ETSI ES 201 770 V4.2.4 (2000-09)

ETSI Standard

**Methods for Testing and Specification (MTS);
Test synchronization architectural reference;
Test Synchronization Protocol 1 plus (TSP1+) specification**

Reference

DES/MTS-00051

Keywords

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

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

Foreword

This ETSI Standard (ES) has been produced by ETSI Technical Committee Methods for Testing and Specification (MTS).

1 Scope

The present 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 the present document 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.

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

Conformance to the present document is achieved by satisfying the requirements identified in the Protocol Implementation Conformance Statement (PICS) proforma in annex A.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, 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.

[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 Modelling 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.

Configuration: 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: the 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
ETS	Executable Test Suite
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
NIT	Network Integration Testing
OMT	Object Modelling Technique
OSI	Open System Interconnection
PCO	Point of Control and Observation
PDU	Protocol Data Unit
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.

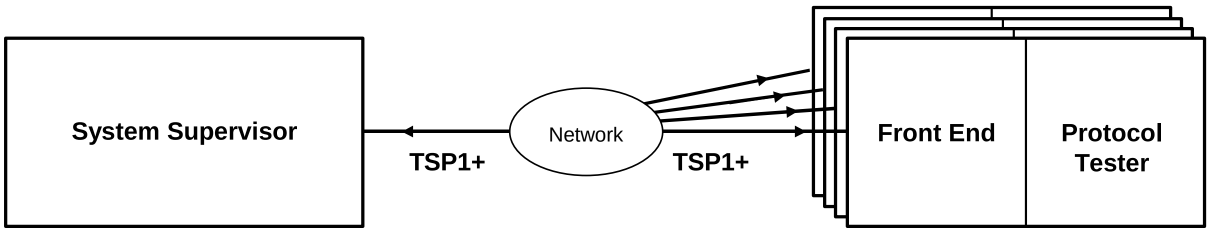


Figure 1: Basic TSP1+ system arrangement

The present document 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;
		
		Invalid parameter;
		
		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;
		
		Time not set;
		
		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;
		
		Unknown test case;
		
		Tree not found;
		
		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/IEC 6-3 [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; Message out of sequence.	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; Message out of sequence.	Mandatory	

4.3.2.3 Test completion

The Front-End serving the main test component shall use the CHECK_TCO_COMPLETED service primitive to report when 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; MOT not ready.		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.

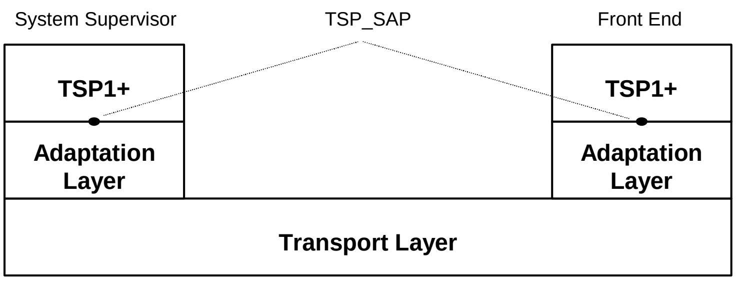


Figure 2: TSP1+ Layers

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.

