on MTC\_CT

{

var ASP\_SCCP\_N\_DISCONNECT\_ind vl\_N\_DISCONNECT\_ind;

timer TL\_timer:= 5.0;

TL\_timer.start;

A\_PORT.send(t\_ASP\_N\_DISCONNECT\_req( omit, // respondingAddress

0, //reason : end user originated, see 3.11/Q.713

omit, //userData

v\_cid\_A,

omit ))

alt {

[] B\_PORT.receive(tr\_ASP\_N\_DISCONNECT\_ind) -> value vl\_N\_DISCONNECT\_ind

{

setverdict( pass );

}

[] B\_PORT.receive

{

log("unexpected asp received on B\_PORT instead of ASP\_SCCP\_N\_DISCONNECT\_ind");

//repeat;

setverdict(fail);

}

[] TL\_timer.timeout

{

setverdict( fail );

log("Timeout....");

};

}//alt

//give time for inner release complete (rlc):

alt {

[] TL\_timer.timeout

{

setverdict( pass );

log("Stopped with expected timeout");

};

}

}//f\_disconnect

//===================================================

// Testcases

//===================================================

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

tc\_ConnlessSendingShortASP

Sends a 300 octet long userdata in one ASP\_SCCP\_N\_UNITDATA\_req

and receives it in one ASP\_SCCP\_N\_UNITDATA\_req.

SCCP transfers information

in udp or (forced) xudp packets.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

testcase tc\_ConnlessSendingShortASP() runs on MTC\_CT

{

var octetstring vl\_userdata;

init();

setAddresses\_gti0001();

vl\_userdata :='12345678901234567890'O;

f\_SendAndReceive1N\_UNITDATA( vl\_userdata );

terminate();

} //tc\_ConnlessSendingShortASP

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

tc\_ConnlessSendingLongASP

Sends a 300 octet long userdata in one ASP\_SCCP\_N\_UNITDATA\_req

and receives it in one ASP\_SCCP\_N\_UNITDATA\_req.

It is used for segmentation and reassembly.

SCCP transfers information

in xudp packets

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

testcase tc\_ConnlessSendingLongASP() runs on MTC\_CT

{

var octetstring vl\_userdata;

var integer vl\_i;

init();

setAddresses\_gti0001();

vl\_userdata := ''O;

for(vl\_i:=0;vl\_i<30;vl\_i:=vl\_i+1) {

vl\_userdata := vl\_userdata &'12345678901234567890'O;

}

f\_SendAndReceive1N\_UNITDATA( vl\_userdata );

terminate();

}//tc\_ConnlessSendingLongASP

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

tc\_ConnOrientedShortASPSending

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

testcase tc\_ConnOrientedShortASPSending() runs on MTC\_CT

{

var octetstring vl\_userdata;

init();

setAddresses\_gti0001();

vl\_userdata := '12345678901234567890'O;

f\_connect( );

f\_send(vl\_userdata);

f\_disconnect();

terminate();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

tc\_ConnOrientedLongASPSending

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

testcase tc\_ConnOrientedLongASPSending() runs on MTC\_CT

{

var octetstring vl\_userdata;

var integer vl\_i;

init();

setAddresses\_gti0001();

vl\_userdata := ''O;

for(vl\_i:=0;vl\_i<30;vl\_i:=vl\_i+1) {

vl\_userdata := vl\_userdata &'12345678901234567890'O;

}

f\_connect( );

f\_send(vl\_userdata);

//f\_SendAndReceive1N\_UNITDATA( vl\_userdata );

f\_disconnect();

terminate();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CONTROL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

control

{

execute( tc\_ConnlessSendingShortASP() );

execute( tc\_ConnlessSendingLongASP() );

execute( tc\_ConnOrientedShortASPSending());

execute( tc\_ConnOrientedLongASPSending());

}

}//module

### Example 2

Example 2 implements a real test situation. The TTCN-3 test suite interacts with a SEA which contains a simulated SS7 signalling node. The test suite sends a message and waits for answer until timeout.

