TC-MTS #29

19 - 21 October 1999

Sophia Antipolis

Subject:	DES/MTS-00051 Version 4.0 TSP1+ Specification
Source	STF153 Leader
Date:	6 October, 1999
Document for:	Decision

Background

The draft TSP1+ specification in this Temporary Document represents part of the output from STF153. This is the textual part of the standard revised to highlight the normative requirements for the TSP1+ protocol. Previous versions of the standard depended solely on the SDL to be the normative part but in Version 4.0, it is the text that has been made normative.

Every attempt has been made to maintain the original essential requirements for TSP1+ and any changes to the actual protocol are entirely unintentional. Much of the text included in earlier versions was informative and background material and this has been moved to Annexes B and C at the rear of the document.

The service primitives have been maintained in the document but have been described in the manner of Stage 2 information flows. A table correlating these primitives with TSP1+ protocol messages has been added and the behaviour of both the System Supervisor and a Front End is now described in a tabular form showing normal and exceptional behaviour at both entities in response to external stimuli and received protocol messages.

Less emphasis has been placed on the function of the Adaptation layer as its behaviour cannot be fully standardized without knowledge of the underlying transport and physical layer protocols. TSP1+ is now specified within this document as a peer-to-peer Layer 3 protocol as is normal for any similar standardized system.

The Word Processing macro tools developed by STF98 have been used to ensure that the lists of references and abbreviations are coherent and that the language used is consistent with a normative specification.

TD 08

DES/MTS-00051 V4.0.0 (1999-07)

Methods for Testing and Specification (MTS); Test Synchronization Protocol 1 + (TSP1+) specification

European Telecommunications Standards Institute

Reference

DES/MTS-00051

Keywords

Network Integration Testing (NIT), Test Synchronization Protocol 1 (TSP1), Open Testing Environment (OTE)

Contents

Intellectual Property Rights

Foreword

Introduction

1 Scope

This document specifies the Test Synchronization Protocol 1+ (TSP1+). The purpose of TSP1+ is to achieve functional co-ordination and timing synchronization between two or more Test Synchronization Architectural Elements (TSAE) involved in a distributed testing system.

The purpose of his standard is to specify the essential requirements to be met by any item of test equipment claiming to support TSP1+ as its distributed test synchronization protocol.

This document is applicable to any telecommunications test equipment implementing TSP1+.

Conformance to this standard is achieved by satisfying the requirements identified in the Protocol Implementation Conformance Statement (PICS) proforma in Annex A.

2 References

References may be made to:

a) specific versions of publications (identified by date of publication, edition number, version number, etc.), in which case, subsequent revisions to the referenced document do not apply; or

b) all versions up to and including the identified version (identified by "up to and including" before the version identity); or

c) all versions subsequent to and including the identified version (identified by "onwards" following the version identity); or

d) publications without mention of a specific version, in which case the latest version applies.

A non-specific reference to an ETS shall also be taken to refer to later versions published as an EN with the same number.

2.1 References

[1] ISO/IEC 9646-3 (1998): "Information technology; Open Systems Interconnection; Conformance testing methodology and framework; Part 3: The Tree and Tabular Combined Notation (TTCN) "

[2] ISO/IEC 9646-7 (1995): "Information technology; Open Systems Interconnection; Conformance testing methodology and framework; Part 7: Implementation Conformance Statements

[3] ITU-T Recommendation X.690 (1994): "Information Technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)

[4] ITU-T Recommendation Z.120 (1993): "Messages sequence charts".

[5] INTOOL/OTE/EC026 (1997): "OTE Piloting protocol design specification"

[6] J.Rumbaugh, M.Blaha, W.Premerlani, F.Eddy and W.Lorensen, "Object Oriented Modeling and Design", Prentice-Hall (1991); ISBN 0136298419

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following definitions apply:

Campaign Management Interface (CMI): the command interface between a TSP1+ System Supervisor and the controlling user of the system.

configurations: the arrangement of Test COmponents (TCO) and their Point of Control and Observation (PCO) as defined in the test component configuration declarations of the C-TTCN.

envelope: header and trailer inserted by TSP1+ adaptation layer for any transport protocol.

Front-end Management Interface (FMI): the information interface between a TSP1+ Front End and its user environment.

session: all the information necessary to execute some test belonging to a given configuration.

simple mean of testing: a type of Test Synchronization Architectural Element which only manages PCOs during a test session.

Test Configuration Element: A Point of Control and Observation (PCO) or a Test COmponent (TCO)

Test Programming Interface: The control interface between a TSP1+ Front End and its Main Test Component (MTC)

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ASN.1	Abstract Syntax Notation 1
ATS	Abstract Test Suite
BER	Basic Encoding Rules
CM	Co-ordination Message
CMI	Campaign Management Interface
CP	Co-ordination Point
C-TTCN	Concurrent TTCN
DSS1	Digital Subscriber Signalling system 1
FE	Front End
FMI	Front-end Management Interface
ISO	International Standard Organization
IUT	Implementation Under Test
LAN	Local Area Network
LT	Lower Tester
LTCF	Lower Tester Control Function
MOT	Means Of Testing
MPTM	Multi Party Testing Method
MTC	Main Test Component
NE	Network Element
NIT	Network Integration Testing
OMT	Object Modelling Technique
OSI	Open System Interconnection
PCO	Point of Control and Observation
PDU	Protocol Data Unit
PNO	Public Network Operator
PT	Protocol Tester

PTC	Parallel Test Component
SDL	Specification and Description Language
sMOT	Simple MOT
SS	System Supervisor
TCE	Test Configuration Element
TCO	Test COmponent
TCP	Test Co-ordination Procedure
TCP/IP	Transmission Control Protocol / Internet Protocol
TMN	Telecommunication Management Network
TPI	Test Programming Interface
TSA	Test Synchronization Architecture
TSAE	Test Synchronization Architectural Element
TSP	Test Synchronization Protocol
TSP1	Test Synchronization Protocol 1
TSP2	Test Synchronization Protocol 2
TTCN	Tree and Tabular Combined Notation
UT	Upper Tester
WAN	Wide Area Network

4 The TSP1+ signalling protocol

4.1 TSP1+ description

Test Synchronization Protocol 1+ (TSP1+) is a high level synchronization protocol for test procedures. It contains primitives and messages to manage a complete testing session. TSP1+ is used in a complex testing environment to provide communication between the item of test equipment providing overall control (System Supervisor) and other separate items of test equipment requiring co-ordination and synchronization. The basic relationship between entities in a TSP1+ system is shown in Figure 1 and for the purposes of describing TSP1+, the most significant entities in this arrangement are as follows:

- System Supervisor (SS)

  manages the test execution. It does not provide any support to implement the necessary configurations on physical machines such as the tester or the IUT (as those configuration can be obtained using TMN or a manual approach). The test configuration needs to be clearly defined, well identified and established before starting the test campaign.

- Front End (FE)

  - translates system supervisor messages in to the appropriate format for each physical tester. Messages between two configuration elements handled by the same Front End are not sent to the system supervisor. In other cases messages are sent to the system supervisor which routes them to the appropriate Front End.

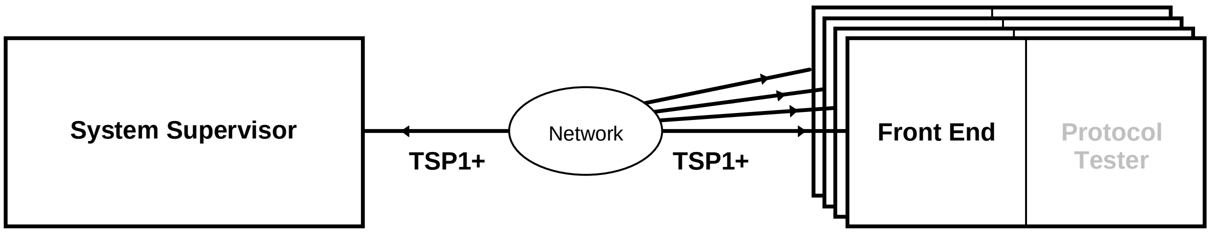


Figure 1: Basic TSP1+ system arrangement

This standard does not specify the underlying network technology for the transport of TSP1+ signalling. It may use directly wired connections, ISDN, TCP/IP LAN or any other standardized or proprietary interconnection method. A more detailed description of the Test Synchronization Architecture (TSA) can be found in Annex B.

The TSP1+ protocol comprises five groups of services to be provided during the different test phases or occurrences. These groups are:

- Group 1, pre testing phase;
- Group 2, testing phase;
- Group 3, post testing phase;
- Group 4, management of exceptional situations;
- Group 5, miscellany.

4.2 TSP1+ operational requirements

4.2.1 System Supervisor requirements

The system supervisor shall be capable of managing the overall execution of test cases in the remote items of test equipment to which it is connected via TSP1+ Front Ends. Before running a test, the supervisor may verify that all test components involved in the test are ready to start. At the end of the test or at the end of test suite execution, it may retrieve the trace of each execution. To achieve this, the system supervisor shall:

- maintain a mapping of the network addresses of each Test Configuration Element (TCE) and its associated Front End (FE);
- provide routeing capabilities to each FE;
- communicate with each Front End;
- provide the ability to control and manage test sessions.

4.2.2 Front End requirements

A Front End provides a control interface between its associated protocol tester and the system supervisor. It shall route all synchronization messages to its TCE. If a Front End identifies a message that is not sent to one of its TCEs, it shall forwards that message to the system supervisor which shall then route it to the appropriate destination. A FE shall:

- provide routeing capabilities toward its TCEs;
- communicate with the system supervisor;
- communicate with its PTs.

4.3 TSP1+ Service Primitive Description

In the following, the services provided by each group are described in terms of primitives. The services are described from the System Supervisor point of view.

4.3.1 Group 1

The Group 1 services shall be used to open a test session, to verify and establish a session of testing and to provide values for all parameters.

4.3.1.1 Test configuration establishment

The System Supervisor shall use the OPEN_SESSION service primitive to open a testing session. On receipt of this primitive, the test system shall be capable of executing the tests.

The OPEN_SESSION primitive shall contain the service elements defined in Table 1.

Table 1: Contents of Service Primitive OPEN_SESSION

Element	Description	Allowed values	Request	Confirm
ETS_ID	Executable test suite identification		Mandatory	
SESSION_ID	Identifier of the session that is to be opened		Mandatory	
CONF_ELEM_ID_LIST	A sequence of configuration element identifiers.	PCO; TCO (Note 1)	Mandatory	
REAL_CONFIG	The mapping between abstract elements (TCO or PCO) of the complete TTCN configuration and TSP1+ addresses		Optional (Note 2)	
TRACE_CONFIG	The method to be used for acquiring traces during test execution.	Delete traces during execution; Save traces for post-execution transfer to supervisor	Mandatory	

SE_ERROR	Returned error code for this operation	Op. successful;
		
		Unknown ETS;
		
		Unknown session;
		
		MOT not ready
				Mandatory

Note 1: This value is permitted for compatibility with TSP1. Note 2: If this information element is omitted, all PDUs shall be sent to the supervisor which shall decode the requests and forward them to the appropriate front end				

4.3.1.2 Test session checking

The System Supervisor shall use the CHECK_SESSION service primitive to determine whether a previously initialized session is still established

The CHECK_SESSION primitive shall contain the service elements defined in Table 2.

Table 2: Contents of Service Primitive CHECK_SESSION

Element	Description	Allowed values	Request	Confirm
SE_ERROR	Returned error code for this operation	Op. successful; MOT not ready		Mandatory

4.3.1.3 Modification of the test suite parameters

If required, the System Supervisor shall use the SET_PARAMETER service primitive to change the values of the test suite parameters in an established session.

The SET_PARAMETER primitive shall contain the service elements defined in Table 3.

Table 3: Contents of Service Primitive SET_PARAMETER

Element	Description	Allowed values	Request	Confirm
PARAM_LIST	List of the names and values of the parameters to be changed		Mandatory	
SE_ERROR	Returned error code for this operation	Op. successful; MOT not ready		Mandatory

4.3.1.4 Setting a unique time stamp

The System Supervisor shall use the SET_TIME service primitive to synchronize all of the test components in an established session with the same time stamp.

The SET_TIME primitive shall contain the service elements defined in Table 4.

Table 4: Contents of Service Primitive SET_TIME

Element	Description	Allowed values	Request	Confirm
TIMESTAMP	The reference time stamp of the System Supervisor		Mandatory	
SE_ERROR	Returned error code for this operation	Op. successful; MOT not ready		Mandatory

4.3.1.5 Looking for the TSP1+ services available in the Front End

The System Supervisor shall use the LIST_FE_SERVICES service primitive to determine which services are available in a Front End.

The LIST_FE_SERVICES primitive shall contain the service elements defined in Table 5.

Table 5: Contents of Service Primitive LIST_FE_SERVICES

Element	Description	Allowed values	Request	Confirm
FE_ID	Destination Front End		Mandatory	
SESSION_SERVICE_LIST	List of services available			Mandatory
FE_ERROR	Returned error code for this operation	Op. successful;		Mandatory

4.3.2 Group 2

The services provided by Group 2 shall be used to coordinate the execution of tests in terms of:

- launching;
- synchronization;
- exchange of messages;
- verdict assignment.

NOTE: Unlike Group 1, primitives introduced here correspond to concepts and keywords specified in concurrent TTCN (ISO/IEC 9646-3 [1])

4.3.2.1 Test execution launch

The System Supervisor shall use the CREATE_TCO service primitive to load and start execution of an executable parallel test component.

The CREATE_TCO primitive shall contain the service elements defined in Table 6.

Table 6: Contents of Service Primitive CREATE_TCO

Element	Description	Allowed values	Request	Confirm
TCO_ID	Identification of the executable test component		Mandatory	
TEST_CASE_ID	The test case to be launched		Mandatory	
TREE	The sub-tree to be launched			
PARAM_LIST	List of test case parameters		Mandatory	
TCO_ERROR	Returned error code for this operation	Op. Successful; MOT not ready		Mandatory

4.3.2.2 Exchanging messages

The following service primitives shall be used by test components to exchange test messages through test interfaces:

- RCV_MSG:

  sent when a message (CM, PDU, ASP) is received in the local queue related to the specified interface under test (PCO, CP);

- SEND_MSG:

  causes a message (ASP, CM, PDU) to be sent to the queue related to the specified interface under test (PCO, CP).

Note: The interface (CP, PCO) model is a first-in first-out (FIFO) queue, as specified in the ISO 9646-3 [1].

The RCV_MSG primitive shall contain the service elements defined in Table 7.

Table 7: Contents of Service Primitive RCV_MSG

Element	Description	Allowed values	Request	Confirm
SRC_ID	Identification of the sending TCE		Mandatory	
DEST_ID	Identification of the destination TCE		Mandatory	
INTERFACE_ID	Identification of the interface through which the message is to be transmitted		Mandatory	

INFO_TYPE_ID	Identification of the message type	PDU;
		
		ASP;
		
		CM
			Mandatory	

VALUE	Message received		Mandatory	
TCE_ERROR	Returned error code for this operation	Op. Successful;	Mandatory	

The SEND_MSG primitive shall contain the service elements defined in Table 8.

Table 8: Contents of Service Primitive SEND_MSG

Element	Description	Allowed values	Request	Confirm
SRC_ID	Identification of the sending TCE		Mandatory	
DEST_ID	Identification of the destination TCE		Mandatory	
INTERFACE_ID	Identification of the interface through which the message is to be transmitted		Mandatory	

INFO_TYPE_ID	Identification of the message type	PDU;
		
		ASP;
		
		CM
			Mandatory	

VALUE	Message received		Mandatory	
TCE_ERROR	Returned error code for this operation	Op. Successful;	Mandatory	

4.3.2.3 Test completion

The Front End serving the main test component shall use the CHECK_TCO_COMPLETED service primitive to check whether the execution of a parallel test component has completed and is waiting for the conclusive verdict message.

The CHECK_TCO_COMPLETED primitive shall contain the service elements defined in Table 9.

Table 9: Contents of Service Primitive CHECK_TCO_COMPLETED

Element	Description	Allowed values	Request	Confirm
TCO_ID	The parallel test component to be checked		Mandatory	
TCO_VERDICT_TYPE	Type of test verdict specified	Final	Mandatory	

TCO_VERDICT_VALUE	The local verdict value	Pass;
		
		Fail;
		
		Inconclusive
			Mandatory	

TCO_ERROR	Returned error code for this operation	Op. Successful;	Mandatory	

4.3.2.4 Temporary verdict assignment

A Front End serving an executable parallel test components shall use the UPDATE_VERDICT service primitive to transmit a temporary local verdict.

The UPDATE_VERDICT primitive shall contain the service elements defined in Table 10.

Table 10: Contents of Service Primitive UPDATE_VERDICT

Element	Description	Allowed values	Request	Confirm
TCO_ID	The parallel test component reporting the verdict		Mandatory	
TCO_VERDICT_TYPE	Type of test verdict specified	Intermediate	Mandatory	

TCO_VERDICT_VALUE	The local verdict value	Pass;
		
		Fail;
		
		Inconclusive
			Mandatory	

TCO_ERROR	Returned error code for this operation	Op. Successful;	Mandatory	

4.3.2.5 Notification during execution

The System Supervisor shall use the NOTIFICATION_EXEC_TRACE service primitive to request information from a Front End regarding the execution of a test at a TCO or PCO.

The NOTIFICATION_EXEC_TRACE primitive shall contain the service elements defined in Table 11.

Table 11: Contents of Service Primitive NOTIFICATION_EXEC_TRACE

Element	Description	Allowed values	Request	Confirm
SRC_ID	The executable test component identifier		Mandatory	

STEP	The reported trace identifier	First;
		
		Current;
		
		Last;
		
		All
				Mandatory

TRACE_TYPE	The type of trace reported			Mandatory
TIME_STAMP	The time reference for the testing phase			Mandatory
TRACE	The reported trace			Mandatory
INFORMATION	Additional information			Optional
TCE_ERROR	The returned error code for this operation	Op. successful; Trace not available		Mandatory

4.3.2.6 Update variable

The front end serving the main test component shall use the UPDATE_VARIABLE service primitive to notify the System Supervisor of a change to a test suite variable.

The UPDATE_VARIABLE primitive shall contain the service elements defined in Table 12.

Table 12: Contents of Service Primitive UPDATE_VARIABLE

Element	Description	Allowed values	Request	Confirm
VARIABLE	A sequence of variable names and values		Mandatory	
TCO_ERROR	The returned error code for this operation	Op. successful		Mandatory

4.3.3 Group 3

The Group 3 services described in this subclause shall be used to request traces and other information related to the result of the execution of the TCOs.

4.3.3.1 Transferring a trace

The System Supervisor shall use the ASK_TRACE service primitive to request the transfer of the execution trace of a test configuration element.

The ASK_TRACE primitive shall contain the service elements defined in Table 13.

Table 13: Contents of Service Primitive ASK_TRACE

Element	Description	Allowed values	Request	Confirm
SRC_ID	The configuration element identifier		Mandatory	

STEP	The progress indicator for the reported trace	First;
		
		Current;
		
		Last;
		
		All
				Mandatory

TRACE_TYPE	The type of trace reported			Mandatory
TIME_STAMP	The time reference for the testing phase			Mandatory
TRACE	The reported trace			Mandatory
INFORMATION	Additional information			Optional
TCE_ERROR	The returned error code for this operation	Op. successful; Trace not available		Mandatory

4.3.3.2 Closing a test session

The System Supervisor shall use the CLOSE_SESSION service primitive to close a test session.

The CLOSE_SESSION primitive shall contain the service elements defined in Table 14.

Table 14: Contents of Service Primitive CLOSE_SESSION

Element	Description	Allowed values	Request	Confirm
SE_ERROR	The returned error code for this operation	Op. successful; MOT not ready		Mandatory

4.3.4 Group 4

The service provided by Group 4 shall be used to solve the problem that could occur during the initialization, execution and closing phases.

4.3.4.1 Cancel a running operation

Having initiated an operation, the System Supervisor needs to have the capability to cancel it.