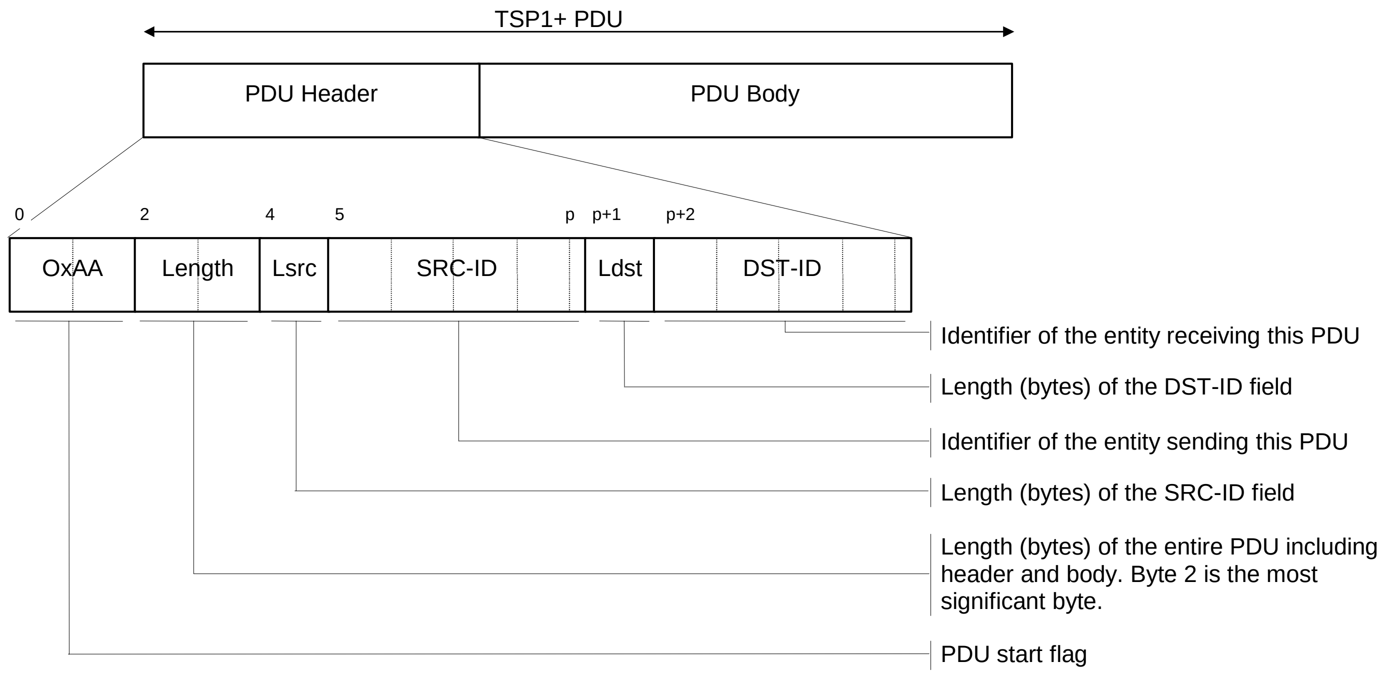

TSP1+ PDU
PDU Header
PDU Body
0
2
4
5
p
p+1
p+2
OxAA
Length
Lsrc
SRC-ID
Ldst
DST-ID
Identifier of the entity receiving this PDU
Length (bytes) of the DST-ID field
Identifier of the entity sending this PDU
Length (bytes) of the SRC-ID field
Length (bytes) of the entire PDU including header and body. Byte 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 [5] specification, "OTE Piloting protocol design specification".

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;


   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

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                     ::= ENUMERATED  { first                   (1),
                                           current                 (0),
                                           last                    (2),
                                           first-last              (3) }

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

TraceSet                 ::= SEQUENCE    { conf-elem-id    ConfElemId,
                                           trace-filter    TraceFilter }
TraceConfig              ::=SEQUENCE OF TraceSet
TraceType                ::=PrintableString

ServiceId                ::= ENUMERATED  { tsp1UnknownService      (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              ::= ENUMERATED  { intermediate-verdict    (0),
                                           final-verdict           (1) }
VerdictValue             ::= ENUMERATED  { fail                    (0),
                                           inconc                  (1),
                                           pass                    (2) }


Err-TSP1-INIT            ::= ENUMERATED  { errMOTNotReady          (1),
                                           errUnknownEts           (3),
                                           errUnknownSession       (4) }

Err-TSP1-CHK-CONF        ::= ENUMERATED  { errMOTNotReady          (1),
                                           errMOTNotConnected      (2),
                                           errUnknownSession       (4) }

Err-TSP1-SET-PARAMETER   ::= ENUMERATED  { errMOTNotReady          (1),
                                           errMOTNotConnected      (2),
                                           errInvalidVariable      (7) }
```

```
   Err-TSP1-SET-TIME        ::= ENUMERATED  { errMOTNotReady          (1),
                                              errMOTNotConnected      (2),
                                              errTimeNotAssigned      (5) }