**SUT (SEA)**

**SCCP User**

SCCPasp\_PT SCCP\_PORT

SCCPasp\_SP\_PT SCCP\_SP\_PORT

**SCCP**

MTP3asp\_SCCP\_PT MTP3SCCP\_PORT

MTP3asp\_SP\_PT MTP3\_SP\_PORT

**SYSTEM (MTP3)**

Figure 4 Scheme in Example 2

#### Configuration file 2

[LOGGING]

#FileName := "SCCP\_Testcases.cfg"

FileMask := LOG\_ALL | DEBUG | MATCHING\_TIMEOUT | MATCHING\_PROBLEM

#ConsoleMask := LOG\_ALL

#ConsoleMask := TESTCASE | PORTEVENT | DEBUG | MATCHING\_TIMEOUT | MATCHING\_PROBLEM

#ConsoleMask := WARNING | ERROR | TESTCASE | STATISTICS | PORTEVENT

#ConsoleMask := LOG\_ALL | DEBUG | MATCHING\_TIMEOUT | MATCHING\_PROBLEM

LogSourceInfo := Yes

[EXECUTE]

#SCCP\_selftest.tc\_ConnlessSendingShortASP

#SCCP\_selftest.tc\_ConnlessSendingLongASP

#SCCP\_selftest.tc\_ConnOrientedShortASPSending

#SCCP\_Testcases.tc\_ConnlessSendingLongASP

SCCP\_Testcases.tc\_ConnOrientedShortASPSending

[TESTPORT\_PARAMETERS]

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// \* DO NOT FORGET TO SET THE FOLLOWING TWO LINE TO YOUR SEA \*

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

system.\*.Hostname := "karasea" //sea server name

system.\*.HttpPort := "5001" //sea http port

system.\*.IID\_String := "b303d76a-266c-11d4-b8f5-08002090d3da"

//Device Type ID

system.\*.Loop:= "OFF"

system.\*.Filter:= "OFF"

system.\*.MTP3ServiceType := "MTP3ttc" // ["MTP3itu" (default)|"MTP3ansi" | "M3UA" |"MTP3tcc]

// CMGW6 -> SCTP\_ASSOC\_10.2.110.102

// CMGW3 data: SCTP\_ASSOC\_10.2.110.2

system.CMGW6.EntityName := "SAALH-0" //"S7ST-0" //device name to connect

system.CMGW6.Sio := "83’O" //or "H'83" =SCCP

system.CMGW6.SUT\_Pc := "2300"

system.CMGW6.TESTER\_Pc := "2351"

system.CMGW6.M3UA\_version:= "1"

[MODULE\_PARAMETERS]

tsp\_own\_GT := '14377760'H

tsp\_remote\_GT := '14375760'H

tsp\_SSN := 8 //8:MSC 5:MAP see 3.4.2.2/Q.713

tsp\_SIO := '83'O //SCCP national

tsp\_own\_SPC := 2351

tsp\_remote\_SPC := 2300// max value on 14 bits

tsp\_SLS := 0

#[tsp\_sccp\_serviceType := "mtp3\_itu" |"mtp3b\_itu"|"mtp3\_ansi"|"mtp3b\_ansi"| "mtp3b\_tcc"]

tsp\_sccp\_serviceType := "mtp3\_ttc"

#for mtp3\_itu/gti0011 or mtp3\_ansi/gti0001 :

#tsp\_translationType := 7

tsp\_SIF\_MaxLength := 272

tsp\_force\_xudt := 0 // 1:yes, 0:no

#### Test Suite 2

// File: SCCP\_TestCases.ttcn

module SCCP\_Testcases

{//startmodule

modulepar

{

hexstring tsp\_own\_GT := '0614377760'H;

hexstring tsp\_remote\_GT := '0614375760'H;

integer tsp\_SSN := 2; //8:MSC 5:MAP see 3.4.2.2/Q.713

octetstring tsp\_SIO := '83'O;//SCCP national

integer tsp\_own\_SPC := 461087; // =0x07091E

integer tsp\_remote\_SPC := 461086; // =0x07091D

integer tsp\_SLS := 0;

charstring tsp\_sccp\_serviceType := "mtp3\_itu";

octetstring MTP3\_UserPart\_SIO;

integer MTP3\_UserPart\_OPC,

MTP3\_UserPart\_DPC,

MTP3\_UserPart\_SLS

}//modulepar

import from General\_Types all;

import from MTP3asp\_Types all;

import from MTP3asp\_PortType all;

import from SCCPasp\_Types all;

import from SCCP\_Types all;

import from SCCP\_Emulation all;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Components