The System Supervisor shall use the CANCEL_OP service primitive to interrupt an operation which is already in progress. After the CANCEL_OP procedure, the System Supervisor shall return to the state in which it was before that operation.

The CANCEL_OP primitive shall contain the service elements defined in Table 15.

Table 15: Contents of Service Primitive CANCEL_OP

Element	Description	Allowed values	Request	Confirm
CONF_ELEM_ID	Interface (PCO/TCO) where operation is to be cancelled		Mandatory	
TCE_ERROR	The returned error code for this operation	Op. successful;		Mandatory

4.3.5 Group 5

4.3.5.1 The DISPLAY feature

The DISPLAY service primitive shall be used by Front Ends and the System Supervisor to transmit information which is to be displayed to the operator. It may be used to send such information from:

- System Supervisor to a Front End;
- a Front End to another Front End.

The DISPLAY primitive shall contain the service elements defined in Table 16.

Table 16: Contents of Service Primitive DISPLAY

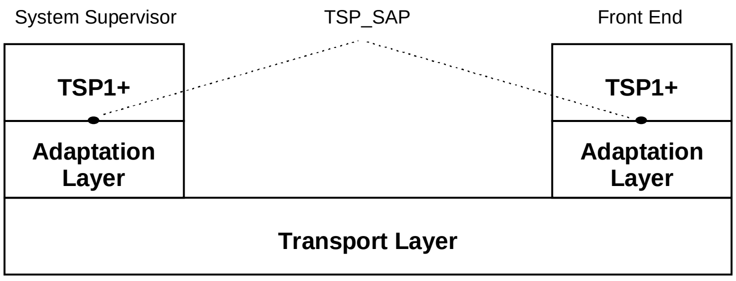
Element	Description	Allowed values	Request	Confirm
FE_ID	Destination Front End		Mandatory	
DISP_MSG	Information to be displayed		Mandatory	

4.4 TSP1+ Protocol Description

4.4.1 Simple layered model

To be able to implement the service primitives described in subclause 4.3, it is necessary to specify a protocol and set of messages. This will provide an end-to-end service between a System Supervisor and one or more items of remote test equipment.

Figure 2 uses a simplified ISO layered model to show the general architecture of the TSP1+ protocol. PDUs are



exchanged by the System Supervisor and Front Ends on the TSP1+ layer.

Figure 2: TSP1+ Layers

4.4.2 Rejecting a corrupted or an out of sequence message

A corrupt or out of sequence TSP1+ protocol message shall be rejected by a Front End by sending an Error PDU with the Error Code set to errUnrecognizedTSP1PDU (see subclause 4.4.5).

4.4.3 TSP1+ coding requirements

4.4.3.1 TSP1+ Message structure

In order to support TSP1+ when transported over protocols that do not maintain boundaries between SDUs (e.g. TCP/IP), each TSP1+ PDU shall be encoded according to the details given in Figure 3.

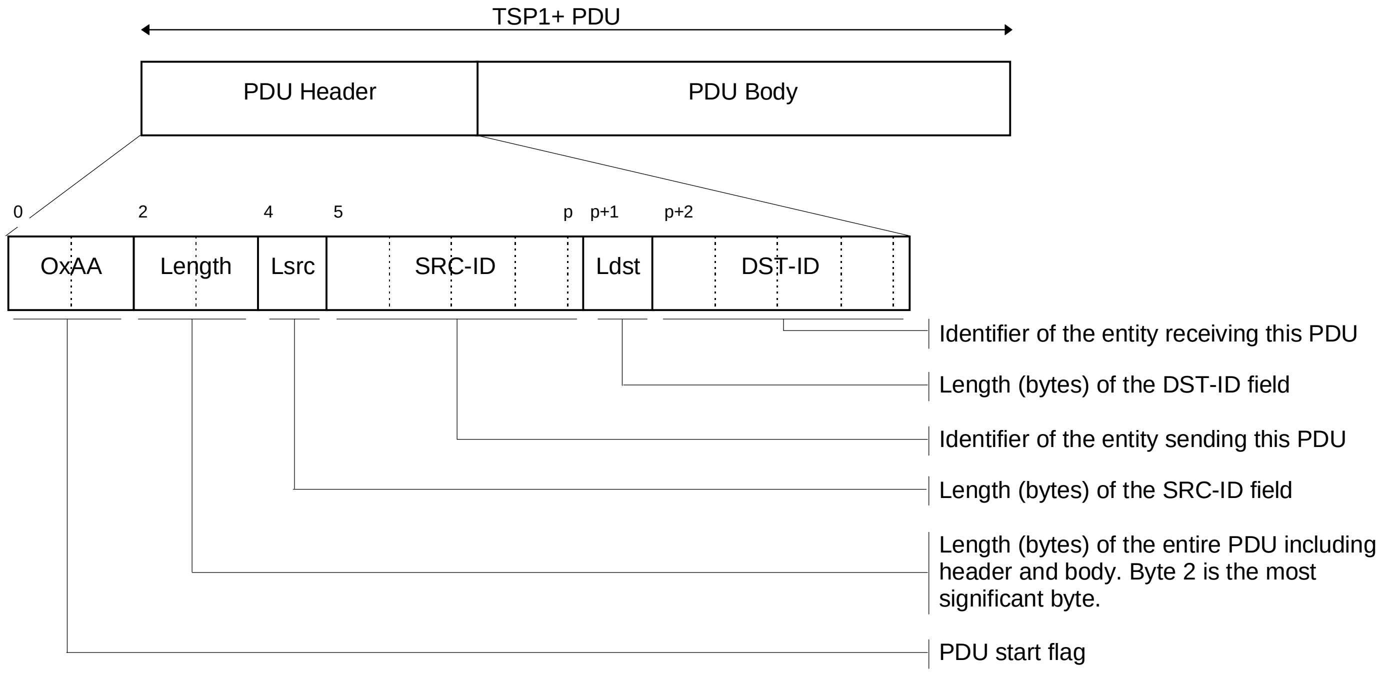

TSP1+ PDU
PDU Header
PDU Body
0
2
4
5
p
p+1
p+2
OxAA
Length
Lsrc
SRC-ID
Ldst
DST-ID
Identifier of the entity receiving this PDU
Length (bytes) of the DST-ID field
Identifier of the entity sending this PDU
Length (bytes) of the SRC-ID field
Length (bytes) of the entire PDU including header and body. Byte 2 is the most significant byte.
PDU start flag

Figure 3: Generalized PDU encoding

SRC-ID and DST-ID may additionally appear in the PDU body. Although this could lead to some redundancy in the encoded PDU, it does mean that a Front End can easily access these values without having to decode the PDU body. This mechanism is compliant with the INTOOL/OTE/EC026 specification, "OTE Piloting protocol design specification" [5].

4.4.3.2 Operations

The operations defined in Abstract Syntax Notation number 1 (ASN.1) in Table 17 shall apply. They shall be encoded using the Basic Encoding Rules (BER) defined in ITU-T Recommendation X.690 [3.]

Table 17: Operations in support of TSP1+

```
TSPone-Operations {itu-t(0) identified-organization (4) etsi (0)
                       nnnn basic-operations (0)}

DEFINITIONS ::=

BEGIN

EXPORTS TspAddress, TSP1_PDU;

TSP1_PDU::= CHOICE {
          tsp1_error                              [ 0] IMPLICIT TSP1_ERR,
          tsp1_init                               [ 1] IMPLICIT TSP1_INIT,
          tsp1_init_ack                           [ 2] IMPLICIT TSP1_INIT_ACK,
          tsp1_init_complete                      [ 3] IMPLICIT TSP1_INIT_COMPLETE,
          tsp1_chk_conf                           [ 4] IMPLICIT TSP1_CHK_CONF,
          tsp1_chk_conf_ack                       [ 5] IMPLICIT TSP1_CHK_CONF_ACK,
          tsp1_set_parameter                      [ 6] IMPLICIT TSP1_SET_PARAMETER,
          tsp1_set_parameter_ack                  [ 7] IMPLICIT TSP1_SET_PARAMETER_ACK,
          tsp1_set_time                           [ 8] IMPLICIT TSP1_SET_TIME,
          tsp1_set_time_ack                       [ 9] IMPLICIT TSP1_SET_TIME_ACK,
          tsp1_list_fe_services                   [10] IMPLICIT TSP1_LIST_FE_SERVICES,
          tsp1_list_fe_services_ack               [11] IMPLICIT TSP1_LIST_FE_SERVICES_ACK,
          tsp1_create                             [12] IMPLICIT TSP1_CREATE,
          tsp1_create_ack                         [13] IMPLICIT TSP1_CREATE_ACK,
          tsp1_info                               [14] IMPLICIT TSP1_INFO,
          tsp1_verdict                            [15] IMPLICIT TSP1_UPDATE_VERDICT,
          tsp1_update_variable                    [16] IMPLICIT TSP1_UPDATE_VARIABLE,

          tsp1_ask_trace                          [18] IMPLICIT TSP1_ASK_TRACE,
          tsp1_ask_trace_ack                      [19] IMPLICIT TSP1_ASK_TRACE_ACK,
          tsp1_end                                [20] IMPLICIT TSP1_END,
          tsp1_end_ack                            [21] IMPLICIT TSP1_END_ACK,

          tsp1_cancel_op                          [23] IMPLICIT TSP1_CANCEL_OP,
          tsp1_cancel_op_ack                      [24] IMPLICIT TSP1_CANCEL_OP_ACK,
          tsp1_display                            [25] IMPLICIT TSP1_DISPLAY
          }

  -- /******************** PDU body description ********************/

  FeId                     ::= PrintableString         -- Front End Identifier

  EtsId                    ::= PrintableString         -- Executable Test Suite Identifier

  ConfElemId               ::= PrintableString         -- PCO or TCO Identifier
  ConfElemIdList           ::= SEQUENCE OF ConfElemId

  SessionId                ::= PrintableString         -- Session Identifier

  TestCaseId               ::= PrintableString         -- Test Case Identifier

  TreeId                   ::= PrintableString         -- Main Tree Identifier

  Param                    ::= SEQUENCE     { param_id       PrintableString,
                                              param_value    OCTET STRING }
  ParamList                ::= SEQUENCE OF Param

  Variable                 ::= SEQUENCE     { variable_name  PrintableString,
                                              variable_value OCTET STRING }

  Msg                      ::= OCTET STRING
  DisplayMsg               ::= PrintableString
```

```
-- **** ASN1 TSP1 description part 2/4

TcoId                    ::= PrintableString         -- TCO Identifier (MTC or PTC)
InterfaceId              ::= PrintableString         -- PCO or CP Identifier
InfoTypeId               ::= PrintableString         -- ASP type, PDU type or CM type Identifier

ConfigId                 ::= PrintableString         -- Abstract real config identifier

TspAddress               ::= SEQUENCE    { length            BIT STRING(SIZE(8)),
                                           value             OCTET STRING }

ConfElemAddress          ::= SEQUENCE    { conf_elem_id       ConfElemId,
                                           conf_elem_address  TspAddress }
ConfElemAddressList      ::= SEQUENCE OF ConfElemAddress

RealConfig               ::= SEQUENCE    { config_id       ConfigId,
                                           mapping         ConfElemAddressList }

Step                     ::= INTEGER     { first                   (1),
                                           current                 (0),
                                           last                    (2),
                                           first_last              (3) }

TraceFilter              ::= INTEGER     { delete                  (1),
                                           notify                  (2),
                                           record                  (3) }

TraceSet                 ::= SEQUENCE    { conf_elem_id    ConfElemId,
                                           trace_filter    TraceFilter }
TraceConfig              ::=SEQUENCE OF TraceSet
TraceType                ::=PrintableString

ServiceId                ::= INTEGER     { tsp1AdaptationLayer     (0),
                                           tsp1Init                (1),
                                           tsp1ChkConf             (2),
                                           tsp1SetParameter        (3),
                                           tsp1SetTime             (4),
                                           tsp1ListFeServices      (5),
                                           tsp1Create              (6),
                                           tsp1Info                (7),
                                           tsp1Verdict             (8),
                                           tsp1UpdateVariable      (9),
                                           tsp1AskTrace           (10),
                                           tsp1End                (11),
                                           tsp1CancelOp           (12),
                                           tsp1Display            (13) }
ServiceList              ::= SEQUENCE OF ServiceId

VerdictType              ::= INTEGER     { intermediate_verdict    (0),
                                           final_verdict           (1) }
VerdictValue             ::= INTEGER     { fail                    (0),
                                           inconc                  (1),
                                           pass                    (2) }

ErrorCode                ::= INTEGER     { okmessage               (0),
                                           errMotNotReady          (1),
                                           errMotNotConnected      (2),
                                           errUnknownEts           (3),
                                           errUnknownSession       (4),
                                           errTimeNotAssigned      (5),
                                           errProcessingFailure    (6),
                                           errBadInvocationContext (7),
                                           errArgumentType         (8),
                                           errMissingArgument      (9),
                                           errBadArgumentFormat   (10),
                                           errBadArgumentContent  (11),

                                           errUnrecognizedTSP1PDU (85),
                                           errOutOfSequenceMessage (86),
                                           errTreeNotFound        (87),

                                           errTraceNotAvailable   (170),
                                           errCommunicationLost   (200),
                                           errTesterCrash         (201),
                                           errUnknownResource     (255) }
```

```
  --**** ASN1 TSP1 description part 3/4 ***

  Tsp1ErrorParam           ::= SEQUENCE { conf_elem_id [0] ConfElemId OPTIONAL,
                                          errString    [1] PrintableString }

  TSP1_ERR                 ::= SEQUENCE { service_id       ServiceId,
                                          err_code         ErrorCode,
                                          err_param        Tsp1ErrorParam }

  TSP1_INIT                ::= SEQUENCE { ets_id            EtsId,
                                          session_id        SessionId,
                                          conf_elem_id_list ConfElemIdList,
                                          real_config   [0] RealConfig  OPTIONAL,
                                          trace_config  [1] TraceConfig OPTIONAL }

  TSP1_INIT_ACK            ::= NULL

  TSP1_INIT_COMPLETE       ::= NULL

  TSP1_CHK_CONF            ::= NULL

  TSP1_CHK_CONF_ACK        ::= NULL

  TSP1_SET_PARAMETER       ::= SEQUENCE { param_list    ParamList }

  TSP1_SET_PARAMETER_ACK   ::= NULL

  TSP1_SET_TIME            ::= SEQUENCE { timestamp         GeneralizedTime }

  TSP1_SET_TIME_ACK        ::= NULL

  TSP1_LIST_FE_SERVICES    ::= NULL

  TSP1_LIST_FE_SERVICES_ACK::= SEQUENCE { fe_service_list   ServiceList }

  TSP1_CREATE              ::= SEQUENCE { tco_id        [0] TcoId         OPTIONAL,
                                          test_case_id  [1] TestCaseId,
                                          tree              TreeId        OPTIONAL,
                                          param_list        ParamList     OPTIONAL }

  TSP1_CREATE_ACK          ::= SEQUENCE { tco_id            TcoId }

  TSP1_INFO                ::= SEQUENCE { src_id            ConfElemId,
                                          dest_id           ConfElemId,
                                          interface_id      InterfaceId,
                                          info_type_id      InfoTypeId,
                                          value             Msg }

  TSP1_UPDATE_VERDICT      ::= SEQUENCE { tco_id       TcoId,
                                          tco_verdict_type   VerdictType,
                                          tco_verdict_value  VerdictValue }

  TSP1_UPDATE_VARIABLE     ::= Variable

  TSP1_ASK_TRACE           ::= SEQUENCE { src_id            ConfElemId }

  TSP1_ASK_TRACE_ACK       ::= SEQUENCE { step          Step,
                                          src_id             ConfElemId,
                                          trace_type         TraceType,
                                          time_stamp         GeneralizedTime,
                                          trace              PrintableString,
                                          information SEQUENCE { dest_id      ConfElemId,
                                                                 info_type_id PrintableString,
                                                                 value        Msg,
                                                                 error_code   ErrorCode } OPTIONAL }

  TSP1_END                 ::= NULL
  TSP1_END_ACK             ::= NULL

  TSP1_CANCEL_OP           ::= SEQUENCE { conf_elem_id  ConfElemId OPTIONAL }
  TSP1_CANCEL_OP_ACK       ::= SEQUENCE { conf_elem_id  ConfElemId OPTIONAL }

  TSP1_DISPLAY             ::= SEQUENCE { fe_id             FeId,
                                          disp_msg          DisplayMsg }

END
```

4.4.4 Relationship between service primitives and TSP1+ protocol messages

The service primitives described in subclause 4.3 shall cause TSP1+ protocol messages to be generated according to the information shown in Table 18.

Table 18: Service primitives and their associated TSP1+ protocol messages

Primitive	Parameters	TSP1+ Message	Parameters
OPEN_SESSION	ETS_ID SESSION_ID CONF_ELEM_ID_LIST REAL_CONFIG SE_ERROR	TSP1_INIT	ETS_ID SESSION_ID TCO_ID_LIST
CHECK_SESSION	SE_ERROR	TSP1_CHK_CONF	
SET_PARAMETER	PARAM_LIST SE_ERROR	TSP1_SET_PARAMETER	PARAM_LIST
SET_TIME	TIMESTAMP SE_ERROR	TSP1_SET_TIME	TIMESTAMP
LIST_FE_SERVICES	FE_ID SESSION_SERVICE_LIS T FE_ERROR	TSP1_LIST_FE_SERVICES	FE_SERVICE_LIST
CREATE_TCO	TCO_ID TEST_CASE_ID TREE PARAM_LIST TCO_ERROR	TSP1_CREATE	TCO_ID TEST_CASE_ID TREE PARAM_LIST
RCV_MSG	SRC_ID DEST_ID INTERFACE_ID INFO_TYPE_ID VALUE	TSP1_INFO	SRC_ID DEST_ID INFO_ID INFO_TYPE_ID VALUE
SEND_MSG	SRC_ID DEST_ID INTERFACE_ID INFO_TYPE_ID VALUE	TSP1_INFO	SRC_ID DEST_ID INFO_ID INFO_TYPE_ID VALUE
CHECK_TCO_COMPLETED	TCO_ID TCO_VERDICT_TYPE TCO_VERDICT_VALUE	TSP1_VERDICT (Note 1)	TCO_ID TCO_VERDICT_TYPE TCO_VERDICT_VALUE
UPDATE_VERDICT	TCO_ID TCO_VERDICT_TYPE TCO_VERDICT_VALUE	TSP_VERDICT (Note 2)	TCO_ID TCO_VERDICT_TYPE TCO_VERDICT_VALUE
NOTIFICATION_EXEC_TRACE	SRC_ID STEP TRACE_TYPE TIME_STAMP TRACE INFORMATION TCE_ERROR	TSP1_ASK_TRACE	SRC_ID STEP TRACE_TYPE TIME_STAMP TRACE INFORMATION
UPDATE_VARIABLE	VARIABLE TCO_ERROR	TSP_UPDATE_VARIABLE	VARIABLE
ASK_TRACE	SRC_ID STEP TRACE_TYPE TIME_STAMP TRACE INFORMATION TCE_ERROR	TSP1_ASK_TRACE	SRC_ID STEP TRACE_TYPE TIME_STAMP TRACE INFORMATION
CLOSE_SESSION	SE_ERROR	TSP1_END	
CANCEL_OP	CONF_ELEM_ID TCE_ERROR	TSP1_CANCEL_OP	CONF_ELEM_ID
DISPLAY	FE_ID DISP_MSG	TSP1_DISPLAY	FE_ID DISP_MSG

4.4.5 Message sequences

This subclause describes some typical message flows for TSP1+ using the Message Sequence Chart (MSC) notation specified in ITU-T Recommendation Z.120 [4]. The figures show messages exchanged between the System Supervisor and a Front End.

4.4.5.1 General PDU error handling

In the event that a Front End is unable to complete the processing of a TSP1+ PDU received from the System Supervisor, the TSP1_ERROR pdu shall be used to give a negative acknowledgement instead of the expected positive acknowledgement normally associated with that PDU. An example of this error handling method is shown in Figure 4.