   Err-TSP1-LIST-FE-SERVICES ::= ENUMERATED { errMOTNotConnected      (2) }

   Err-TSP1-CREATE          ::= ENUMERATED  { errMOTNotReady          (1),
                                              errMOTNotConnected      (2),
                                              errUnknownETS           (3),
                                              errTreeNotFound         (87) }

   Err-TSP1-INFO            ::= ENUMERATED  { errMOTNotConnected      (2),
                                              errOutOfSequenceMessage (86) }

   Err-TSP1-ASK-TRACE       ::= ENUMERATED  { errMOTNotReady          (1),
                                              errMOTNotConnected      (2),
                                              errTraceNotAvailable    (6) }

   Err-TSP1-END             ::= ENUMERATED  { errMOTNotReady          (1) }

   Err-TSP1-CANCEL-OP       ::= ENUMERATED  { errMOTNotReady          (1),
                                              errMOTNotConnected      (2) }

   Err-TSP1-TCO-FAILURE     ::= ENUMERATED  { errProcessingFailure  (170),
                                              errCommunicationLost  (200),
                                              errTesterCrash        (201) }

   ErrorCode                ::= CHOICE      { initErrs      [0] Err-TSP1-INIT,
                                              chkConfErrs   [1] Err-TSP1-CHK-CONF,
                                              setParsErrs   [2] Err-TSP1-SET-PARAMETER,
                                              setTimeErrs   [3] Err-TSP1-SET-TIME,
                                              listServErrs  [4] Err-TSP1-LIST-FE-SERVICES,
                                              createErrs    [5] Err-TSP1-CREATE,
                                              infoErrs      [6] Err-TSP1-INFO,
                                              askTraceErrs  [7] Err-TSP1-ASK-TRACE,
                                              endErrs       [8] Err-TSP1-END,
                                              cancelOpErrs  [9] Err-TSP1-CANCEL-OP }

   TSP1-ERR                 ::= SEQUENCE { service-id       ServiceId,
                                           err-code         ErrorCode,
                                           err-cause [0]    PrintableString OPTIONAL }
--                The content of 'err-cause' is manufacturer specific and therefore out --
--                of scope of this standard.                                            --

   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            Has no parameters

-- TSP1-INIT-COMPLETE       Has no parameters

-- TSP1-CHK-CONF            Has no parameters

-- TSP1-CHK-CONF-ACK        Has no parameters

   TSP1-SET-PARAMETER       ::= ParamList

-- TSP1-SET-PARAMETER-ACK   Has no parameters

   TSP1-SET-TIME            ::= GeneralizedTime

-- TSP1-SET-TIME-ACK        Has no parameters

-- TSP1-LIST-FE-SERVICES    Has no parameters

   TSP1-LIST-FE-SERVICES-ACK::= ServiceList

   TSP1-CREATE              ::= SEQUENCE { tco-id        [0] TcoId         OPTIONAL,
                                           test-case-id  [1] TestCaseId,
                                           tree              TreeId        OPTIONAL,
                                           param-list        ParamList     OPTIONAL }

   TSP1-CREATE-ACK          ::= 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           ::= 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 } OPTIONAL }

-- TSP1-END                 Has no parameters

-- TSP1-END-ACK             Has no parameters

   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 }

   TSP1-TCO-FAILURE         ::= SEQUENCE { fe-id             FeId,
                                           fault-code        Err-TSP1-TCO-FAILURE }

-- Definition of 'GeneralizedTime' which is not supported by Telelogic Tau.