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Model of MSC:

type component MSC\_ST {

port MTP3asp\_SP\_PT CMGW6;

//port MTP3asp\_PT CMGW6;

};

// Main test component with behaviour SCCPuserA andSCCPuserB

type component MTC\_CT {

var SCCP\_PAR\_Address v\_CalledAddress, v\_CallingAddress;

var integer v\_testvalue;

var SCCP\_CT vc\_SCCP\_A ;

var MSC\_SCCP\_MTP3\_parameters v\_BootA;

var SCCP\_PAR\_Connection\_Id v\_cid\_A ;

port SCCPasp\_PT A\_PORT; //SCCPuserA

}

function initBootParams() runs on MTC\_CT

{

v\_BootA :=

{ sio:=

{ ni:= substr(oct2bit(tsp\_SIO),0,2),

prio:= substr(oct2bit(tsp\_SIO),2,2),

si:= substr(oct2bit(tsp\_SIO),4,4)

},

opc:=tsp\_own\_SPC,

dpc:=tsp\_remote\_SPC,

sls:=tsp\_SLS,

sccp\_serviceType:=tsp\_sccp\_serviceType,

ssn:= tsp\_SSN

};

return;

} //initBootParams

function init() runs on MTC\_CT //system MSC\_ST

{

initBootParams();

log("v\_BootA:",v\_BootA);

// Protocol Stack A creation & connections:

vc\_SCCP\_A:=SCCP\_CT.create;

map(vc\_SCCP\_A:MTP3sccp\_PORT:MTP3user\_sccp\_PORT,system:CMGW6);

connect(self:A\_PORT,vc\_SCCP\_A:SCCP\_PORT);

vc\_SCCP\_A.start( SCCPStart( v\_BootA ) ); // Bootparameters

log( "init() is done");

}// init

function terminate( ) runs on MTC\_CT //system MSC\_ST

{

log( "termitate() started");

/\*while( all component.running != true )

{

//waits

}\*/

all component.stop;

unmap(vc\_SCCP\_A:MTP3sccp\_PORT:MTP3user\_sccp\_PORT,system:CMGW6);

disconnect(self:A\_PORT,vc\_SCCP\_A:SCCP\_PORT);

log(" all components stopped");

self.stop;

log( "termitate() finished");

} //terminate

// function getOddEven returns '0'

// if even number of dec digit can be found in GT see Q.713

function getOddEven( in hexstring pl\_GT) return bitstring

{

return int2bit( (lengthof(pl\_GT) mod 2) ,1);

}

function getOddEven\_ansi( in hexstring pl\_GT) return bitstring

{

if( (lengthof(pl\_GT) mod 2) == 0 ) { return '0010'B;} //even

else { return '0001'B;} //odd

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//function setAddresses\_gti0001() runs on MTC\_CT

// Sets CalledAddress and CallingAddress as a gti001-type address

// according to the cfg file.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function setAddresses\_gti0001() runs on MTC\_CT

{

if( (tsp\_sccp\_serviceType == "mtp3\_itu") or

(tsp\_sccp\_serviceType == "mtp3b\_itu") or

(tsp\_sccp\_serviceType == "mtp3\_ttc") or

(tsp\_sccp\_serviceType == "mtp3b\_ttc")

) {

v\_CalledAddress :={

addressIndicator := {

pointCodeIndic := '1'B,

ssnIndicator := '1'B,

globalTitleIndic := '0001'B,

routingIndicator := '0'B

},//addressIndicator

signPointCode := SCCP\_SPC\_int2bit(tsp\_remote\_SPC, tsp\_sccp\_serviceType, tsp\_SIO), // see SCCP.ttcn

subsystemNumber := tsp\_SSN,

globalTitle := {

gti0001:= {

natureOfAddress := '0000011'B,

oddeven := getOddEven( tsp\_remote\_GT ),

globalTitleAddress := tsp\_remote\_GT

}

}//globalTitle

} // v\_CalledAddress

v\_CallingAddress :={

addressIndicator := {

pointCodeIndic := '1'B,

ssnIndicator := '1'B,

globalTitleIndic := '0001'B,

routingIndicator := '0'B

},//addressIndicator

signPointCode := SCCP\_SPC\_int2bit(tsp\_own\_SPC, tsp\_sccp\_serviceType, tsp\_SIO), // see SCCP.ttcn

subsystemNumber := tsp\_SSN,

globalTitle:= {

gti0001 := {

natureOfAddress := '0000011'B,

oddeven := getOddEven( tsp\_own\_GT ),

globalTitleAddress := tsp\_own\_GT

}

}//globalTitle

} // v\_CallingAddress

} else if(

(tsp\_sccp\_serviceType == "mtp3\_ansi") or

(tsp\_sccp\_serviceType == "mtp3b\_ansi")