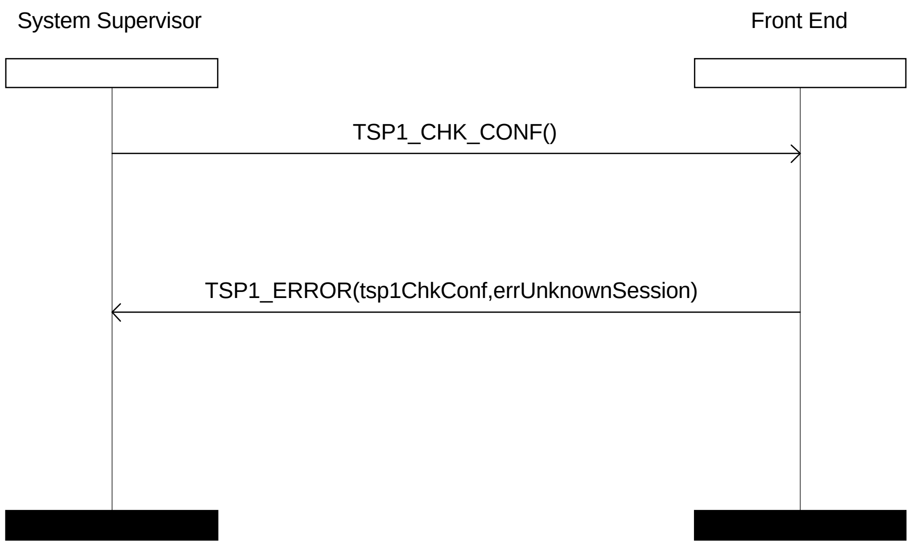


Figure 4: Example message sequence for TSP1+ processing error

If a PDU is received and is not recognisable as a defined TSP1+ PDU, it shall be rejected with an error code of "errUnrecognizedTSP1PDU" as shown in Figure 5.

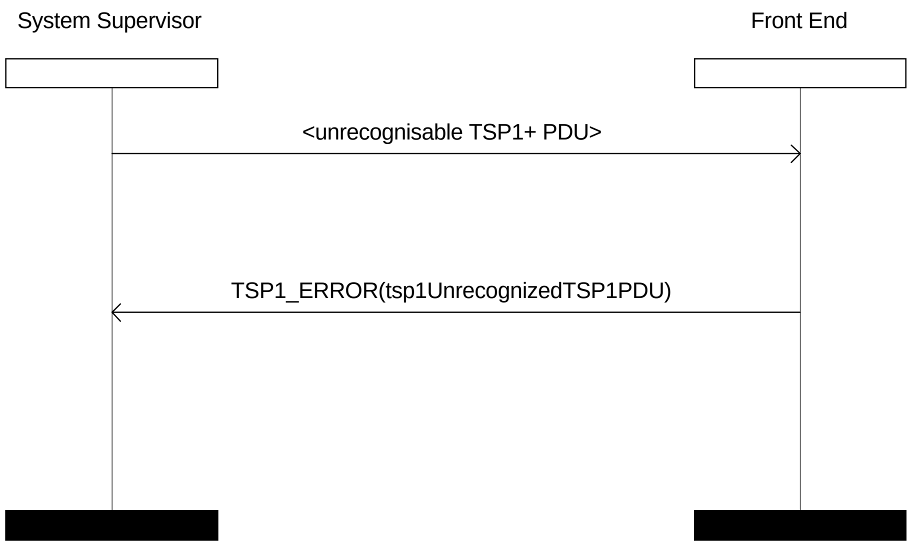

System Supervisor
Front End
<unrecognisable TSP1+ PDU>
TSP1_ERROR(tsp1UnrecognizedTSP1PDU)

Figure 5: Generic message sequence for rejected unrecognisable TSP1+ PDU

The MSCs in Figure 6 to Figure 23 show examples of the normal transmission of TSP1+ PDUs.

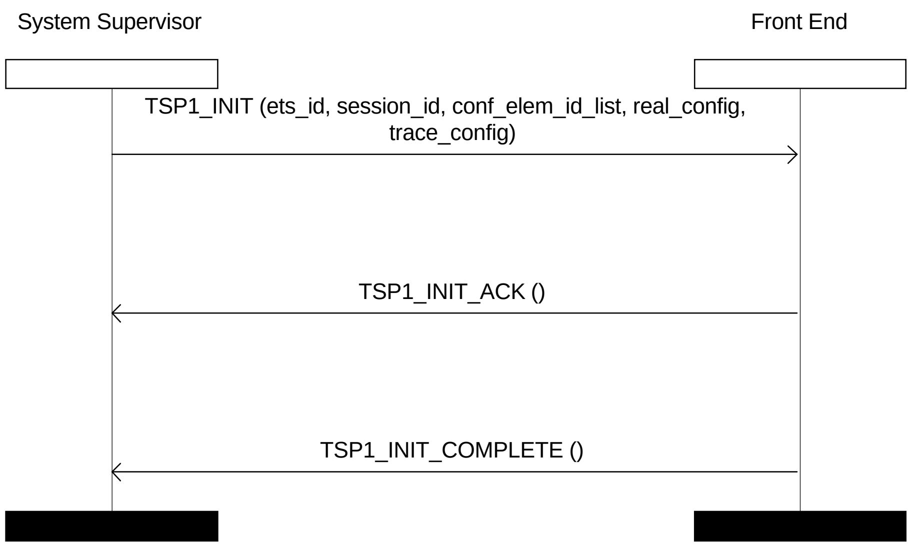

System Supervisor
Front End
TSP1_INIT (ets_id, session_id, conf_elem_id_list, real_config, trace_config)
TSP1_INIT_ACK ()
TSP1_INIT_COMPLETE ()

Figure 6: Example of normal operation of the TSP1_INIT PDU

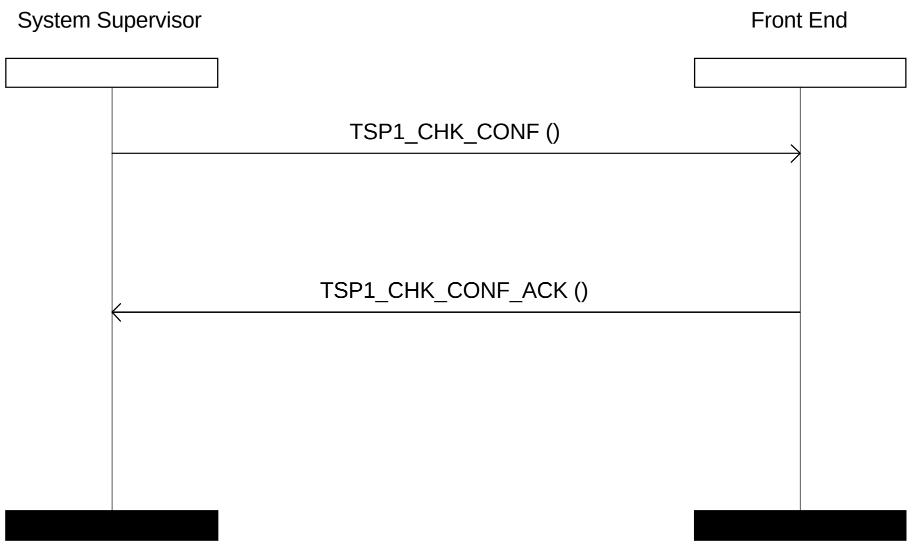


Figure 7: Example of normal operation of the TSP1_CHK_CONF PDU

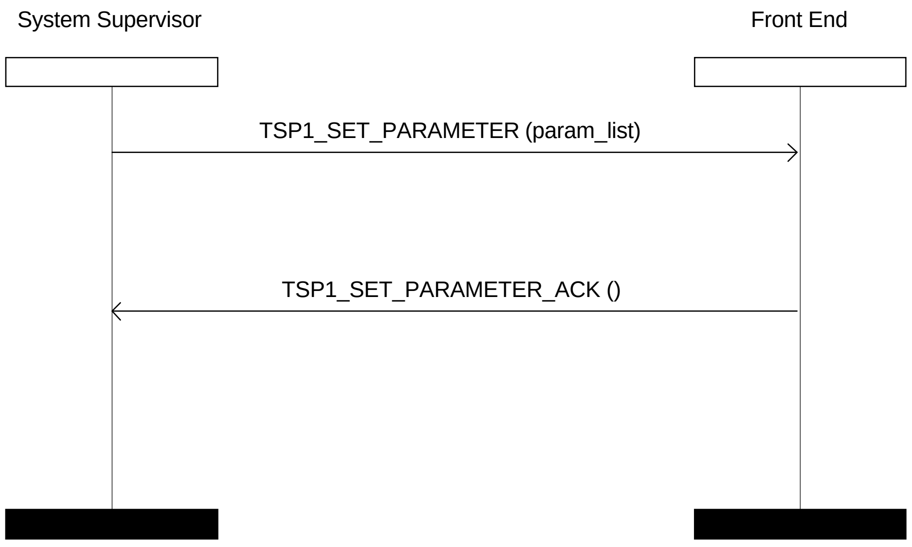


Figure 8: Example of normal operation of the TSP1_SET_PARAMETER PDU

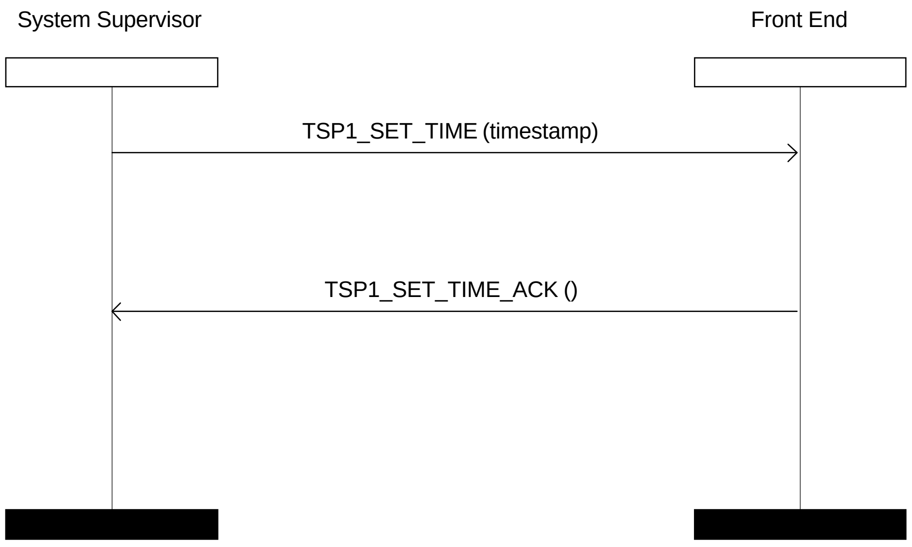


Figure 9: Example of normal operation of the TSP1_SET_TIME PDU

NOTE: The TSP1_SET_TIME PDU can be used to synchronize the running clock times within the System Supervisor and the Front Ends. It may be necessary to add an offset to the System Supervisor time to take account of network transmission and processing delays. The method for determining the value of this offset is beyond the scope of this standard.

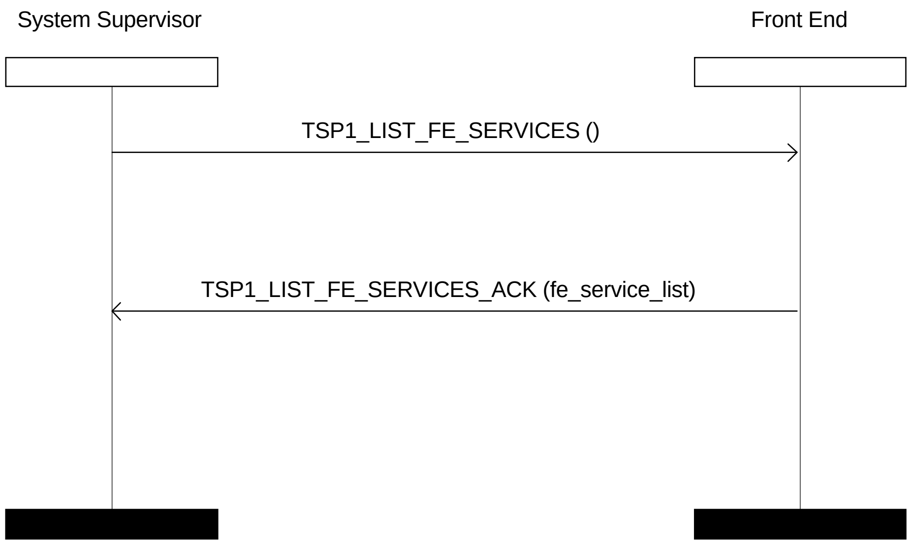


Figure 10: Example of normal operation of the TSP1_LIST_FE_SERVICES PDU

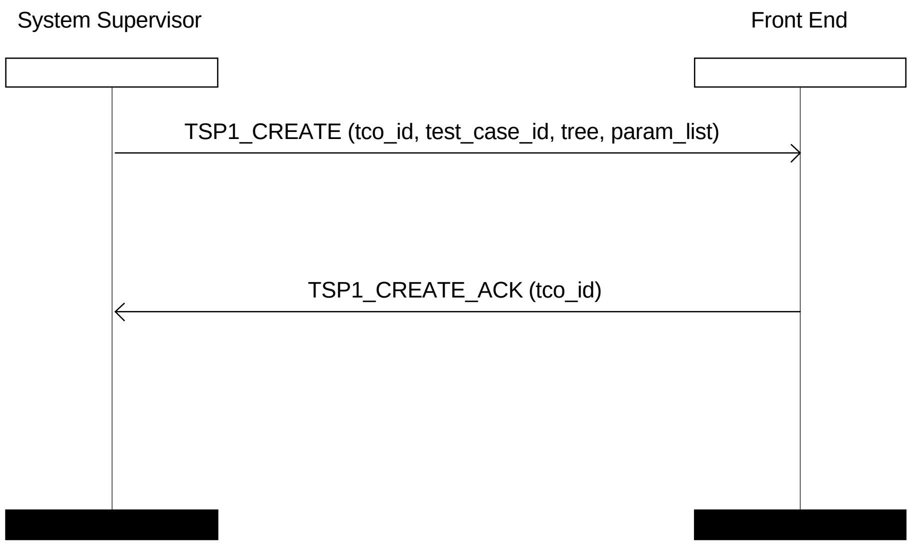


Figure 11: Example of normal operation of the TSP1_CREATE PDU

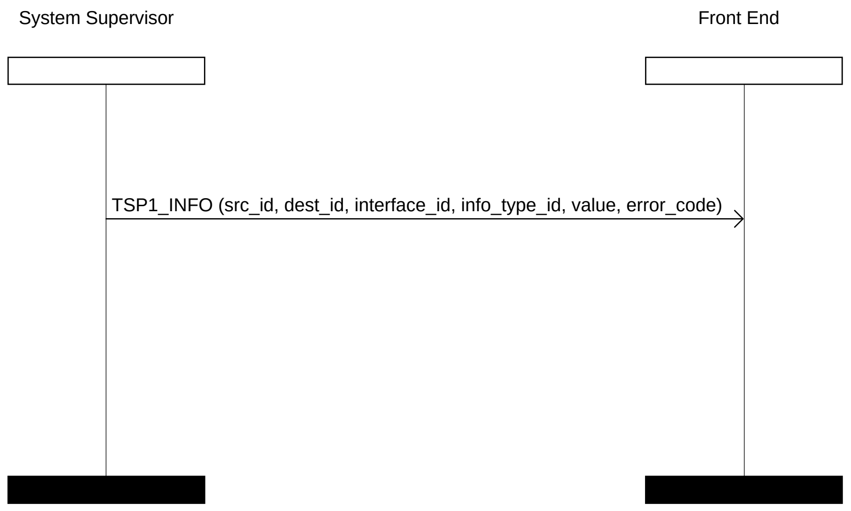

System Supervisor
Front End
TSP1_INFO (src_id, dest_id, interface_id, info_type_id, value, error_code)

Figure 12: Example of normal operation of the TSP1_INFO PDU to send a message from the System Supervisor to a Front End

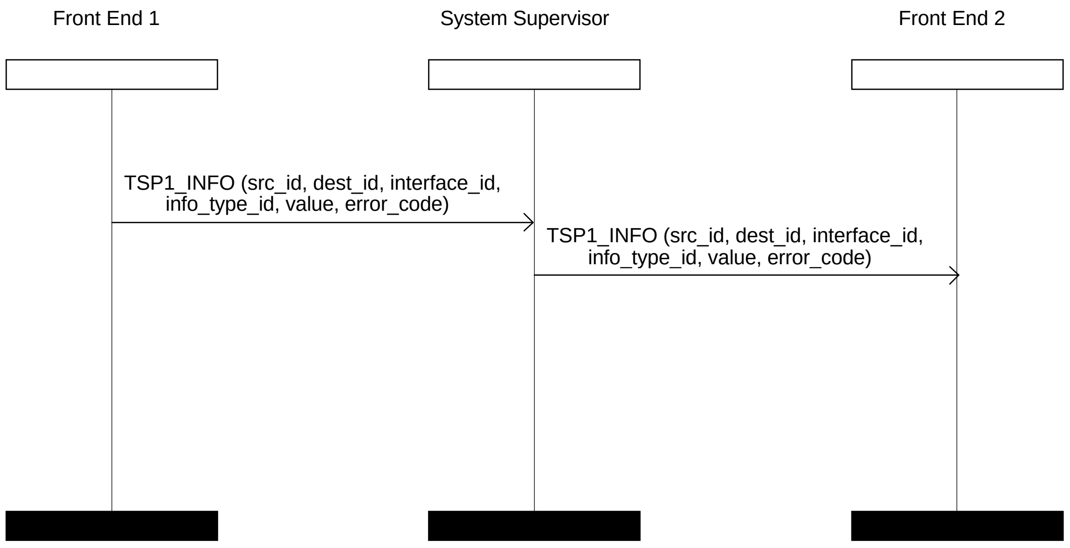

Front End 1
System Supervisor
Front End 2
TSP1_INFO (src_id, dest_id, interface_id, info_type_id, value, error_code)
TSP1_INFO (src_id, dest_id, interface_id, info_type_id, value, error_code)

Figure 13: Example of normal operation of the TSP1_INFO PDU to send a message from one Front End to another

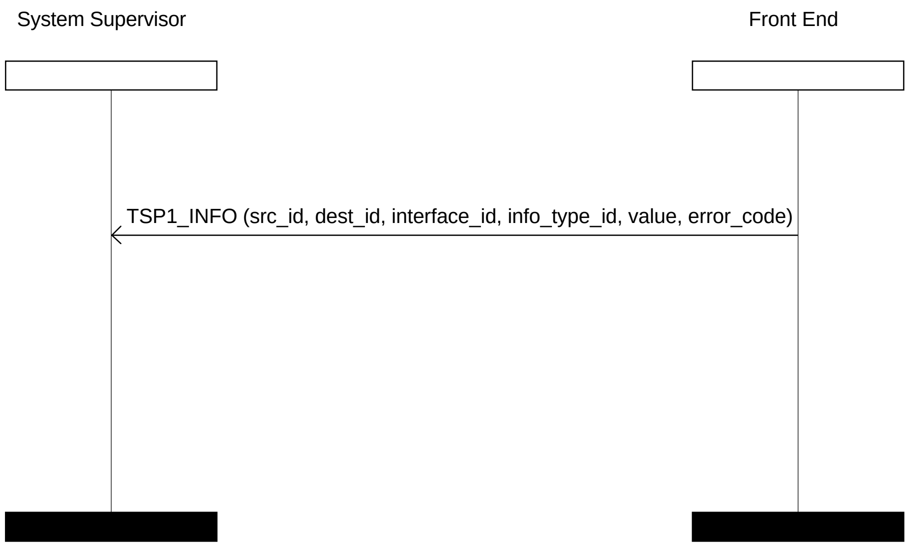

System Supervisor
Front End
TSP1_INFO (src_id, dest_id, interface_id, info_type_id, value, error_code)