  GeneralizedTime            ::= PrintableString

 END
```

NOTE: Where necessary, the identifiers specified in table 17 use the dash ("-") character rather than an underscore ("_") as this is syntactically correct ASN.1. Throughout the remainder of the document, these dash characters are replaced by underscores to ensure alignment with the identifiers used in the TSP1+ SDL model.

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_LIST 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	TSP1_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	TSP1_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
NOTE 1: The value of TCO_VERDICT_TYPE is fixed at "intermediate-verdict". NOTE 2: The value of TCO_VERDICT_TYPE is fixed at "final-verdict".			

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.

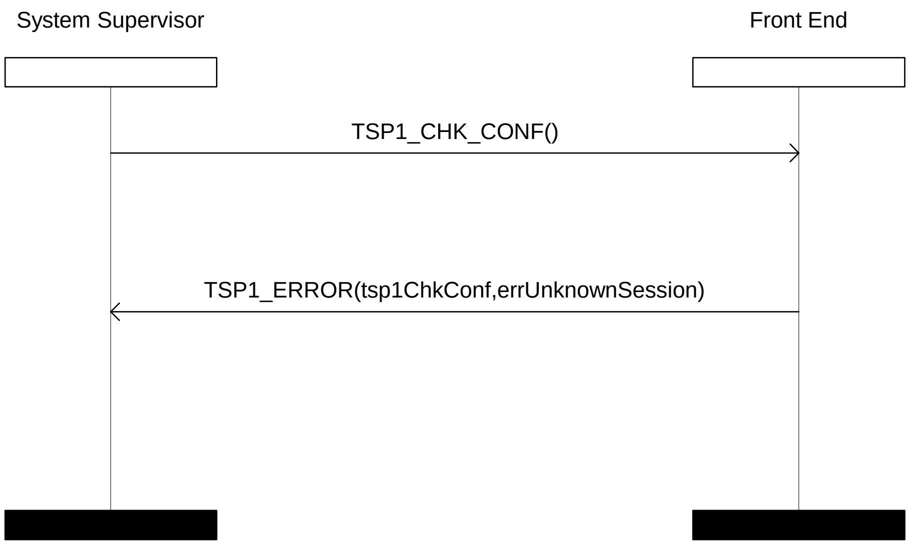


Figure 4: Example message sequence for TSP1+ processing error

4.4.5.2 Test Component fault handling

If a Front-End detects a fault condition in one of its Test Components, it shall report this to the System Supervisor using the TSP1_TCO_FAILURE pdu as shown in figure 5.

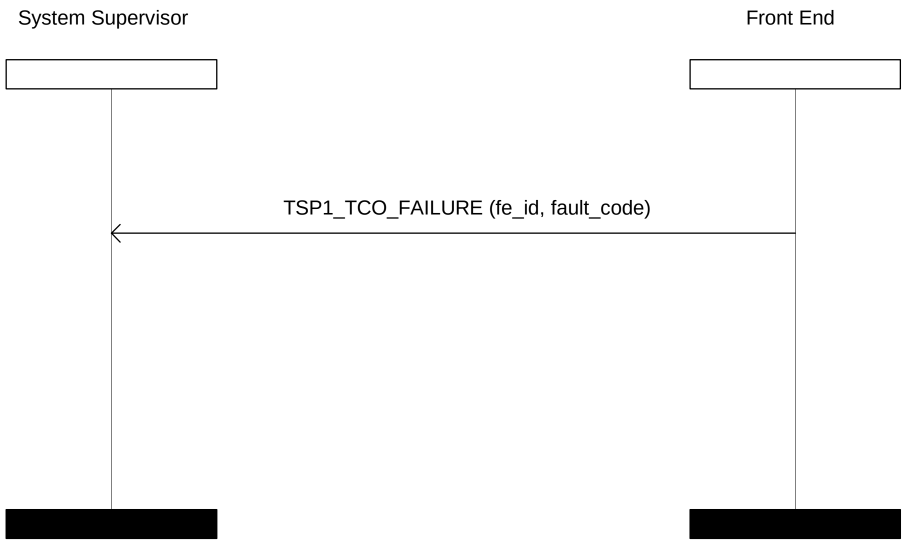