) {

v\_CalledAddress :={

addressIndicator := {

pointCodeIndic := '1'B,

ssnIndicator := '1'B,

globalTitleIndic := '0001'B,

routingIndicator := '0'B

},//addressIndicator

signPointCode := SCCP\_SPC\_int2bit(tsp\_remote\_SPC, tsp\_sccp\_serviceType, tsp\_SIO), // see SCCP.ttcn

subsystemNumber := tsp\_SSN,

globalTitle := {

gti0011:= {

translationType := int2oct(7,1),

encodingScheme := getOddEven\_ansi( tsp\_remote\_GT ),

numberingPlan := '0111'B, //ISDN/mobile numbering plan, see T1.112.3-2001/3.4.2.3.1

globalTitleAddress:= tsp\_remote\_GT

}

}//globalTitle

} // v\_CalledAddress

v\_CallingAddress :={

addressIndicator := {

pointCodeIndic := '1'B,

ssnIndicator := '1'B,

globalTitleIndic := '0001'B,

routingIndicator := '0'B

},//addressIndicator

signPointCode := SCCP\_SPC\_int2bit(tsp\_remote\_SPC, tsp\_sccp\_serviceType, tsp\_SIO), // see SCCP.ttcn

subsystemNumber := tsp\_SSN,

globalTitle := {

gti0011:= {

translationType := int2oct(7,1),

encodingScheme := getOddEven\_ansi( tsp\_own\_GT ),

numberingPlan := '0111'B, //ISDN/mobile numbering plan, see T1.112.3-2001/3.4.2.3.1

globalTitleAddress:= tsp\_own\_GT

}

}//globalTitle

} // v\_CallingAddress

}//if

}//setAddresses\_gti001

function f\_SendAndReceive1N\_UNITDATA(in octetstring pl\_userdata) runs on MTC\_CT

{

var ASP\_SCCP\_N\_UNITDATA\_ind vl\_N\_UNITDATA\_ind;

timer TL\_timer:= 120.0;

TL\_timer.start;

A\_PORT.send( t\_ASP\_N\_UNITDATA\_req( v\_CalledAddress,

v\_CallingAddress,

'00000001'B, //sequence control

'00000001'B, //return option

pl\_userdata,

omit ) );

alt {

[] A\_PORT.receive( tr\_ASP\_N\_UNITDATA\_ind ) -> value vl\_N\_UNITDATA\_ind

{

if( (vl\_N\_UNITDATA\_ind.calledAddress == v\_CalledAddress ) and

(vl\_N\_UNITDATA\_ind.callingAddress == v\_CallingAddress) and

(vl\_N\_UNITDATA\_ind.userData == pl\_userdata) )

{

log("Correct CalledAddress, CallingAddress and userData received, data are correct");

setverdict(pass);

}

else

{

log("Some data corrupted");

setverdict( fail );

}

};

[] TL\_timer.timeout

{

setverdict( fail );

log("Timeout....");

};

} //alt

TL\_timer.stop;

}//f\_SendAndReceive1N\_UNITDATA

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Connection Oriented Part

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function f\_connect

Establishes a connection

(Sends an ASP\_SCCP\_N\_CONNECT\_req on A\_PORT and waits for

N\_CONNECT\_ind on B\_PORT. If it is received,

it sends back an ASP\_SCCP\_N\_CONNECT\_res on B\_PORT and waits for

N\_CONNECT\_cfm on A\_PORT)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function f\_connect() runs on MTC\_CT return boolean