Figure 14: Example of normal operation of the TSP1_INFO PDU to send a message from a Front End to the System Supervisor

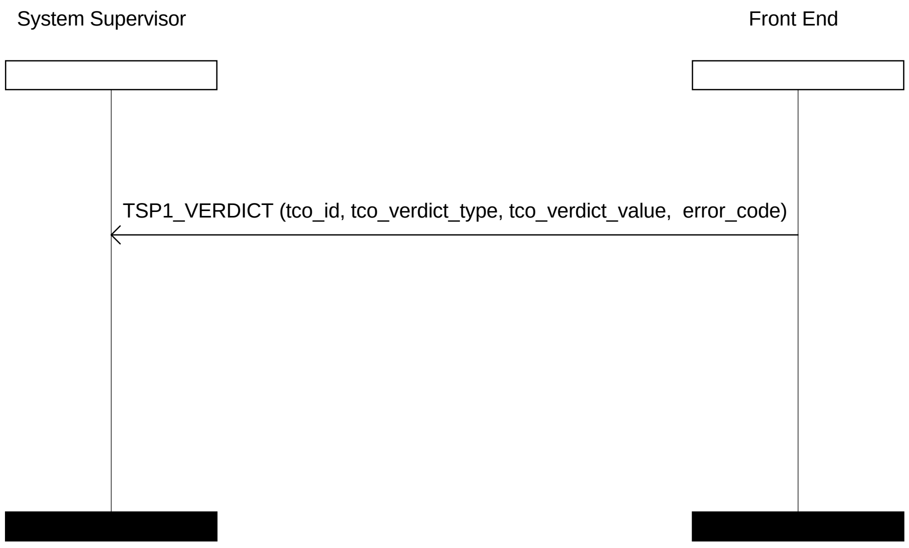

System Supervisor
Front End
TSP1_VERDICT (tco_id, tco_verdict_type, tco_verdict_value, error_code)

Figure 15: Example of normal operation of the TSP1_VERDICT PDU

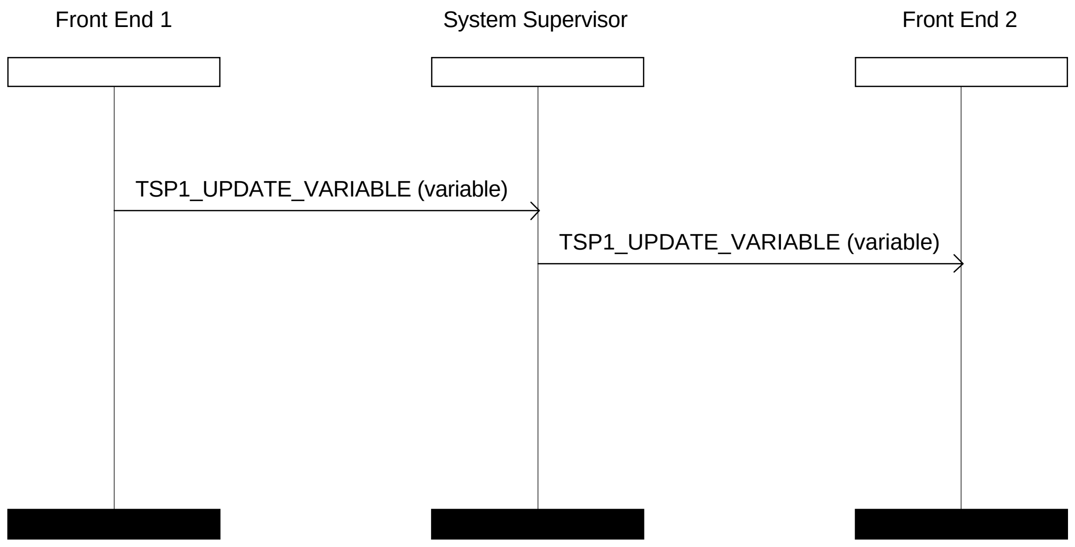

Front End 1
System Supervisor
Front End 2
TSP1_UPDATE_VARIABLE (variable)
TSP1_UPDATE_VARIABLE (variable)

Figure 16: Example of normal operation of the TSP1_UPDATE_VARIABLE PDU

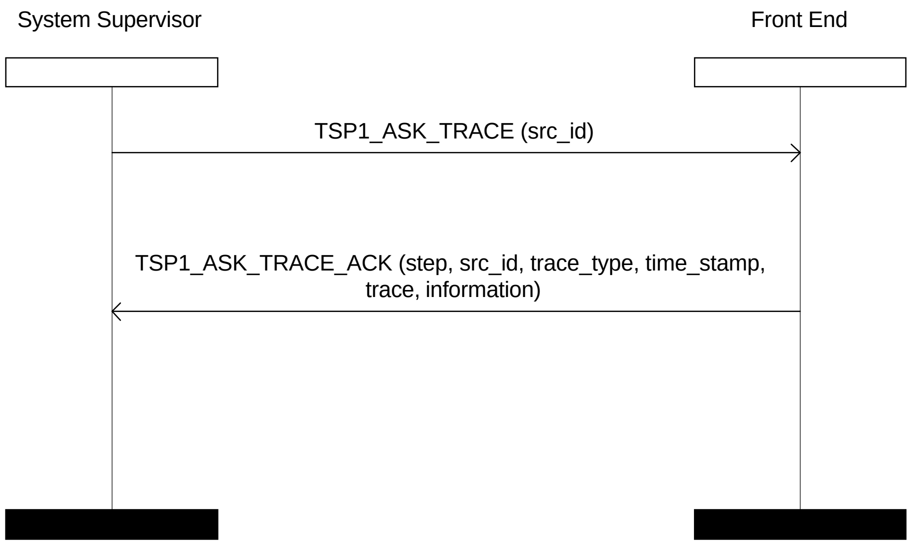

System Supervisor
Front End
TSP1_ASK_TRACE (src_id)
TSP1_ASK_TRACE_ACK (step, src_id, trace_type, time_stamp, trace, information)

Figure 17: Example of normal operation of the TSP1_ASK_TRACE PDU

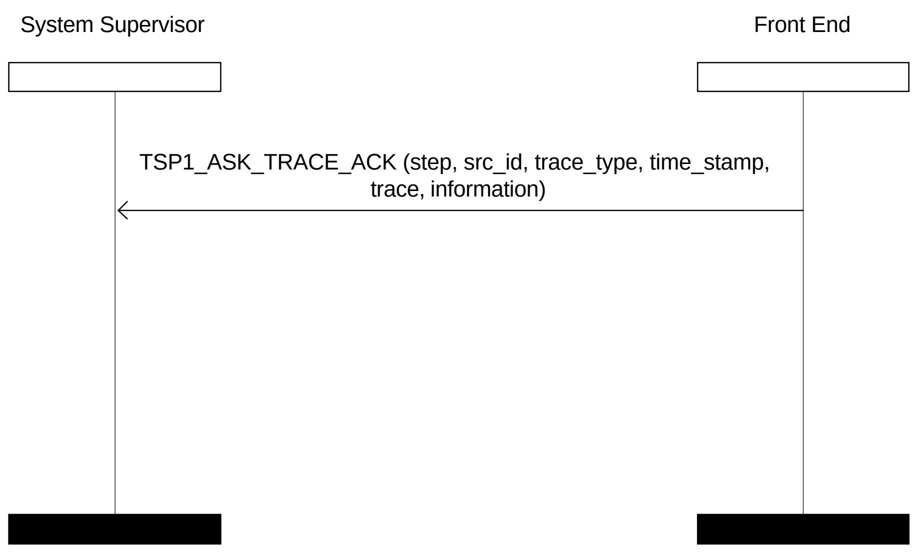

System Supervisor
Front End
TSP1_ASK_TRACE_ACK (step, src_id, trace_type, time_stamp, trace, information)

Figure 18: Example of the use of the TSP1_ASK_TRACE_ACK PDU to provide unsolicited trace information

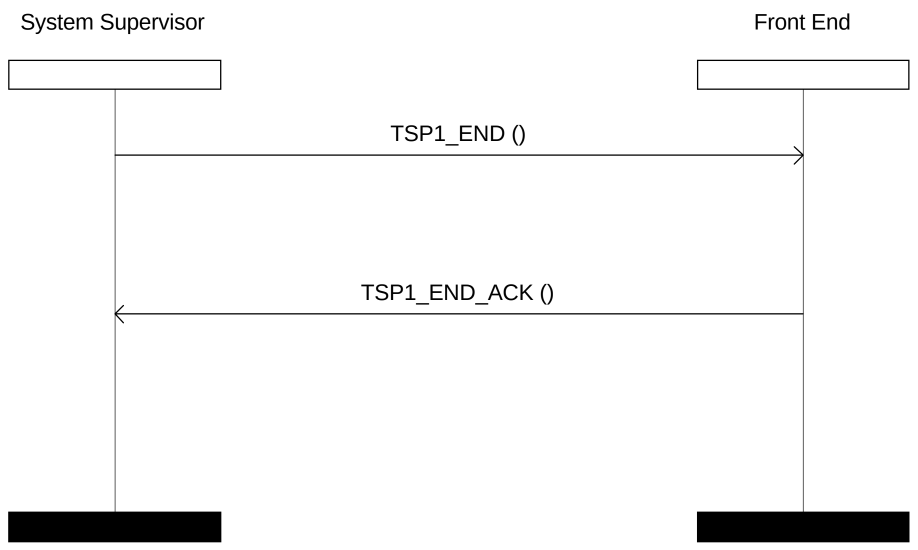


Figure 19: Example of normal operation of the TSP1_END PDU

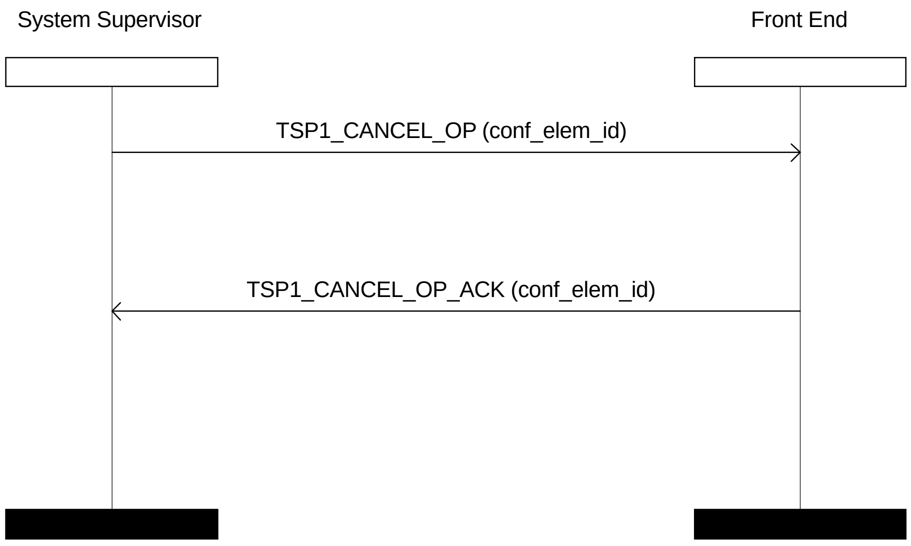

System Supervisor
Front End
TSP1_CANCEL_OP (conf_elem_id)
TSP1_CANCEL_OP_ACK (conf_elem_id)

Figure 20: Example of normal operation of the TSP1_CANCEL_OP PDU

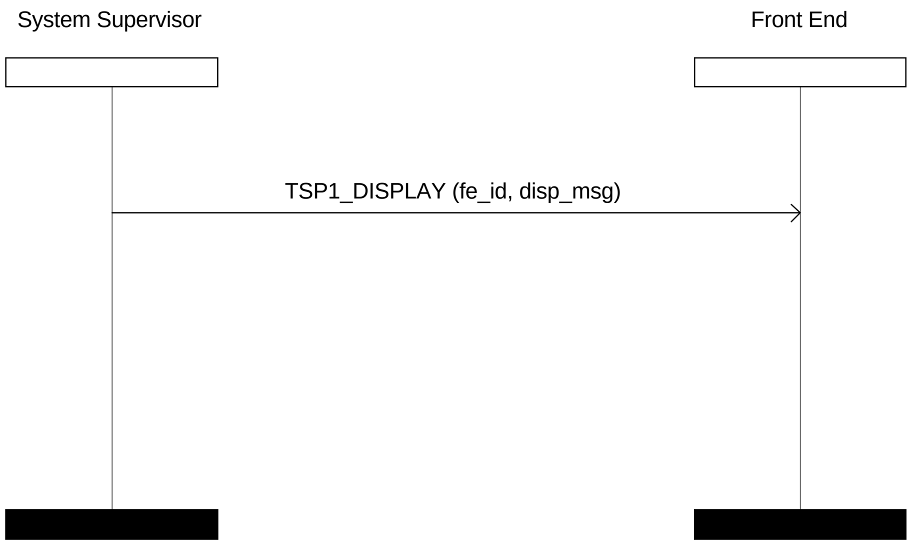


Figure 21: Example of normal operation of the TSP1_DISPLAY PDU to send display data from the System Supervisor to a Front End

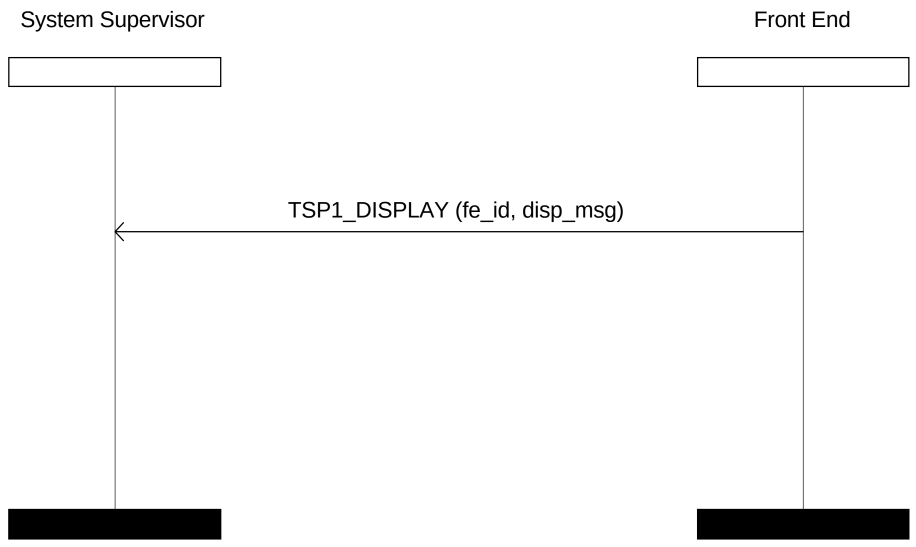


Figure 22: Example of normal operation of the TSP1_DISPLAY PDU to send display data from a Front End to the System Supervisor

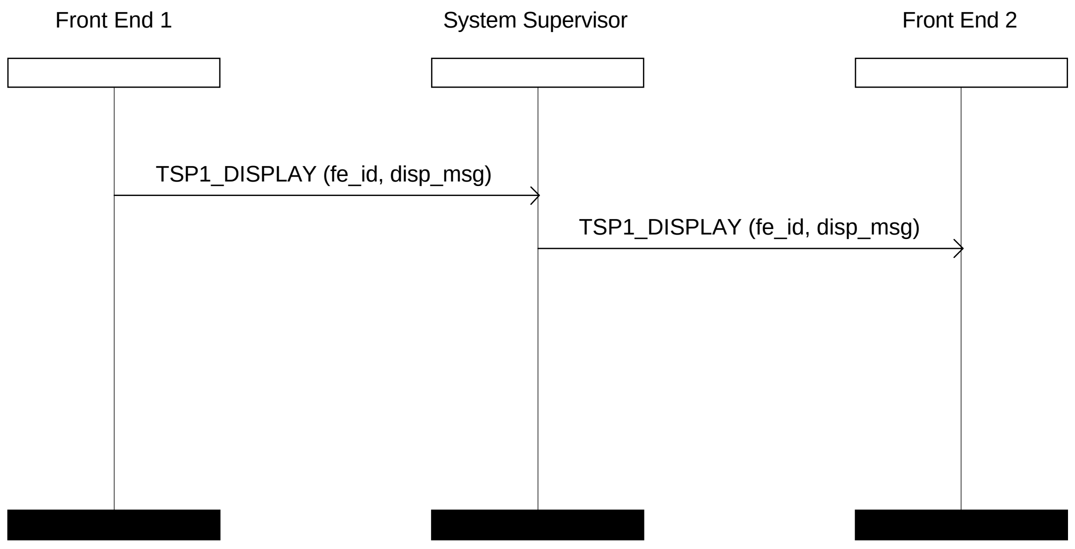

Front End 1
System Supervisor
Front End 2
TSP1_DISPLAY (fe_id, disp_msg)
TSP1_DISPLAY (fe_id, disp_msg)

Figure 23: Example of normal operation of the TSP1_DISPLAY PDU to send display data from one Front End to another

Figure 24 shows an example of how TSP1+ messages can be used together in a full test session.

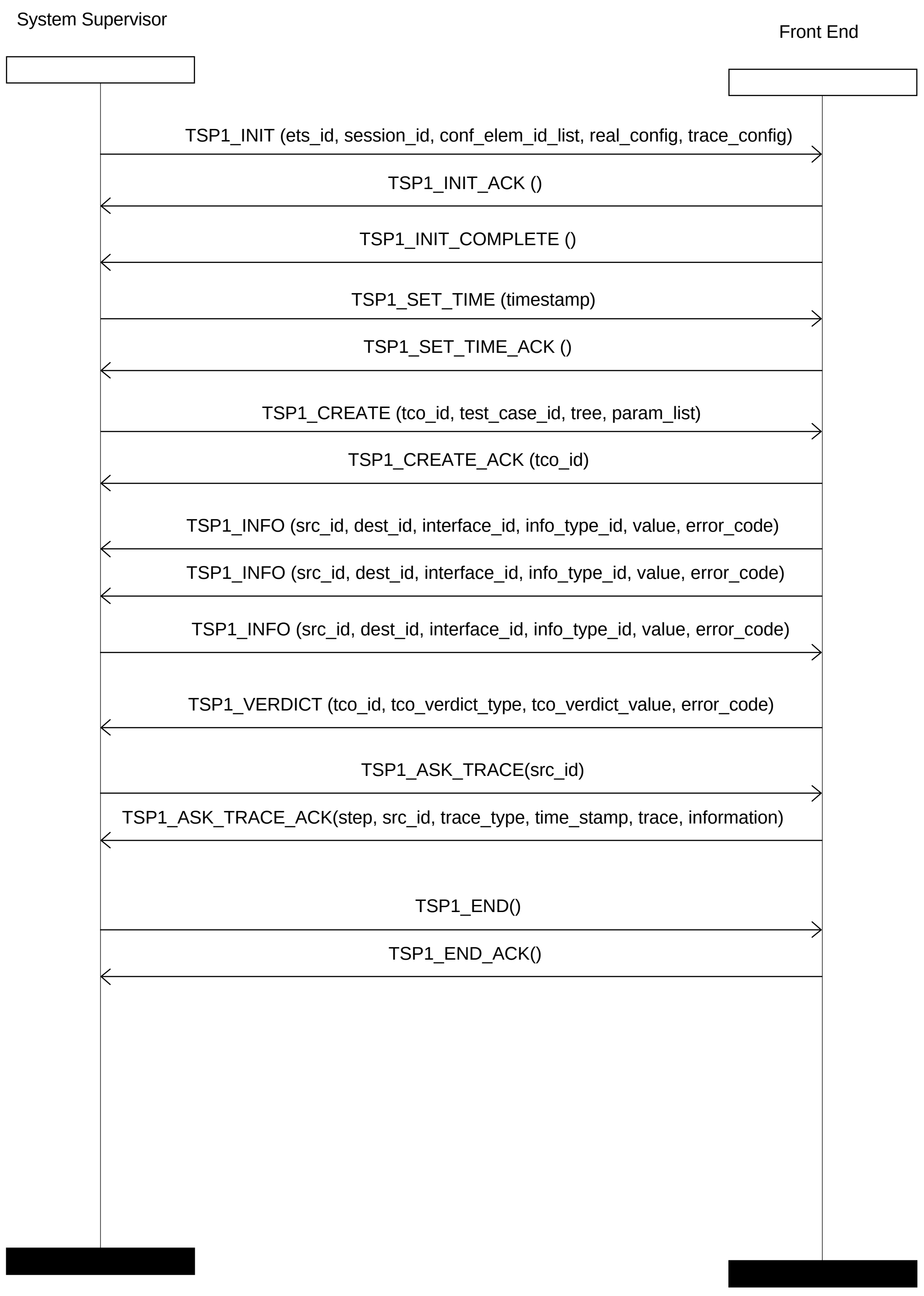

System Supervisor
Front End
TSP1_INIT (ets_id, session_id, conf_elem_id_list, real_config, trace_config)
TSP1_INIT_ACK ()
TSP1_INIT_COMPLETE ()
TSP1_SET_TIME (timestamp)
TSP1_SET_TIME_ACK ()
TSP1_CREATE (tco_id, test_case_id, tree, param_list)
TSP1_CREATE_ACK (tco_id)
TSP1_INFO (src_id, dest_id, interface_id, info_type_id, value, error_code)
TSP1_INFO (src_id, dest_id, interface_id, info_type_id, value, error_code)
TSP1_INFO (src_id, dest_id, interface_id, info_type_id, value, error_code)
TSP1_VERDICT (tco_id, tco_verdict_type, tco_verdict_value, error_code)
TSP1_ASK_TRACE(src_id)
TSP1_ASK_TRACE_ACK(step, src_id, trace_type, time_stamp, trace, information)
TSP1_END()
TSP1_END_ACK()

Figure 24: Example sequence of TSP1+ messages in a full test session

4.4.6 TSP1+ state definitions

4.4.6.1 States at the System Supervisor

The procedures for the System Supervisor are written in terms of the following conceptual states existing within its TSP1+ service control entity.