Figure 5: Example message flow on detection of a failure in a TCO

The following failure conditions shall be valid:

- Processing Failure;
- Communication Lost;
- Tester Crash.

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

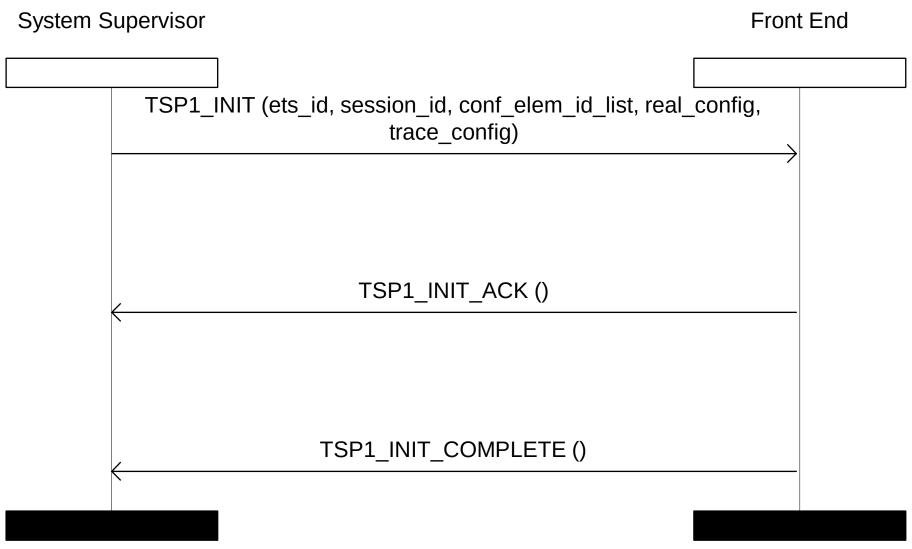

System Supervisor
Front End
TSP1_INIT (ets_id, session_id, conf_elem_id_list, real_config, trace_config)
TSP1_INIT_ACK ()
TSP1_INIT_COMPLETE ()

Figure 6: Example of normal operation of the TSP1_INIT PDU

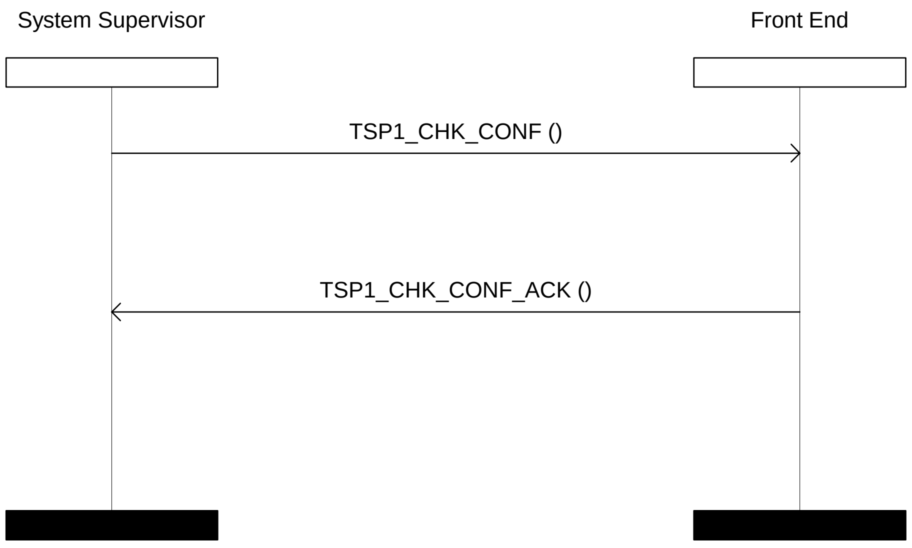


Figure 7: Example of normal operation of the TSP1_CHK_CONF PDU