{

var ASP\_SCCP\_N\_CONNECT\_ind vl\_N\_CONNECT\_ind;

var ASP\_SCCP\_N\_CONNECT\_cfm vl\_N\_CONNECT\_cfm;

setverdict(none);

v\_cid\_A := 13;

timer TL\_timer:= 120.0;

TL\_timer.start;

A\_PORT.send( t\_ASP\_N\_CONNECT\_req( v\_CalledAddress,

v\_CallingAddress,

omit, //expeditedDataSel

omit, //QoS

omit, //userData

v\_cid\_A,

omit //importance

) );

alt {

[] A\_PORT.receive( tr\_ASP\_N\_CONNECT\_cfm ) -> value vl\_N\_CONNECT\_cfm

{

setverdict( pass );

log("f\_connect finished successfully");

return true;

}

[] TL\_timer.timeout

{

setverdict( fail );

log("Timeout....");

return false;

}

}// alt

log("f\_connect finished");

return false;

}//f\_connect

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function f\_send

Sends an ASP\_SCCP\_N\_DATA\_req on A\_PORT and waits for answer in

A\_PORT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function f\_send(in octetstring pl\_userdata) runs on MTC\_CT

{

timer TL\_timer:= 120.0;

TL\_timer.start;

A\_PORT.send( t\_ASP\_N\_DATA\_req ( pl\_userdata, v\_cid\_A, omit) ) ;

alt {

[] A\_PORT.receive

{

setverdict( pass );

log("f\_connect finished successfully");

}

[] TL\_timer.timeout

{

setverdict( fail );

log("Timeout....");

}

} //alt

}//f\_send

//f\_disconnect with timeout

function f\_disconnect( ) runs on MTC\_CT

{

var ASP\_SCCP\_N\_DISCONNECT\_ind vl\_N\_DISCONNECT\_ind;

timer TL\_timer:= 25.0;

TL\_timer.start;

A\_PORT.send( t\_ASP\_N\_DISCONNECT\_req( omit, // respondingAddress

0, //reason : end user originated, see 3.11/Q.713

omit, //userData

v\_cid\_A,

omit ))

alt {

[] A\_PORT.receive

{

repeat;

setverdict(pass);

}

[] TL\_timer.timeout

{

setverdict( pass );

log("Timeout....");

};

}//alt

}//f\_disconnect

//===================================================

// Testcases

//===================================================

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

tc\_ConnlessSendingShortASP

Sends a 300 octet long userdata in one ASP\_SCCP\_N\_UNITDATA\_req

and receives it in one ASP\_SCCP\_N\_UNITDATA\_req.

SCCP transfers information

in udp or (forced) xudp packets.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

testcase tc\_ConnlessSendingShortASP() runs on MTC\_CT

{

var octetstring vl\_userdata;

init();

setAddresses\_gti0001();

vl\_userdata :='12345678901234567890'O;

f\_SendAndReceive1N\_UNITDATA( vl\_userdata );

terminate();

} //tc\_ConnlessSendingShortASP

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

tc\_ConnlessSendingLongASP

Sends a 300 octet long userdata in one ASP\_SCCP\_N\_UNITDATA\_req

and receives it in one ASP\_SCCP\_N\_UNITDATA\_req.

It is used for segmentation and reassembly.

SCCP transfers information

in xudp packets

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

testcase tc\_ConnlessSendingLongASP() runs on MTC\_CT system MSC\_ST

{

var octetstring vl\_userdata;

var integer vl\_i;

init();

setAddresses\_gti0001();

vl\_userdata := ''O;

for(vl\_i:=0;vl\_i<30;vl\_i:=vl\_i+1) {

vl\_userdata := vl\_userdata &'12345678901234567890'O;

}

f\_SendAndReceive1N\_UNITDATA( vl\_userdata );

terminate();

}//tc\_ConnlessSendingLongASP

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

tc\_ConnOrientedShortASPSending

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

testcase tc\_ConnOrientedShortASPSending() runs on MTC\_CT system MSC\_ST

{

var octetstring vl\_userdata;

init();

setAddresses\_gti0001();

vl\_userdata := '12345678901234567890'O;

f\_connect( );

f\_send(vl\_userdata);

f\_disconnect();

terminate();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

tc\_ConnOrientedLongASPSending

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

testcase tc\_ConnOrientedLongASPSending() runs on MTC\_CT

{

var octetstring vl\_userdata;

var integer vl\_i;

init();

setAddresses\_gti0001();

vl\_userdata := ''O;

for(vl\_i:=0;vl\_i<30;vl\_i:=vl\_i+1) {

vl\_userdata := vl\_userdata &'12345678901234567890'O;