4.4.6.1.1 Supervisor idle

Ready for an instruction to load a particular Executable Test Suite (ETS).

4.4.6.1.2 Initiating test procedure

Ready for an instruction to begin a particular session of tests

4.4.6.1.3 Executing test session

Ready to receive test status information, trace information or test-related messages from the Front End(s) or an instruction to terminate the test session.

4.4.6.1.4 Test session completed

Ready to receive an instruction to request trace information from a Front End.

Ready to receive instructions to either restart or close the current session.

4.4.6.2 States at a Front End

4.4.6.2.1 Front End idle

Ready to receive instructions from the Supervisor to initiate a test procedure.

4.4.6.2.2 Test procedure requested

Ready to receive timestamp and test parameter settings from the Supervisor.

4.4.6.2.3 Executing test session

Ready to receive requests to report verdicts and traces.

Ready to receive instructions to modify test parameters, to load and start test components or to cancel execution of the current test session.

Ready to report verdicts, traces and received test messages as configured at the start of the session.

4.4.6.2.4 Test session completed

Ready to receive instructions to report final verdicts and traces.

Ready to receive instructions to modify test parameters, to load and start test components or restart or terminate the current test session.

4.4.7 TSP1+ signalling procedures for invocation and operation

The actions at the System Supervisor and Front Ends are specified in the following subclauses in terms of the TSP1+ protocol messages described in subclause 4.4.4.

4.4.7.1 TSP1_INIT

The System Supervisor and any Front Ends shall process the TSP1_INIT and TSP1_INIT_ACK protocol messages according to the procedures described in Table 19.

Table 19: Procedures related to TSP1_INIT and TSP1_INIT_ACK

TSP1_INIT			**TSP1_INIT_ACK**
Actions at the System Supervisor		**Actions at a Front End**	
Normal Procedures	**Exceptional Procedures**	**Normal Procedures**	**Exceptional Procedures**
On receipt of an instruction from the Campaign Management Interface (CMI) to open a specified session of testing, send TSP1_INIT to the front End(s)	On receipt of TSP1_ERROR from any Front End, report error to the CMI and close the current test session.	Send TSP1_INIT_ACK as soon as TSP1_INIT is received. Carry out initialization of attached TCOs and PCOs.	If the ETS specified I the TSP1_INIT message is invalid, send TSP1_ERROR with the error type "Unknown ETS" If the Session identifier specified in the TYSP1_INIT message is valid, send TSP1_ERROR with the error type "Unknown Session"

4.4.7.2 TSP1_INIT_COMPLETE

The System Supervisor and any Front Ends shall process the TSP1_INIT_COMPLETE protocol message according to the procedures described in Table 20.

Table 20: Procedures related to TSP1_INIT_COMPLETE

TSP1_INIT_COMPLETE			
Actions at the System Supervisor		**Actions at a Front End**	
Normal Procedures	**Exceptional Procedures**	**Normal Procedures**	**Exceptional Procedures**
On receipt of TSP1_INIT_COMPLETE from each Front End, report success status to the CMI	On receipt of TSP1_ERROR from any Front End, report error to the CMI and close the current test session.	When all TCOs and PCOs have been successfully initialized, send TSP1_INIT_COMPLETE to the System Supervisor	If any of the TCOs associated with the Front End do not respond within 120s, send TSP1_ERROR with the error type "Means Of Testing (MOT) not ready".

4.4.7.3 TSP1_CHK_CONF

The System Supervisor and any Front Ends shall process the TSP1_CHK_CONF and TSP1_CHK_CONF_ACK protocol messages according to the procedures described in Table 21.

Table 21: Procedures related to TSP1_CHK_CONF and TSP1_CHK_CONF_ACK

TSP1_CHK_CONF			TSP_CHK_CONF_ACK
Actions at the System Supervisor		**Actions at a Front End**	
Normal Procedures	**Exceptional Procedures**	**Normal Procedures**	**Exceptional Procedures**
On receipt of an instruction from the CMI to verify that the correct software is loaded in each Test Configuration Element (TCE), send a TSP1_CHK_CONF message to the Front End(s). On receipt of TSP1_CHK_CONF_ACK from each Front End, report success status to the CMI	On receipt of TSP1_ERROR from any Front End, report error to the CMI and close the current test session.	On receipt of TSP1_CHK_CONF, request status information from each associated TCE. If the responses provided by the TCEs indicate that they are all initialized in the correct session, send TSP1_CHK_CONF_ACK to the System Supervisor	If any of the TCEs fail to respond within 120s, send TSP1_ERROR with the error type "MOT not ready".

4.4.7.4 TSP1_SET_PARAMETER

The System Supervisor and any Front Ends shall process the TSP1_SET_PARAMETER and TSP1_SET_PARAMETER_ACK protocol messages according to the procedures described in Table 22.

Table 22: Procedures related to TSP1_SET_PARAMETER and TSP1_SET_PARAMETER_ACK

TSP1_SET_PARAMETER			TSP1_SET_PARAMETER_ACK
Actions at the System Supervisor		**Actions at a Front End**	
Normal Procedures	**Exceptional Procedures**	**Normal Procedures**	**Exceptional Procedures**
On receipt of an instruction from the CMI to change specific parameter values within the TCOs, send TSP1_SET_PARAMETER to the Front End(s) with details of the parameters to be modified and the values to which they are to be set. On receipt of TSP1_SET_PARAMETER_ACK from each Front End, report success status to the CMI	On receipt of TSP1_ERROR from any Front End, report error to the CMI and close the current test session.	On receipt of TSP1_SET_PARAMETER, send details of parameter changes to each of the associated TCEs. If the responses provided by the TCEs indicate that the parameters have been set successfully, send TSP1_SET_PARAMETER_ACK to the System Supervisor	If any of the TCEs fail to respond within 120s, send TSP1_ERROR with the error type "MOT not ready".

4.4.7.5 TSP1_SET_TIME

The System Supervisor and any Front Ends shall process the TSP1_SET_TIME and TSP1_SET_TIME_ACK protocol messages according to the procedures described in Table 23.

Table 23: Procedures related to TSP1_SET_TIME and TSP1_SET_TIME_ACK

TSP1_SET_TIME			**TSP1_SET_TIME_ACK**
Actions at the System Supervisor		**Actions at a Front End**	
Normal Procedures	**Exceptional Procedures**	**Normal Procedures**	**Exceptional Procedures**
On receipt of an instruction from the CMI to set the time-of-day in each of the TCOs, send TSP1_SET_TIME to the Front End(s) with the time-stamp to be set.	On receipt of TSP1_ERROR from any Front End, report error to the CMI and close the current test session.	On receipt of TSP1_SET_TIME, set the time in the Front End to the value indicated and instruct each of the associated TCOs to do the same. If the responses provided by the TCEs indicate that the time has been set successfully in each one, send TSP1_SET_TIME_ACK to the System Supervisor	If any of the TCEs fail to respond within 120s, send TSP1_ERROR with the error type "MOT not ready".

4.4.7.6 TSP1_LIST_FE_SERVICES

The System Supervisor and any Front Ends shall process the TSP1_LIST_FE_SERVICES and TSP1_LIST_FE_SERVICES_ACK protocol messages according to the procedures described in Table 24 .

Table 24: Procedures related to TSP1_LIST_FE_SERVICES and TSP1_LIST_FE_SERVICES_ACK

TSP1_LIST_FE_SERVICES			**TSP1_LIST_FE_SERVICESACK**
Actions at the System Supervisor		**Actions at a Front End**	
Normal Procedures	**Exceptional Procedures**	**Normal Procedures**	**Exceptional Procedures**
On receipt of an instruction from the CMI to request information on the TSP1+ services implemented by a particular Front End, send TSP1_LIST_FE_SERVICES to the indicated Front End.	If a response is not received from the Front End within 120s, report a timeout error to the CMI and close the current test session.	On receipt of TSP1_LIST_FE_SERVICES, send TSP1_LIST_FE_SERVICES_ACK with an identification of each of the TSP1+ services supported by the Front End	

4.4.7.7 TSP1_CREATE

The System Supervisor and any Front Ends shall process the TSP1_CREATE and TSP1_CREATE_ACK protocol messages according to the procedures described in Table 25.

Table 25: Procedures related to TSP1_CREATE and TSP1_CREATE_ACK

TSP1_CREATE			**TSP1_CREATE_ACK**
Actions at the System Supervisor		**Actions at a Front End**	
Normal Procedures	**Exceptional Procedures**	**Normal Procedures**	**Exceptional Procedures**
On receipt of an instruction from the CMI to load and start a specific TCO, send TSP1_CREATE to the Front End associated with that TCO indicating which test case is to be started and what, if any, parameter values are to be set.	On receipt of TSP1_ERROR from the Front End, report error to the CMI and close the current test session.	On receipt of TSP1_CREATE, send a request to the attached test equipment to load and start the identified TCO. When a positive response is returned from the test equipment indicating that the selected TCO has been loaded and started, send TSP1_CREATE_ACK to the System Supervisor.	If the test equipment fails to respond within 120s, send TSP1_ERROR with the error type "MOT not ready"

4.4.7.8 TSP1_INFO

The System Supervisor and any Front Ends shall process the TSP1_INFO protocol message according to the procedures described in Table 26.

Table 26: Procedures related to TSP1_INFO

TSP1_INFO			
Actions at the System Supervisor		**Actions at a Front End**	
Normal Procedures	**Exceptional Procedures**	**Normal Procedures**	**Exceptional Procedures**
On receipt of TSP1_INFO from a Front End, use the *dest_id* input parameter value to determine the destination of the information carried in the message. Send TSP1_INFO to the Front End associated with the TCE.		On receipt of an instruction from the Test Programming Interface (TPI), send TSP1_INFO to the System Supervisor or to another front end as indicated in the instruction from the TPI. On receipt of TSP1_INFO from another Front End, report the contained information to the TPI.	

4.4.7.9 TSP1_VERDICT

The System Supervisor and any Front Ends shall process the TSP1_VERDICT protocol message according to the procedures described in Table 27.

Table 27: Procedures related to TSP1_VERDICT

TSP1_VERDICT			
Actions at the System Supervisor		**Actions at a Front End**	
Normal Procedures	**Exceptional Procedures**	**Normal Procedures**	**Exceptional Procedures**
On receipt of TSP1_VERDICT from a Front End, report the type of verdict and its value to the CMI. If the verdict type is "Final", wait for a further instruction from the CMI.		On receipt of either a final or an intermediate verdict from the TPI, send TSP1_VERDICT to the System Supervisor indicating the type and value of the verdict If the verdict type is "Final", send instructions to all connected TCEs to terminate any running tests and then wait for further input from the System Supervisor.	

4.4.7.10 TSP1_UPDATE_VARIABLE

The System Supervisor and any Front Ends shall process the TSP1_UPDATE_VARIABLE protocol messages according to the procedures described in Table 28.

Table 28: Procedures related to TSP1_UPDATE_VARIABLE

TSP1_UPDATE_VARIABLE			
Actions at the System Supervisor		**Actions at a Front End**	
Normal Procedures	**Exceptional Procedures**	**Normal Procedures**	**Exceptional Procedures**
On receipt of TSP1_UPDATE_VARIABLE from a Front End, report the revised variable to the CMI. When starting a subsequent session (within the same test campaign), include the revised variable in the parameter list of TSP1_CREATE.		On receipt of an indication from the TPI that a test variable has been modified, send TSP1_UPDATE_VARIABLE to the System Supervisor indicating the identity of the revised variable and its new value.	

4.4.7.11 TSP1_ASK_TRACE

The System Supervisor and any Front Ends shall process the TSP1_ASK_TRACE and TSP1_ASK_TRACE_ACK protocol messages according to the procedures described in Table 29.

Table 29: Procedures related to TSP1_ASK_TRACE and TSP1_ASK_TRACE_ACK

TSP1_ASK_TRACE			**TSP1_ASK_TRACE_ACK**
Actions at the System Supervisor		**Actions at a Front End**	
Normal Procedures	**Exceptional Procedures**	**Normal Procedures**	**Exceptional Procedures**
On receipt of an instruction from the CMI to request trace information from a particular TCO, send TSP1_ASK_TRACE the Front End associated with the TCO	On receipt of TSP1_ERROR from the Front End, report error to the CMI and close the current test session.	On receipt of TSP1_ASK_TRACE from the System Supervisor, send a trace request to the TCO indicated in the input message parameters. On receipt of trace information from a TCO (either as a result of a direct request or according to the criteria established during initialization), send TSP1_ASK_TRACE_ACK tot he System Supervisor	Following a request to a TCO to provide trace information, if the TCO reports that this is not available, send TSP1_ERROR with the error type set to "Trace not available"

4.4.7.12 TSP1_END

The System Supervisor and any Front Ends shall process the TSP1_END and TSP1_END_ACK protocol messages according to the procedures described in Table 30.

Table 30: Procedures related to TSP1_END and TSP1_END_ACK

TSP1_END			**TSP1_END_ACK**
Actions at the System Supervisor		**Actions at a Front End**	
Normal Procedures	**Exceptional Procedures**	**Normal Procedures**	**Exceptional Procedures**
On receipt of an instruction from the CMI to close the current test session, send TSP1_END to the Front End(s).	On receipt of TSP1_ERROR from the Front End, report error to the CMI and close the current test session.	On receipt of TSP1_END from the System Supervisor. Send instructions to all connected TCEs to terminate and close the current test session. If the responses provided by the TCEs indicate that the session has been closed successfully in each one, send TSP1_END_ACK to the System Supervisor	If any of the TCEs fail to respond within 120s, send TSP1_ERROR with the error type "MOT not ready".

4.4.7.13 TSP1_CANCEL_OP

The System Supervisor and any Front Ends shall process the TSP1_CANCEL_OP and TSP1_CANCEL_OP_ACK protocol messages according to the procedures described in Table 31.

Table 31: Procedures related to TSP1_CANCEL_OP and TSP1_CANCEL_OP_ACK

TSP1_CANCEL_OP			**TSP1_CANCEL_OP_ACK**
Actions at the System Supervisor		**Actions at a Front End**	
Normal Procedures	**Exceptional Procedures**	**Normal Procedures**	**Exceptional Procedures**
On receipt of an instruction from the CMI to cease execution of the currently running test, send TSP1_CANCEL_OP to the Front End(s).	On receipt of TSP1_ERROR from the Front End, report error to the CMI and close the current test session.	On receipt of TSP1_CANCEL_OP, send instructions to all connected TCE to terminate execution of the current test.	If any of the TCEs report an error condition during cancellation of the running test, send TSP1_ERROR to the System Supervisor with the error type set to the type reported by the TCE.

4.4.7.14 TSP1_DISPLAY

The System Supervisor and any Front Ends shall process the TSP1_DISPLAY protocol message according to the procedures described in Table 32.

Table 32: Procedures related to TSP1_DISPLAY

TSP1_DISPLAY			
Actions at the System Supervisor		**Actions at a Front End**	
Normal Procedures	**Exceptional Procedures**	**Normal Procedures**	**Exceptional Procedures**
On receipt of an instruction from the CMI to send information to a specific Front End for display purposes, send TSP1_DISPLAY to that Front End. On receipt of TSP1_DISPLAY from a Front End, report the contents of the received message to the CMI		On receipt of TSP1_DISPLAY from the System Supervisor, report the contents of the received message to the Front end Management Interface (FMI). On receipt of an instruction from the FMI to send information to the System Supervisor for display purposes, send TSP1_DISPLAY.	

4.5 TSP1+ SDL description

A detailed description of the TSP1+ protocol using the Specification and Description Language (SDL) can be found in Annex D.

Annex A: TSP1+ ICS proforma

A.1 Guidance for completing the ICS proforma

A.1.1 Purposes and structure

The purpose of this ICS proforma is to provide a mechanism whereby a supplier of an implementation of the requirements defined by ETSI for TSP1+, may provide information about the implementation in a standardized manner.

The proforma is subdivided into subclauses for the following categories of information:

- guidance for completing the proformas;
- identification of the implementation;
- global statement of conformance;

A.1.2 Abbreviations and conventions

The ICS proforma contained in annex A is comprised of information in tabular form in accordance with the guidelines presented in ISO/IEC 9646-7 [3].

Item column

The item column contains a number which identifies the item in the table.

Item description column

The item description column describes in free text each respective item (e.g. parameters, timers, etc). It implicitly means "is <item description> supported by the implementation?".

Status column

The following notations, defined in ISO/IEC 9646-7 [3], are used for the status column:

m	mandatory **-** the capability is required to be supported.
o	optional **-** the capability may be supported or not.
n/a	not applicable **-** in the given context, it is impossible to use the capability.
x	prohibited (excluded) **-** there is a requirement not to use this capability in the given context.
o.i	qualified optional **-** for mutually exclusive or selectable options from a set. "i" is an integer which identifies an unique group of related optional items and the logic of their selection which is defined immediately following the table.
ci	conditional **-** the requirement on the capability ("m", "o", "x" or "n/a") depends on the support of other optional or conditional items. "i" is an integer identifying an unique conditional status expression which is defined immediately following the table.

Support column

The support column shall be filled in by the supplier of the implementation. The following common notations, defined in ISO/IEC 9646-7 [3], are used for the support column:

Y or y	supported by the implementation

N or n	not supported by the implementation
N/A, n/a or **-**	no answer required (allowed only if the status is n/a, directly or after evaluation of a conditional status)

It is also possible to provide a comment to an answer in the space provided at the bottom of the table.

Values allowed column

The values allowed column contains the type, the list, the range, or the length of values allowed. The following notations are used:

- range of values: <min value> .. <max value>

  example: 5 .. 20

- list of values: <value1>, <value2>, ..., <valueN>

  example: 2 ,4 ,6 ,8, 9

  example: '1101'B, '1011'B, '1111'B

  example: '0A'H, '34'H, 2F'H

- list of named values: <name1>(<val1>), <name2>(<val2>), ..., <nameN>(<valN>

  example: reject(1), accept(2)

- length: size (<min size> .. <max size>)

  example: size (1 .. 8)

Values supported column

The values supported column shall be filled in by the supplier of the implementation. In this column, the values or the ranges of values supported by the implementation shall be indicated.