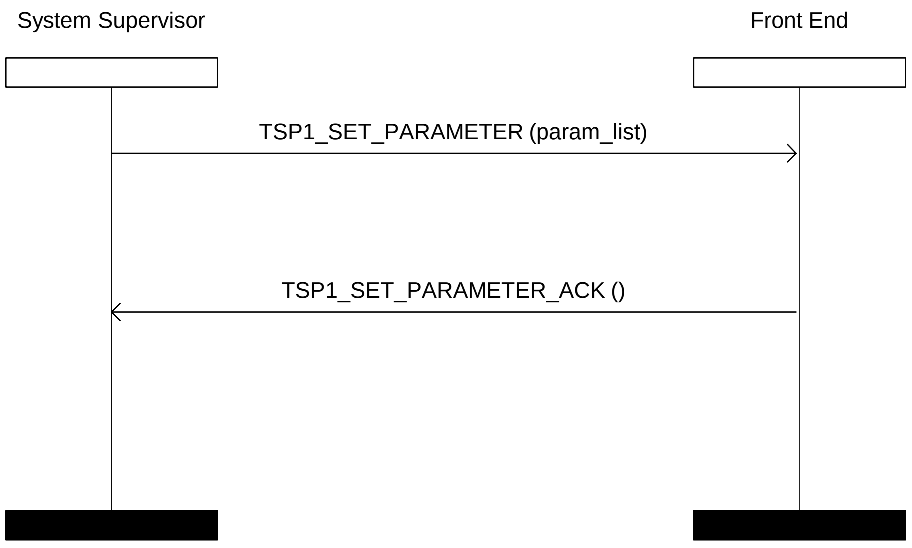


Figure 8: Example of normal operation of the TSP1_SET_PARAMETER PDU

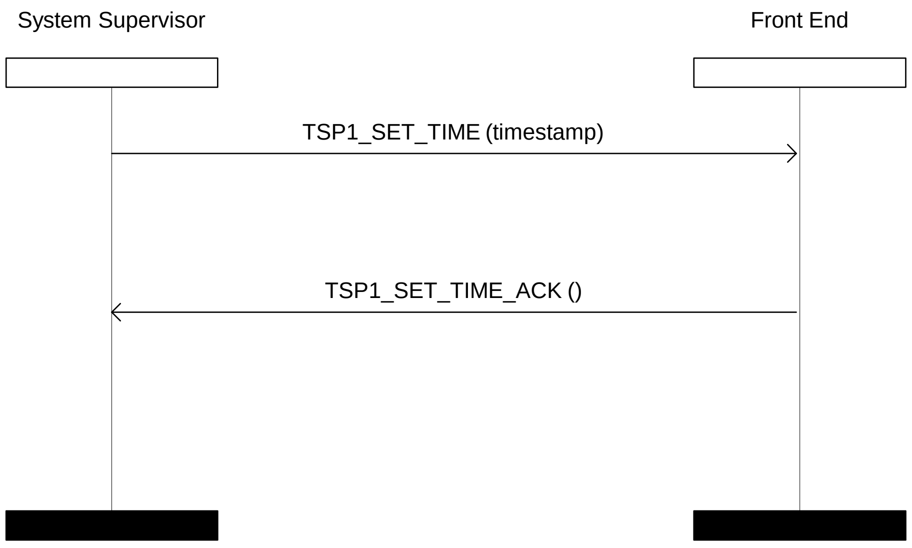


Figure 9: Example of normal operation of the TSP1_SET_TIME PDU

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 the present document.

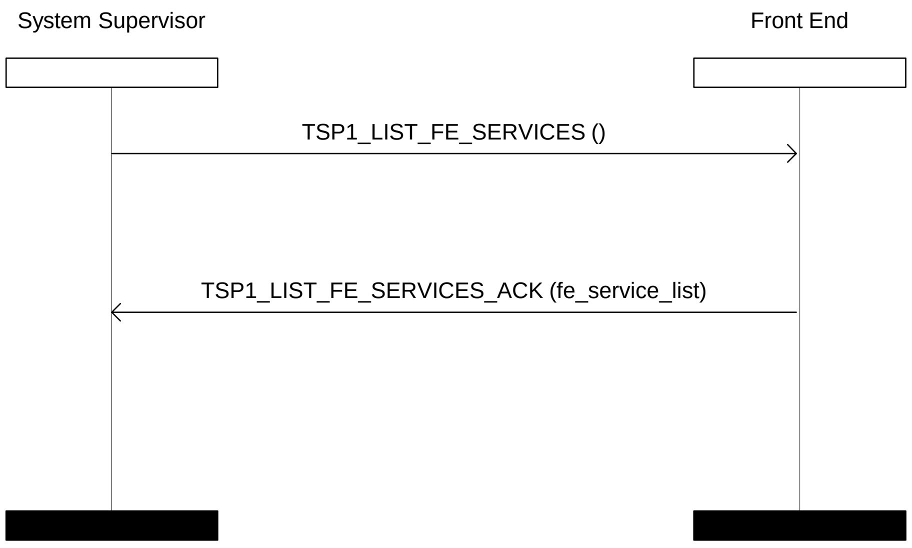


Figure 10: Example of normal operation of the TSP1_LIST_FE_SERVICES PDU

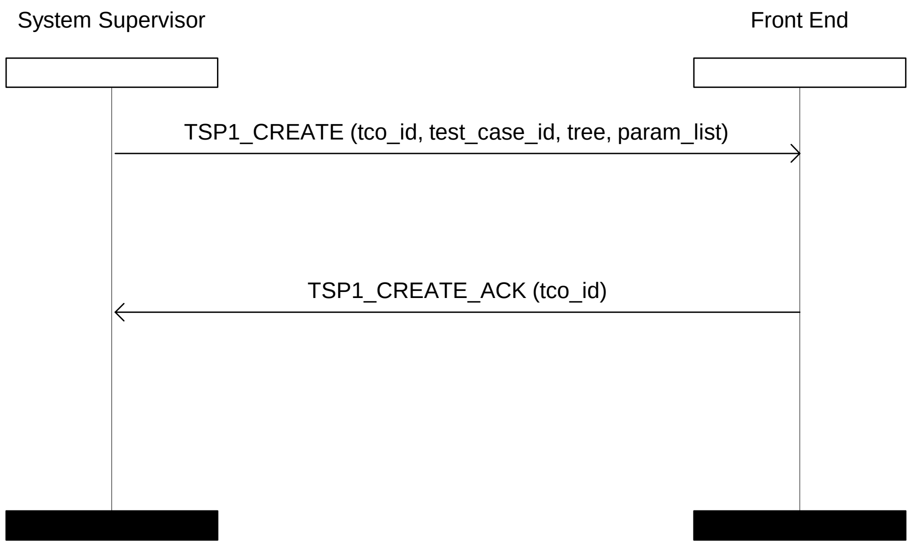


Figure 11: Example of normal operation of the TSP1_CREATE PDU

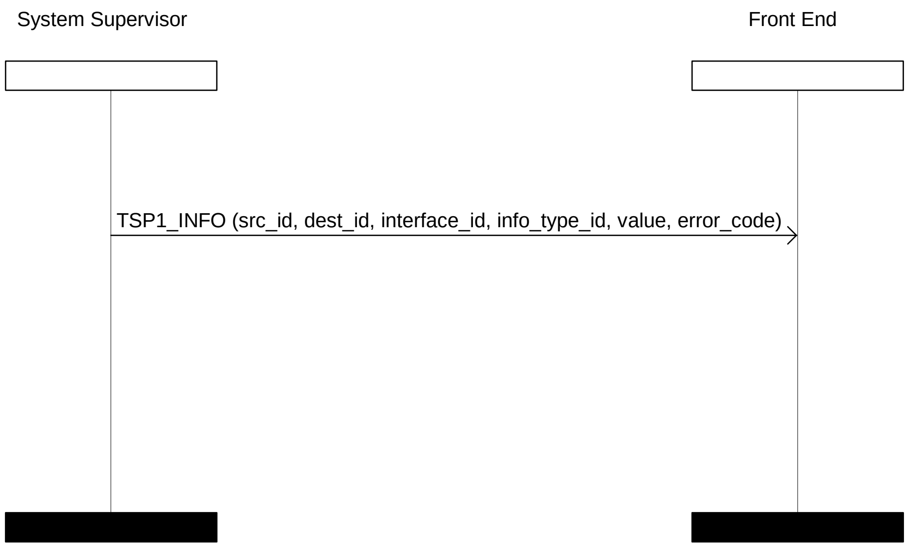

System Supervisor
Front End