}

f\_connect( );

f\_send(vl\_userdata);

//f\_SendAndReceive1N\_UNITDATA( vl\_userdata );

f\_disconnect();

terminate();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CONTROL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

control

{

execute( tc\_ConnlessSendingShortASP() );

execute( tc\_ConnlessSendingLongASP() );

execute( tc\_ConnOrientedShortASPSending());

execute( tc\_ConnOrientedLongASPSending());

}

}//module

# Terminology

Protocol Emulation: An instance which implements messages and dynamic behaviour of a given protocol layer.

SCCP Protocol Emulation: Implementation of SCCP as specified in [6].

SCCP User: Protocol, which uses services of SCCP.

## Abbreviations

ANSI American National Standards Institute

ASP Abstract Service Primitive

DPC Destination Point Code

ES ETSI Standard

ETSI European Telecommunications Standards Institute

IETF Internet Engineering Task Force

ITU International Telecommunication Union

ITU-T Telecommunication Standardization Sector of ITU

IUT Implementation Under Test

MPT Ministry of Post and Telecommunication (China)

MTP3 Message Transfer Part Level 3

M3UA MTP3 User Adaptation Layer

NI Network Indicator

OPC Originating Point Code

PC Point Code

PDU Protocol Data Unit

PE Protocol Emulation

SAP Service Access Point

SCCP Signalling Connection Control Part

SCCP PE SCCP Protocol Emulation

SS7 Signalling System No 7

TTCN-3 Testing and Test Control Notation version 3.

TTC Telecommunications Technology Committee (Standardization body of Japan)

# References

1. ETSI ES 201 873-1 V3.2.1 (2007-02)  
   Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3. Part 1: TTCN-3 Core Language
2. 1/1531-CRL 113 200 Uen  
   Installation Guide for the TITAN TTCN 3 Test Executor
3. 1/198 17-CRL 113 200 Uen  
   User Guide for the TITAN TTCN 3 Test Executor
4. EED/Z/P-03:015 Rev B  
   PDU CNES – TTCNV3 Requirement Specification
5. 109 21-CNL 113 341-5 Uen  
   SCCP Protocol Emulation for TTCN-3 Toolset, Product Revision Information
6. 1/1553-CRL 113 200 Uen

User Documentation for the TITAN TTCN-3 Test Executor

1. ITU-T Recommendation Q.711 (07/96)  
   Specifications of Signalling System No. 7 Signalling connection control part  
   Functional Description of the Signalling Connection Control Part
2. ITU-T Recommendation Q.712 (07/96)  
   Specifications of Signalling System No. 7- Signalling connection control part (SCCP)  
   Definition and function of signalling connection control part messages
3. ITU-T Recommendation Q.713 (07/96)  
   Specifications of Signalling System No. 7- Signalling connection control part (SCCP)  
   Signalling Connection Control Part formats and codes
4. ITU-T Recommendation Q.714 (07/96)  
   Specifications of Signalling System No. 7 - Signalling connection control part  
   Signalling connection control part procedures
5. ANSI T1.112-2001  
   Signalling System Number 7 (SS7) –Signalling Connection Control Part (SCCP)
6. JT-Q711 (23/04/97)  
   Functional Description of the Signalling Connection Control Part (SCCP)
7. JT-Q712 (23/04/97)  
   Definition and Function of SCCP Messages
8. JT-Q713 (20/04/2000)  
   SCCP Formats and Codes
9. JT-Q714 (23/04/97)  
   Signalling Connection Control Part Procedures
10. 1/15517-FAY 112 013/3 Uen   
    SCCP, Introduction (ITU-T, MPT, TTC)
11. 1/15517-FAY 112 019/3 Uen   
    SCCP, Connection-oriented Signalling Procedures (ITU, MPT, TTC)
12. 2/15517-FAY 112 013/4 Uen  
    SCCP, Formats and Codes
13. 3/15517-FAY 112 013/4 Uen  
    SCCP, Connectionless Signalling Procedures (ITU, MPT, TTC)

# Change information