A.1.3 Instructions for completing the ICS proforma

The supplier of the implementation shall complete the ICS proforma in each of the spaces provided. In particular, an explicit answer shall be entered, in each of the support or supported column boxes provided, using the notation described in subclause A.1.2.

If necessary, the supplier may provide additional comments in space at the bottom of the tables, or separately on sheets of paper.

More detailed instructions may be given at the beginning of the different subclauses of the ICS proforma.

A.2 Identification of the TSP1+ implementation

Identification of the TSP1+ Implementation should be filled in so as to provide as much detail as possible regarding version numbers and configuration options.

The product supplier information and client information should both be filled in if they are different.

A person who can answer queries regarding information supplied in the ICS and IXIT should be named as the contact person.

A.2.1 Date of the statement

....

A.2.2 TSP1+ identification

TSP1+ name:

………

………

TSP1+ version:

………

A.2.3 ICS contact person

(A person to contact if there are any queries concerning the content of the ICS or IXIT)

Name:

………

Telephone number:

………

Facsimile number:

………

E-mail address:

………

Additional information:

………

………

………

A.3 Identification of the document

This ICS proforma applies to the TSP1+ standard.

A.4 Global Statement of conformance

Are all mandatory capabilities implemented? (Yes/No) …………

NOTE 1: Answering "No" to this question indicates non-conformance to the TSP1+ specification. Non-supported mandatory capabilities are to be identified in the ICS, with an explanation of why the implementation is non-conforming, on pages attached to the ICS proforma.

A.5 Detailed requirements

A.5.1 General

The supplier of the implementation shall state the support of the roles of the implementation, in the boxes below.

Table A-1: TSP1+ Roles

Item	Description	Reference	Status	Support
A1	Behaviour as a System Supervisor	4.2.1	o.1	
A2	Behaviour as a Front End	4.2.2	o.1	

o.1: It is mandatory to support at least one of these items

A.5.2 Procedures

Table A-2: TSP1+ Procedures

Item	Description	Reference	Status	Support
B1	Signalling procedures at a System Supervisor	4.4.7	A1:m	
B2	Signalling procedures at a Front End	4.4.7	A2:m	

A.5.3 Actions

A.5.3.1 Actions at the System Supervisor

Table A-3: TSP1+ PDU Actions at the System Supervisor

Item	Description	Reference	Status	Support
C1	Receipt of TSP1-ERROR PDU	4.4.5.1	B1:m	
C2	Sending of TSP1-INIT and receipt of TSP1-INIT-ACK PDU	4.4.7.1	B1:m	
C3	Receipt of TSP1-INIT-COMPLETE PDU	4.4.7.2	B1:m	
C4	Sending of TSP1-CHK-CONF PDU and receipt of TSP1-CHK-CONF-ACK PDU	4.4.7.3	B1:m	
C5	Sending of TSP1-SET-PARAMETER PDU and receipt of TSP1-SET-PARAMETER PDU	4.4.7.4	B1:m	
C6	Sending of TSP1-SET-TIME PDU and receipt of TSP1-SET-TIME-ACK PDU	4.4.7.5	B1:m	
C7	Sending of TSP1-LIST-FE-SERVICES PDU and receipt of TSP1-LIST-FE-SERVICES-ACK PDU	4.4.7.6	B1:m	
C8	Sending of TSP1-CREATE PDU and receipt of TSP1-CREATE-ACK PDU	4.4.7.7	B1:m	
C9	Receipt of TSP1-INFO PDU	4.4.7.8	B1:m	
C10	Receipt of TSP1-VERDICT PDU	4.4.7.9	B1:m	
C11	Receipt of TSP1-UPDATE-VARIABLE PDU	4.4.7.10	B1:m	
C12	Sending of TSP1-ASK-TRACE PDU and receipt of TSP1-ASK-TRACE-ACK PDU	4.4.7.11	B1:m	
C13	Sending of TSP1-END PDU and receipt of TSP1-END-ACK PDU	4.4.7.12	B1:m	
C14	Sending of TSP1-CANCEL-OP PDU and receipt of TSP1-CANCEL-OP-ACK PDU	4.4.7.13	B1:m	
C15	Sending of TSP1-DISPLAY PDU	4.4.7.14	B1:m	
C16	Receipt of TSP1-DISPLAY PDU	4.4.7.14	B1:m	

A.5.3.2 Actions at a Front End

Table A-4: TSP1+ PDU Actions at a Front End

Item	Description	Reference	Status	Support
D1	Sending of TSP1-ERROR PDU	4.4.5.1	B2:m	
D2	Receipt of TSP1-INIT and sending of TSP1-INIT-ACK PDU	4.4.7.1	B2:m	
D3	Sending of TSP1-INIT-COMPLETE PDU	4.4.7.2	B2:m	
D4	Receipt of TSP1-CHK-CONF PDU and sending of TSP1-CHK-CONF-ACK PDU	4.4.7.3	B2:m	
D5	Receipt of TSP1-SET-PARAMETER PDU and sending of TSP1-SET-PARAMETER PDU	4.4.7.4	B2:m	
D6	Receipt of TSP1-SET-TIME PDU and sending of TSP1-SET-TIME-ACK PDU	4.4.7.5	B2:m	
D7	Receipt of TSP1-LIST-FE-SERVICES PDU and sending of TSP1-LIST-FE-SERVICES-ACK PDU	4.4.7.6	B2:m	
D8	Receipt of TSP1-CREATE PDU and sending of TSP1-CREATE-ACK PDU	4.4.7.7	B2:m	
D9	Sending of TSP1-INFO PDU	4.4.7.8	B2:m	
D10	Sending of TSP1-VERDICT PDU	4.4.7.9	B2:m	
D11	Sending of TSP1-UPDATE-VARIABLE PDU	4.4.7.10	B2:m	
D12	Receipt of TSP1-ASK-TRACE PDU and sending of TSP1-ASK-TRACE-ACK PDU	4.4.7.11	B2:m	
D13	Receipt of TSP1-END PDU and sending of TSP1-END-ACK PDU	4.4.7.12	B2:m	
D14	Receipt of TSP1-CANCEL-OP PDU and sending of TSP1-CANCEL-OP-ACK PDU	4.4.7.13	B2:m	
D15	Receipt of TSP1-DISPLAY PDU	4.4.7.14	B2:m	
D16	Sending of TSP1-DISPLAY PDU	4.4.7.14	B2:m	

A.5.4 Coding

A.5.4.1 Coding of PDUs at the System Supervisor

Table A-5: TSP1+ PDU Coding at the System Supervisor

Item	Description	Reference	Status	Support
E1	TSP1-ERROR	4.4.3	B1:m	
E2	TSP1-INIT	4.4.3	B1:m	
E3	TSP1-INIT-ACK	4.4.3	B1:m	
E4	TSP1-INIT-COMPLETE	4.4.3	B1:m	
E5	TSP1-CHK-CONF	4.4.3	B1:m	
E6	TSP1-CHK-CONF-ACK	4.4.3	B1:m	
E7	TSP1-SET-PARAMETER	4.4.3	B1:m	
E8	TSP1-SET-PARAMETER	4.4.3	B1:m	
E9	TSP1-SET-TIME	4.4.3	B1:m	
E10	TSP1-SET-TIME-ACK	4.4.3	B1:m	
E11	TSP1-LIST-FE-SERVICES	4.4.3	B1:m	
E12	TSP1-LIST-FE-SERVICES-ACK	4.4.3	B1:m	
E13	TSP1-CREATE	4.4.3	B1:m	
E14	TSP1-CREATE-ACK	4.4.3	B1:m	
E15	TSP1-INFO	4.4.3	B1:m	
E16	TSP1-VERDICT	4.4.3	B1:m	
E17	TSP1-UPDATE-VARIABLE	4.4.3	B1:m	
E18	TSP1-ASK-TRACE	4.4.3	B1:m	
E19	TSP1-ASK-TRACE-ACK	4.4.3	B1:m	
E20	TSP1-END	4.4.3	B1:m	
E21	TSP1-END-ACK	4.4.3	B1:m	
E22	TSP1-CANCEL-OP	4.4.3	B1:m	
E23	TSP1-CANCEL-OP-ACK	4.4.3	B1:m	
E24	TSP1-DISPLAY	4.4.3	B1:m	

A.5.4.2 Coding of PDUs at a Front End

Table A-6: TSP1+ PDU coding at a Front End

Item	Description	Reference	Status	Support
F1	TSP1-ERROR	4.4.3	B2:m	
F2	TSP1-INIT	4.4.3	B2:m	
F3	TSP1-INIT-ACK	4.4.3	B2:m	
F4	TSP1-INIT-COMPLETE	4.4.3	B2:m	
F5	TSP1-CHK-CONF	4.4.3	B2:m	
F6	TSP1-CHK-CONF-ACK	4.4.3	B2:m	
F7	TSP1-SET-PARAMETER	4.4.3	B2:m	
F8	TSP1-SET-PARAMETER	4.4.3	B2:m	
F9	TSP1-SET-TIME	4.4.3	B2:m	
F10	TSP1-SET-TIME-ACK	4.4.3	B2:m	
F11	TSP1-LIST-FE-SERVICES	4.4.3	B2:m	
F12	TSP1-LIST-FE-SERVICES-ACK	4.4.3	B2:m	
F13	TSP1-CREATE	4.4.3	B2:m	
F14	TSP1-CREATE-ACK	4.4.3	B2:m	
F15	TSP1-INFO	4.4.3	B2:m	
F16	TSP1-VERDICT	4.4.3	B2:m	
F17	TSP1-UPDATE-VARIABLE	4.4.3	B2:m	
F18	TSP1-ASK-TRACE	4.4.3	B2:m	
F19	TSP1-ASK-TRACE-ACK	4.4.3	B2:m	
F20	TSP1-END	4.4.3	B2:m	
F21	TSP1-END-ACK	4.4.3	B2:m	
F22	TSP1-CANCEL-OP	4.4.3	B2:m	
F23	TSP1-CANCEL-OP-ACK	4.4.3	B2:m	
F24	TSP1-DISPLAY	4.4.3	B2:m	

Annex B (informative): Test Synchronization Architecture

B.1 Introduction

Figure B-1 shows how the Test Synchronization Architecture (TSA) is created by inserting a middle layer functional entity called the Front End (FE) into the Multi-Party Testing Method (MPTM). In fact, all the configurations necessary to simultaneously check several interfaces can be realized with Multi-Party Testing. The System Supervisor has the function of an LTCF, and each Test Component is an LT. The use of FEs is a way of solving the communication problems which the Tester cannot solve and of giving the System Supervisor a homogeneous and logical view of the testers in terms of test configuration elements (test components and Points of Control and Observation).

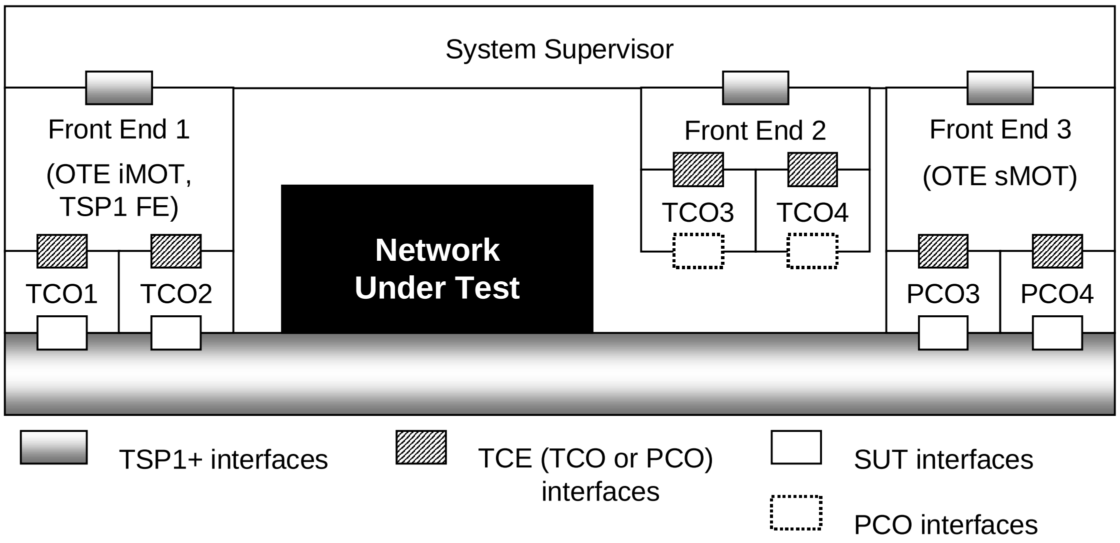


Figure B-1: Test Synchronization Architecture

The addition of a Front End in the TS Architecture, introduces the concept of a "virtual tester". The virtual tester concept means that each Front End gives the System Supervisor a homogeneous view of the controlled tester. In this way the System Supervisor can manage every tester as a set of generic test configuration elements without influence of the tester machine manufacturer.

An example of the Test Synchronization Architecture applied to Network Integration Testing is shown in Figure B-2. In this example, there are various groups of geographically distributed Protocol Testers (PT), each of which is controlled by a FE that is able to communicate with a System Supervisor by means of a high-level protocol. Each FE can control various local PTs with a simple proprietary protocol between FE and PT.

Future protocol testers can be directly interfaced with a supervisor using TSP1+ with an embedded TSP1+ front end function.

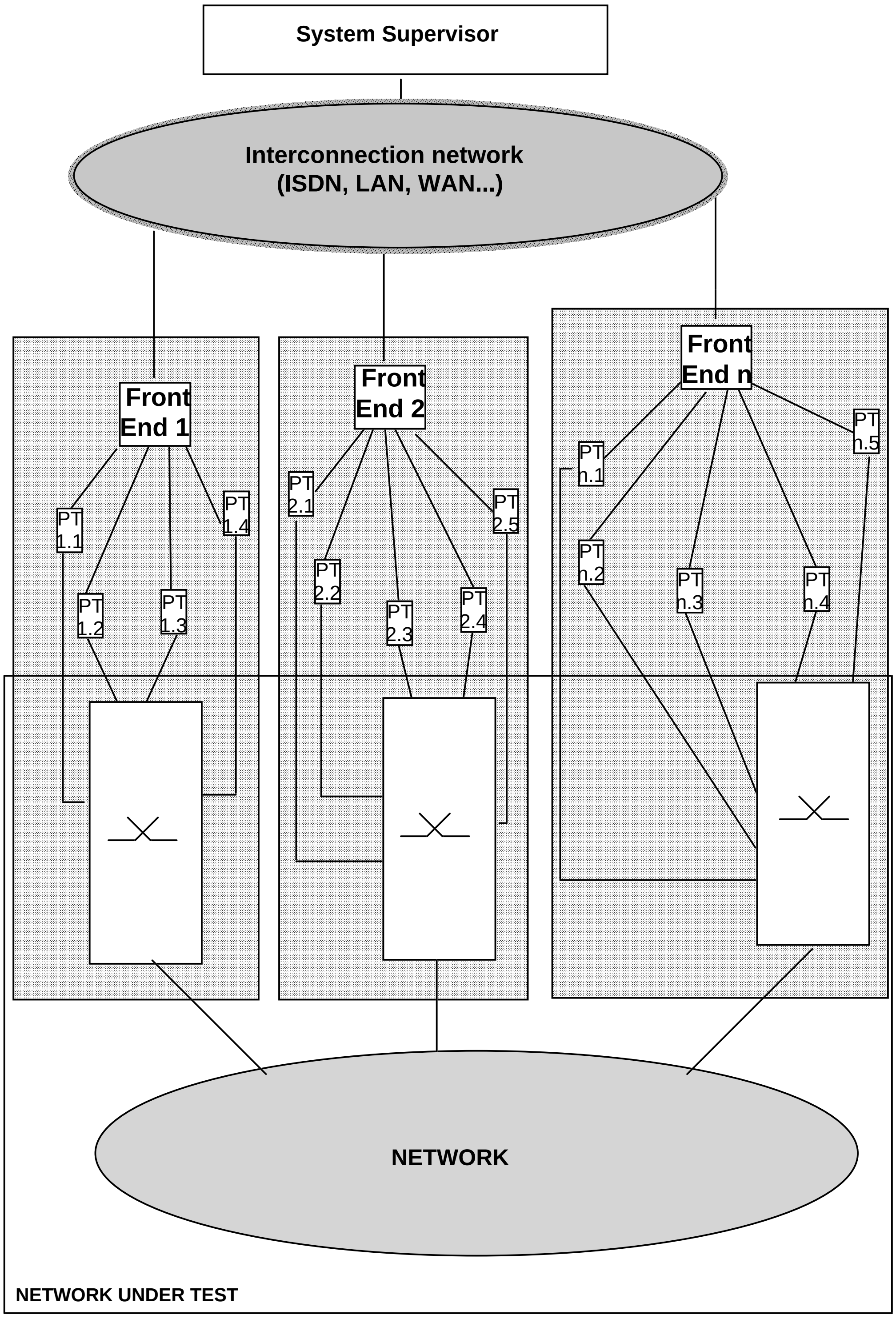


Figure B-2: Test Synchronization Architecture applied on NIT

Note: The System Supervisor is not a distributed system, but it is located only in one place.

The functionality of the different components are:

- System Supervisor (SS)

  manages the test execution. It does not provide any support to implement the necessary configurations on physical machines such as the tester or the IUT (as those configuration can be obtained using TMN or a manual approach). The test configuration need to be known, well identified and set up before starting the test campaign.

- Front End (FE)
  - translates system supervisor messages to the appropriate format for each physical tester. Messages between two configuration elements handled by the same Front End are not sent to the system supervisor. In other cases messages are sent to the system supervisor that routes them to the appropriate Front End.
- Test Component (TCO)

  controls the execution of part of a test case behaviour.
- Point of Control and Observation (PCO)

  controls the operation of test interfaces (between the test component and the System Under Test).

The communication between those components may be represented as a layered protocol model as shown in Figure B-3.

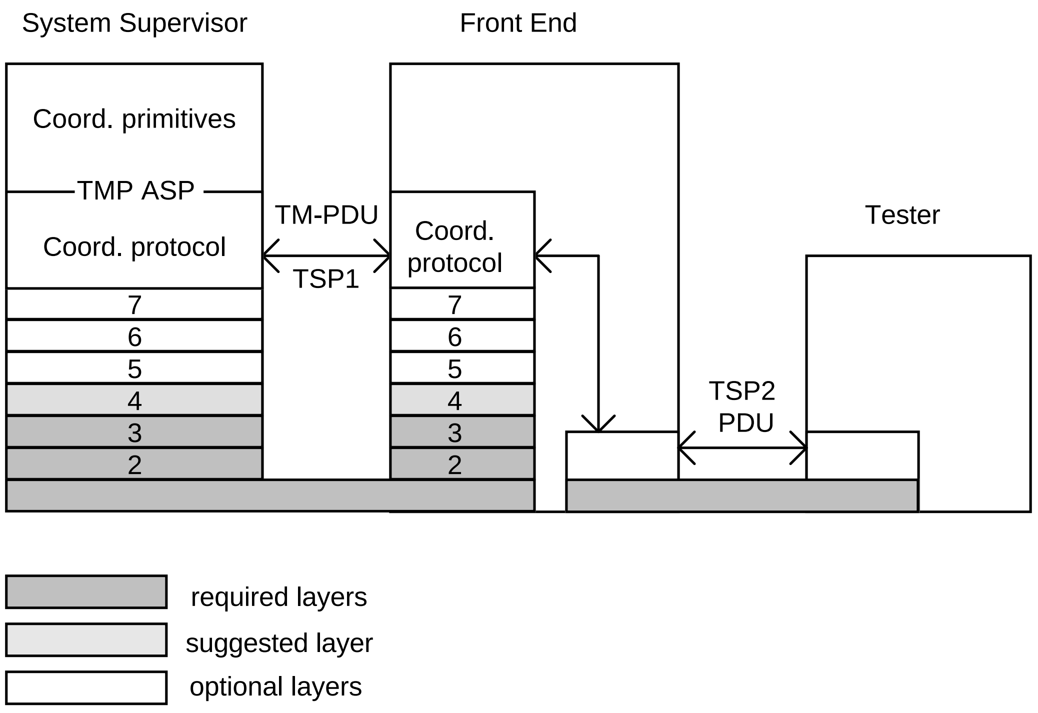


Figure B-3: TSP protocols

The co-ordination messages use the services provided by the OSI stack at any layer from the 3rd up to the 7th layer. They can be transported by any transport mechanism.

The functional architecture of the co-ordination services is shown in the upper part of Figure B-4. Some possible alternative solutions for the OSI stack 1 to 7 are shown in the lower part

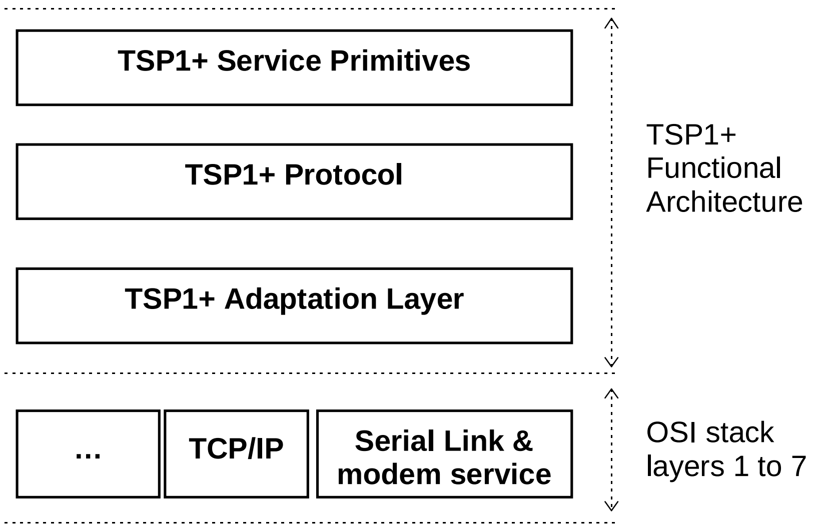


Figure B-4: TSP1+ functional architecture

Figure B-3 shows the structure of the connection between TSAEs, and the protocols used to connect them. What that structure can solve from the implementation point of view is the transport of the Test Co-ordination Procedures (TCPs) which are defined in a generic C-TTCN ATS. TCP could be carried using a protocol (TSP1+ in the figure) that is supported by the System Supervisor and each Front End. TSP1+ can use the services of a number of protocols in order to carry the information between SS and FE. The choice of the protocol used below, depends on the network that is used for the transport of the synchronization information. There will be a protocol stack for each type of network (e.g. TCP/IP for Internet).

TSP1+ messages decoded by a Front End are sent to a PT using a protocol (TSP2 in Figure B-3)which may be different for each PT.

The structure in Figure B-2 does not indicate that the System Supervisor has to be remote from each Front End. In fact, the system supervisor can be either in the same machine as one front end or in a different machine but in the same place as Front End. This would allow a "test island" to act only as a front end in one instance and as system supervisor plus front end in another.

Another feature is that with this three level architecture, different protocol testers from different manufacturers can be controlled with the same TSP1+ set of messages. The Front End converts TSP1+ signalling to the appropriate proprietary TSP2 messages toward PTs.

The tester in Figure B-3 can be either a sMOT able only to send and receive PDUs/ASPs or a real tester machine able to manage a complete test component. In the former case, TSP1+ is able to carry PDUs/ASPs which have to be sent or received through PCOs.

Annex C (Informative): TSP1+ Objects Models

The supplementary specifications of the TSP1+ protocol in this annex use the object oriented Object Modelling Technique (OMT) to describe the relationships between the elements of TSP1+ using OMT object models.

For readers unfamiliar with OMT graphical notation, a brief description in natural language follows each model.

The object models that follow describe a TSP1+ architecture from different complementary points of view (using OMT notation [6]):

- Domain analysis models: TTCN models (TTCN declaration model, execution model);
- architectural point of view: system model;
- interaction with transport layer: adaptation layer model;
- process point of view: process model
- service primitives point of view: interface model
- error handling point of view: errors model

C.1 Domain analysis models

These two models introduce all the TTCN concepts, which are in the focus of this specification. They present the classes and objects of the *problem domains*.

The other models have been produced, starting from these two models, with the aim of providing a distributed architecture for executing test campaign based on a concurrent TTCN Abstract Test Suite.

This domain analysis gives an answer to the following questions:

1. what is a TTCN test suite made of, which are its elements, how are they linked together? The answer to these questions can be found in TTCN declaration model (fig 5).
2. During the execution, how do all the previous test suite elements behave? Do they link differently? Do they produce new objects? The answer to these questions can be found in TTCN execution model (fig. 6)

C.1.1 TTCN *declaration* part

A concurrent TTCN abstract test suite specifies the following elements (which can be used to derive an ETS and execute it):

- Test suite structure declaration (which is made of test groups and test cases organized in a directory structure way);
- Test components configuration declaration (each configuration is made of test components, points of control and observation, and co-ordination points);
- Message types declaration (types of message that can be sent/received through a test interface: co-ordination message type for the co-ordination points and ASP/PDU type for the points of control and observation);
- Test suite parameters declaration (their values allow to select and parameterize all the tests to be executed);
- Test suite variables declaration (which allow to share among the different Main Test Components some global values all along the different test sessions of the test campaign).

C.1.2 TTCN *execution* part

During the derivation phase, an executable test suite is produced.

During the execution phase:

- Test suite parameters and test suite variables receive values;
- Behaviour trees are attached to each test component of the configuration used by a test case;
- During their evaluation, each behaviour tree describes, snapshot by snapshot, the different sequences of test events expected on each test interface and the verdict brought by the component for all of these sequences.

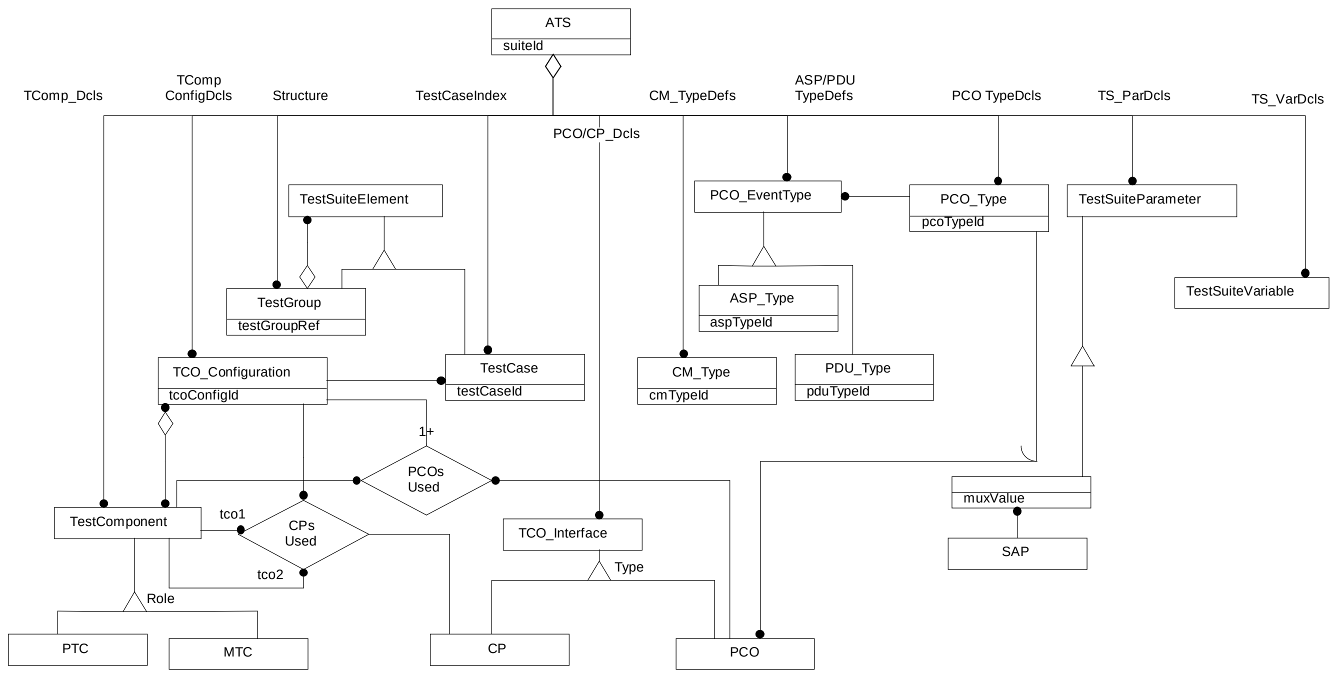


Figure C-1: Data declaration in TTCN model

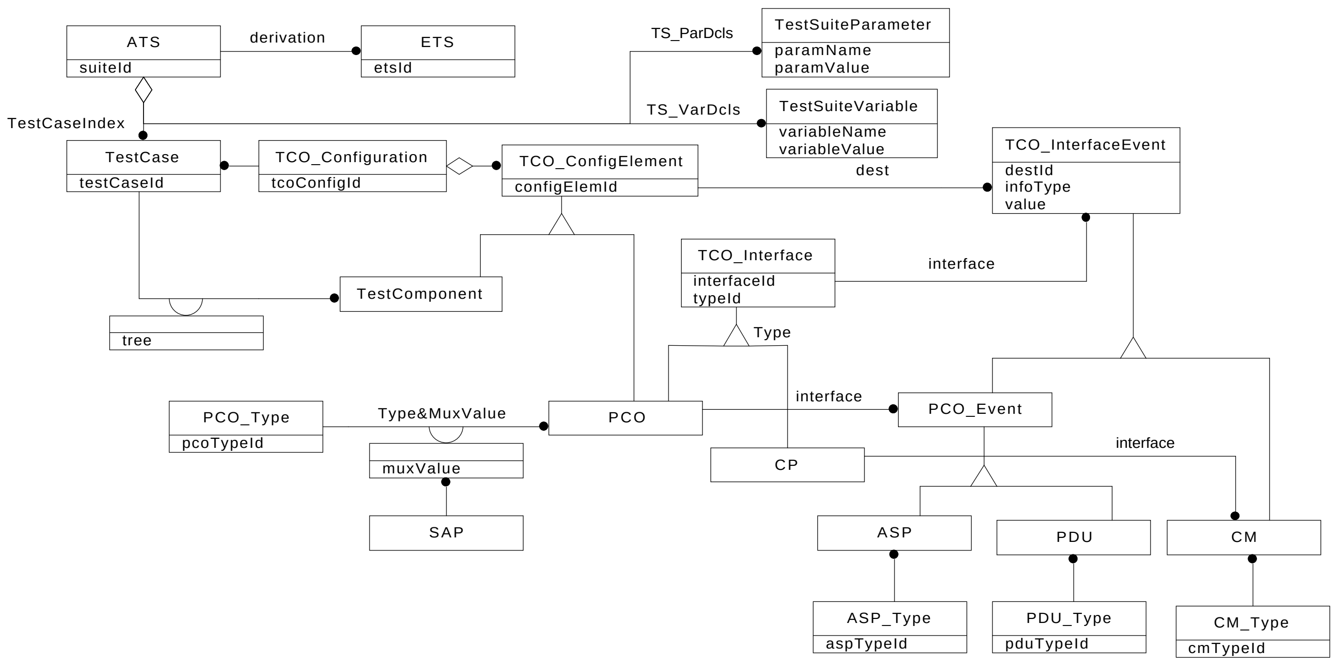


Figure C-2: Execution part in TTCN model

C.2 System model

This model is the first one dealing with the architectural design. It presents the different hardware components of such an architecture.

A TSP1+ system (Test Synchronization Architecture) is made of resources (Test Synchronization Architectural Elements) which are physical hardware machines linked up together with some connection oriented network.

A resource can assume two different roles:

1. Supervisor role:
   unique and global controlling element of the platform;

2. Front End role:
   the mediator to one or several real protocol testers.

Any kind of connection can be used between a supervisor and its front ends. These connections can be of different kinds at one time.

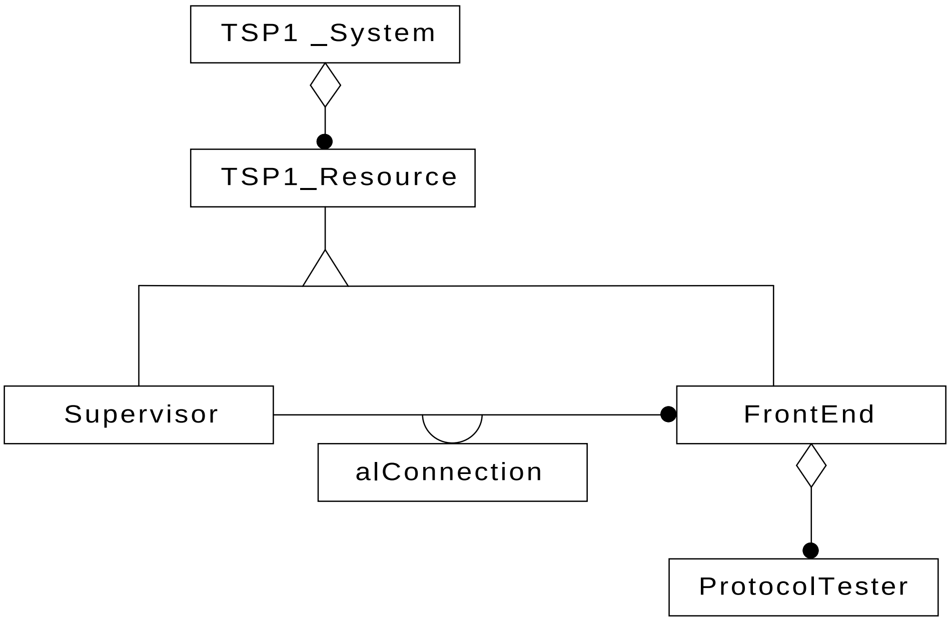


Figure C-3: System Model

During the first design phases, one level of abstraction has been introduced in order to manage any kind of connections in one specified common way and to make the TSP1+ specification independent of how the real connection are established, used and released.

The result of this choice is a (small) formal specification of an underlying layer called Adaptation Layer. Each real transport mechanism implementation will have to follow the external behaviour specified for this layer.

C.3 Adaptation Layer model

The aim of this layer is to manage different kind of connections (establish, use, release and manage a connection).

A client system for this layer (the layer above) contains different entities, which can have two roles according to its behaviour during the establishment phase:

- Connector role (who is initiating a connection: the "calling party"),
- Acceptor role (who is receiving a connection: the "called party").

Once the connection is established there is no more behaviour differences between each side of the connection (the common behaviour is factorised in the Side class).

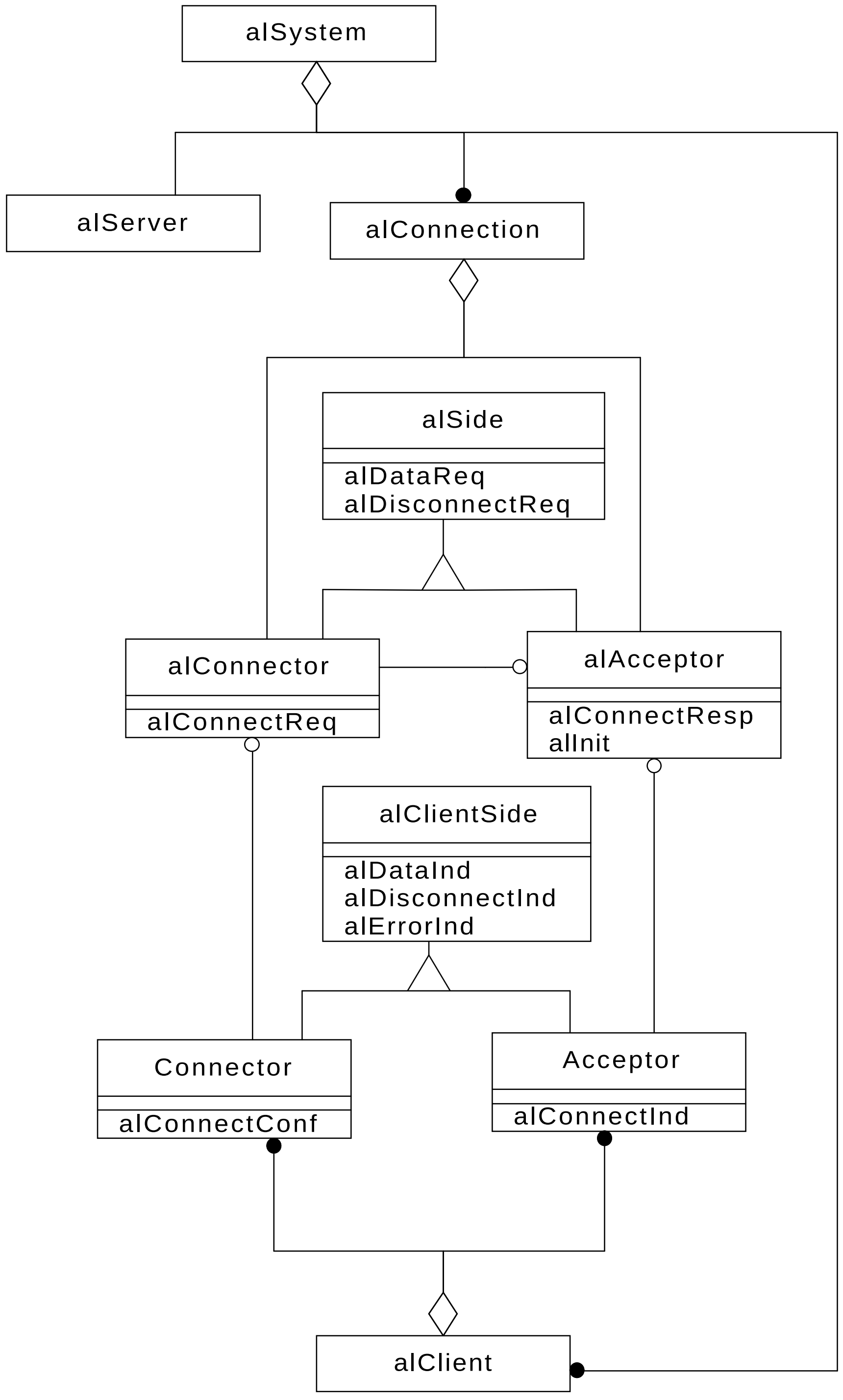


Figure C-4: Adaptation Layer Model

After the presentation of the TSP1+ process model, the architecture of the adaptation layer will be mapped to the TSP1+ architecture, in order to show how the TSP1+ layer can be a client of the adaptation layer (how this two layered architecture is built).

C.4 TSP1+ system as an adaptation layer client

Here is the way a TSP1+ system is "plugged" as an adaptation layer client:

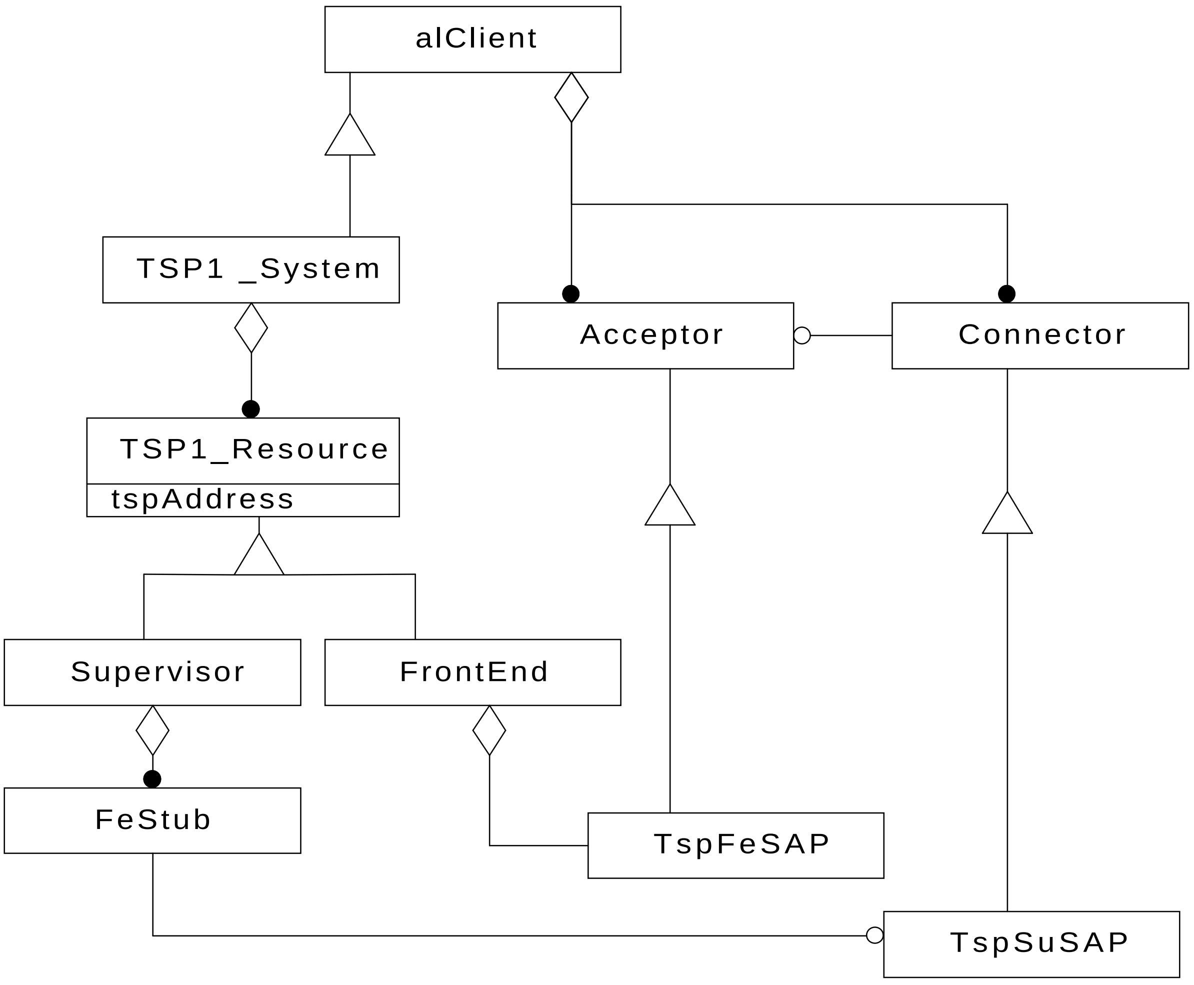


Figure C-5: TSP1+ system as an Adaptation Layer Client

As said before, the TSP1+ SAP are used to interface the system with this adaptation layer:

- Supervisor SAP takes the role of connector;
- Front end SAP takes the role of acceptor.

When using TSP1+ low level routeing capabilities, a TSP1+ address is generated by the TSP1+ Supervisor SAP for each TSP1+ connection. This address is then used by the TSP1+ SAP to fill in or interpret the src-id and dest-id fields of the TSP1+ envelope, and to route the embedded PDU.

C.5 Process model

The TSP1+ architecture now allows distributing the elements of the *TTCN machine* all over the test platform:

- TTCN Test Components (MTC/PTC),
- Point of Control and Observations (PCO).

The TTCN machine is made of one MTC, zero or several PTC, zero or several interfaces (PCO or CP).

During a test session, a front end can manage the MTC, one or more PTC and one or more PCO: MTC, PTC, and PCO are the TTCN elements which can be freely mapped on the desired front end.

On the front-end side, the used TTCN machine is more complicated because it contains both local and distant configuration elements (stubs representing elements managed by another front end).

As a controlling element, the supervisor needs to exchange TSP1+ PDU with all the front ends involved in a session. Its dialog contexts with all the required front ends is managed by the class FeStub (in fact, FeStub is a kind of connection, but at the TSP1+ level).

Each FeStub is in charge of one (and only one) FrontEnd. To communicate with it, it relies on the TSP1+ Supervisor Service Access Point (TspSuSAP). Each FeStub exchanges data trough TspSuSAP using TSP1+ Supervisor Abstract Service Primitive (TspSuASP).

On the other TSP1+ connection side, each front end uses its own TSP1+ Front End Service Access Point (TspFeSAP) and exchange with it TSP1+ Front End Abstract Service Primitive (TspFeASP).

Here are the complete responsibilities of the TSP1+ SAP:

- Converting TSP1+ request and response into its corresponding PDU (encoding TSP1+ services);
- Converting each received PDU into its corresponding indication and confirmation (decoding TSP1+ services);
- Assuring TSP1+ low level routeing capabilities (this feature allows the supervisor to route the enveloped PDU just reading the envelope and without having to decode the TSP1+ ASN.1 part of the message);
- Interfacing the system with the adaptation layer.

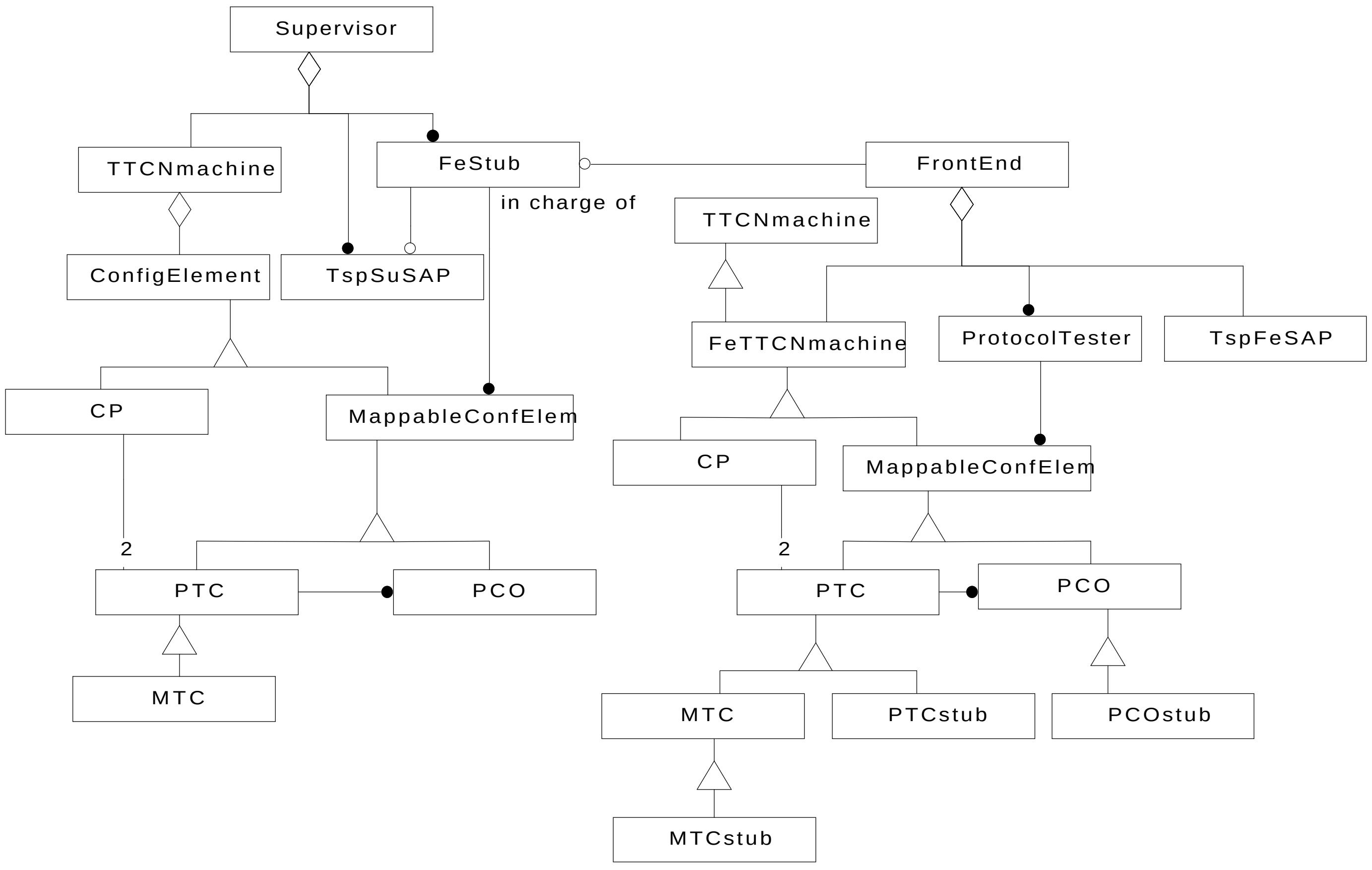


Figure C-6: Process Model

C.6 Interface model

This model introduces all the TSP1+ high level services brought by the TSP1+ architecture. All these services have been attached as operations of the class which is in charge of them.

Interface model introduces all the classes and relationships necessary for a TSP1+ test execution. All the TSP1+ abstract service primitives are attached to their corresponding classes as operation of these classes.

Most of these classes and relationships come from the previous model:

- From TTCN model:
    - ATS, ETS, TestCase, TestSuiteVariable,
    - TCO_Configuration, TCO_ConfigElement, TestComponent, MTC, TCO_Interface (PCO or CP).
- From TSP1+ system model:
    - FrontEnd

New classes introduced by this model:

- A TSP1+ campaign deals with one TTCN ETS.
- A TSP1+ campaign is composed of several test sessions. Selected test cases for a particular test session share the same abstract and real test configurations.
- A real (test) configuration is composed by choosing which front end will be in charge of handling each TTCN test configuration element. (This is modelled by the ternary association "mapping").
- A test (component) trace is identified by its test session, its test case and the test configuration element which produced it. (this is modelled by the ternary association class "Trace").
- A trace is made of several (test) event sent by one element of the configuration.

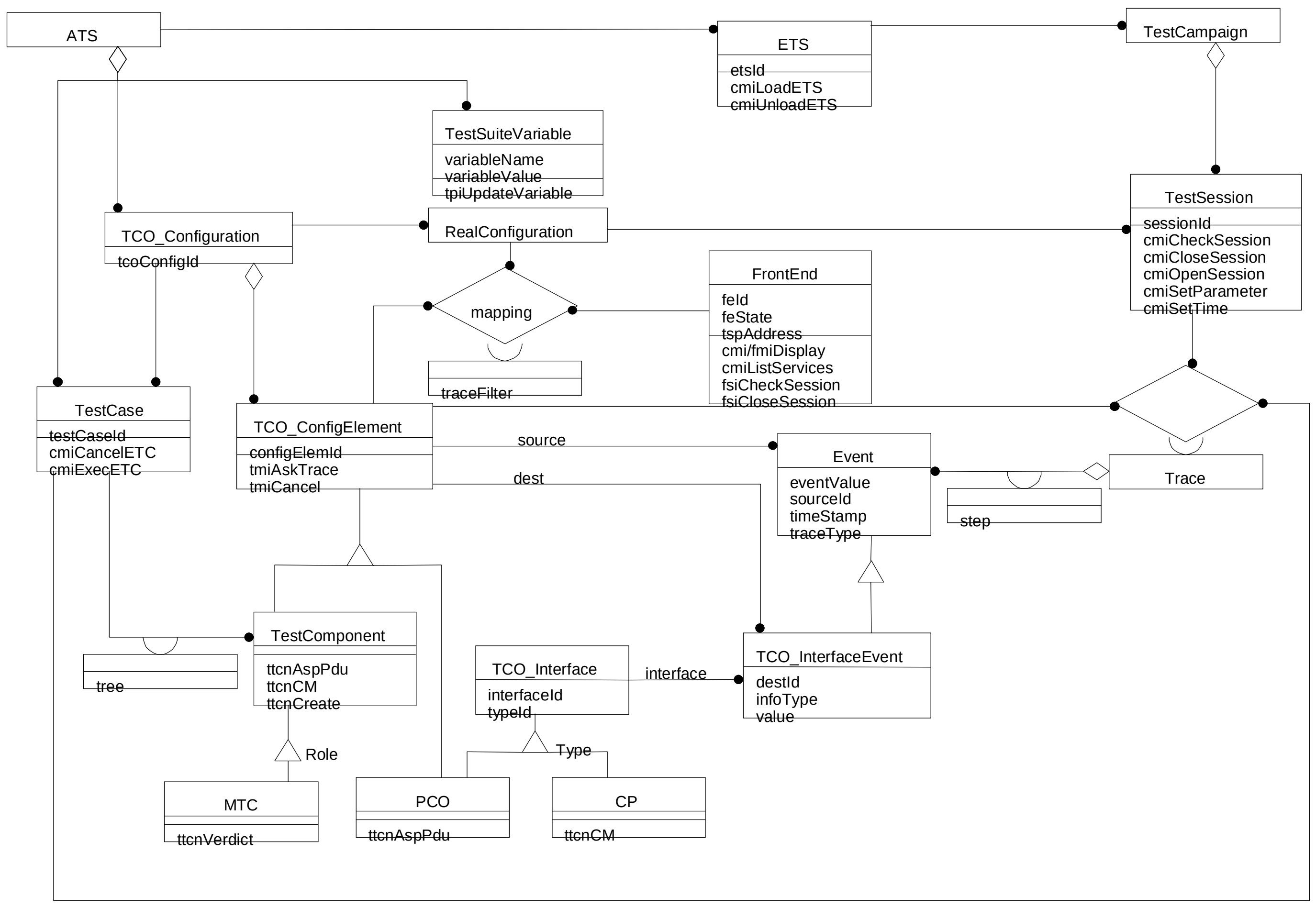


Figure C-7: Interface Model

C.7 Errors model

Error data model is presenting the way the errors are collected and linked to be globally signalled to and managed by the supervisor side.

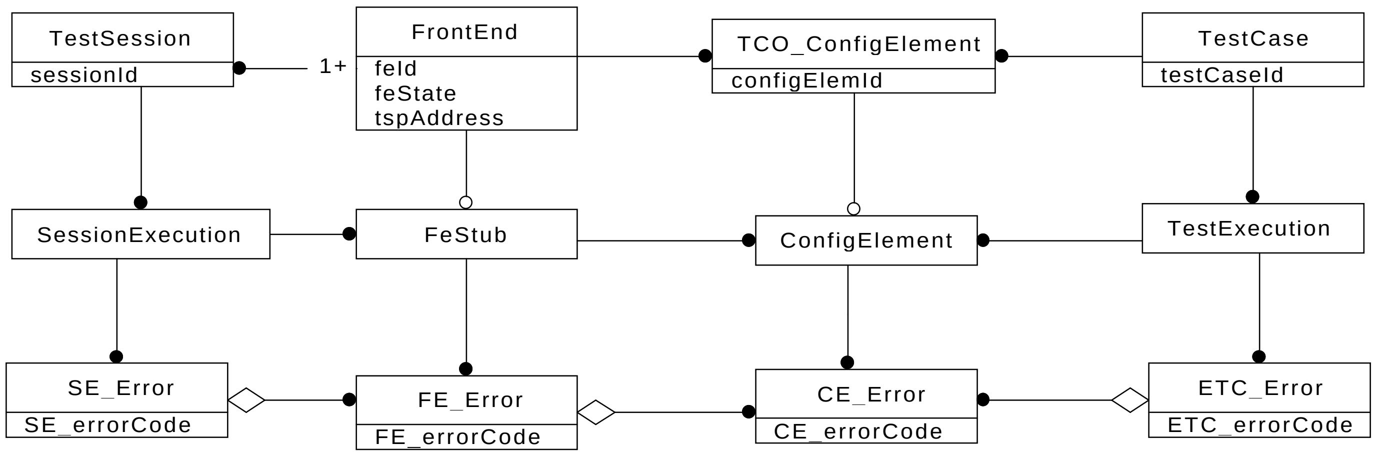


Figure C-8: Errors Model

A test session uses a set of front ends, managing a set of test configuration elements.

During a test execution, the supervisor uses front end stubs and configuration element stubs processes in order to manage all the real platform processes (FrontEnd, MTC, PTC, PCO).

This model aims at giving the supervisor a global (as complete as possible) view of the "state" of all the platform running processes. From a supervisor point of view:

- A session error contains a session error code and is made of several front end errors;
- A front end error contains a front end error code, and is made of several Configuration Element Error;
- A test case Error contains a test case Error code, and is made of several test configuration element errors;

C.8 Typical adaptation Layer Implementation

C.8.1 TCP/IP

TCP/IP is a protocol which is very likely to be used to transport TSP1+ PDUs. Like the OSI transport protocol of class 4, TCP offers a reliable, connection oriented peer-to-peer transport service on an unreliable lower layer. The services of TCP/IP can easily be accessed through a programming interface (function calls) called sockets. A socket is an connection endpoint (like an OSI-SAP) where user processes can connect to from above. The table below lists the main services provided by the socket interface. Details about socket programming will not be provided.

Table C-1: Services provided by the TCP socket

Socket	create TSAP
Bind	associate an ASCII name to a socket
Listen	specify queue for incoming connections
Accept	accept incoming connection
Connect	establish connection with remote socket
Shutdown	close connection with remote socket
Send	send data
Receive	receive data
Select	Check socket for reading or writing

A FE shall be server, that means it shall make it's socket known to the public with Bind and wait for an incoming connection with Listen and Accept. The SS will establish the connection with Connect. When creating the socket, make sure to create a socket of type SOCK_STREAM to obtain reliable transport service. To avoid configuration problems on the SS, it is suggested, that the FE always waits for incoming connections on port 7000, but any other reasonable value is allowed. Of course, choosing a different value here leads to a different configuration in the SS.

Theoretically, within TCP/IP the transported data is a byte stream. For an implementor of TSP1+ using sockets, this means that is not easy to find out, when the amount of data representing an SDU is complete. For that reason, TSP1+ PDUs start with a header containing a flag indicating the beginning of the PDU and a length field (see chapter 14.1).

C.8.2 Serial Port with modem

The basic requirements for TSP1+ transport using modem and PSTN are:

- asynchronous mode, 8 bit, no parity;
- error corrected link;
- flow control between the PC (or workstation) and the modem;

In that case V.32 or V.32bis modems with V.42 error correction are recommended to be used (as minimum technical characteristics). Note that programming such a modem, supporting HAYES command set, may be performed as follows (for example):

- DTR (normal): AT&D2
- flow control (RTS/CTS): AT&K3
- ECM (V.42): AT&Q5 and AT\N3
- Compression (no): AT%C0
- Retrain (yes): AT%E2
- Automode (yes): ATN1
- Initialisation string: AT&F&C1&D2&K3\N3%C0%E2N1

The recommended technical requirements for the PC or workstation piloting the modem are:

- RTS/CTS flow control.
- asynchronous mode: proposed baud rate 19.2 kbit/s, 8 bit, no parity

Annex D (Informative): TSP1 SDL Model

Annex E (informative): Bibliography

The following material, though not specifically referenced in the body of the present document, gives supporting information.

- ETR 141 (1994): "Methods for Testing and Specification (MTS); Protocol and profile conformance testing specifications The Tree and Tabular Combined Notation (TTCN) style guide"
- ETR 193 (1995): "Method for Testing and Specification (MTS); Network Integration Testing; Methodological Aspects; Test Co-ordination Procedure Styleguide
- ETR 303 (1997): "Method for Testing and Specification (MTS); Network Integration Testing; Test Synchronization; Architectural Reference; Test Synchronization Protocol 1 specification."
- TR 101 667 (1999): "Methods for Testing and Specification (MTS); Network Integration/Interconnection Testing (NIT); Reasons and goals for a global service testing approach
- ISO/IEC 9646-1 (1994): "Information technology; Open Systems Interconnection; Conformance testing methodology and framework; Part 1: General concepts "
- INTOOL/OTE/EC007 (1997): "OTE Architecture"
- "Inter-Domain Management: Specification Translation", The Open Group (1997); ISBN 1859121500

History

Document history		
4.0.0	09-Oct-1999	Complete revision of V3.4 text to make it (rather than the SDL) normative