TSP1_INFO (src_id, dest_id, interface_id, info_type_id, value, error_code)

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

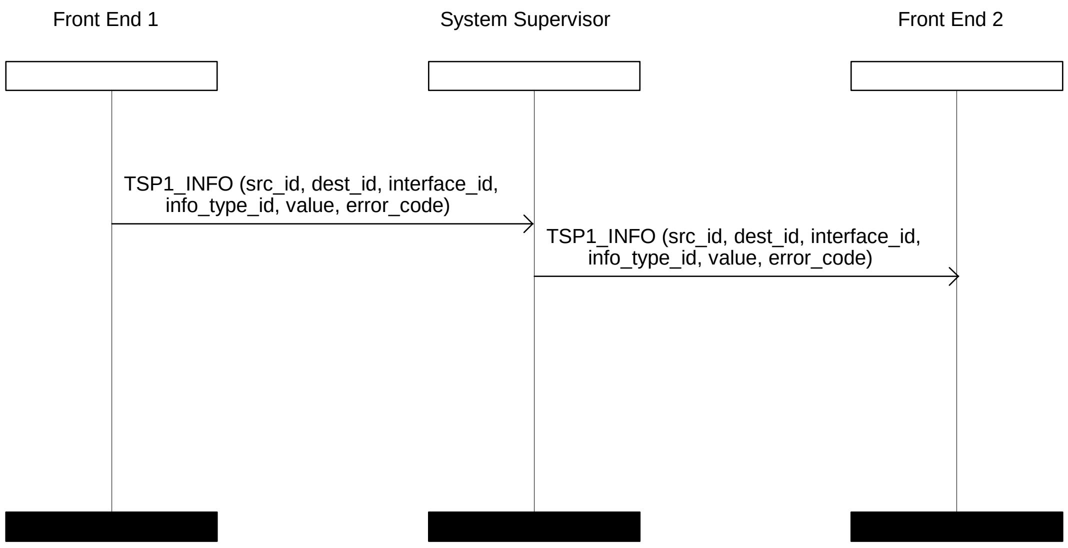

Front End 1
System Supervisor
Front End 2
TSP1_INFO (src_id, dest_id, interface_id, info_type_id, value, error_code)
TSP1_INFO (src_id, dest_id, interface_id, info_type_id, value, error_code)

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

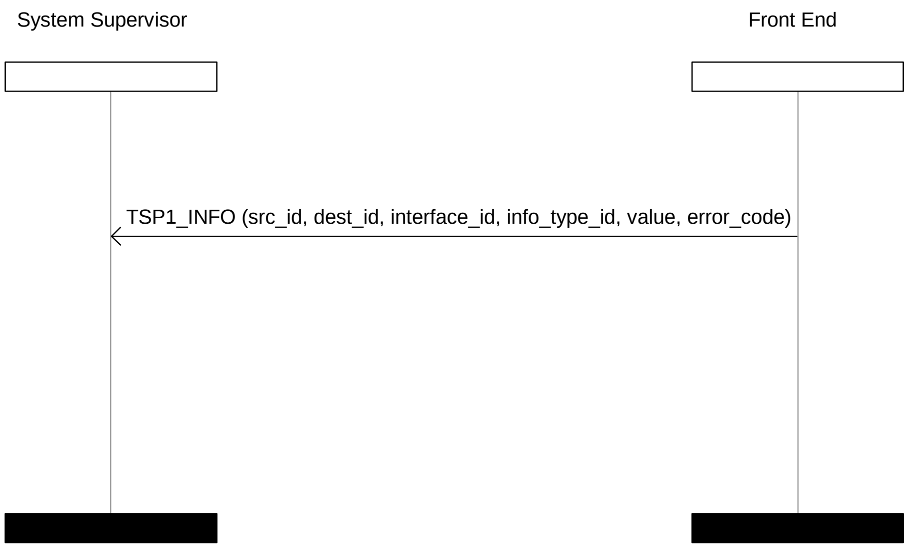

System Supervisor
Front End
TSP1_INFO (src_id, dest_id, interface_id, info_type_id, value, error_code)

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

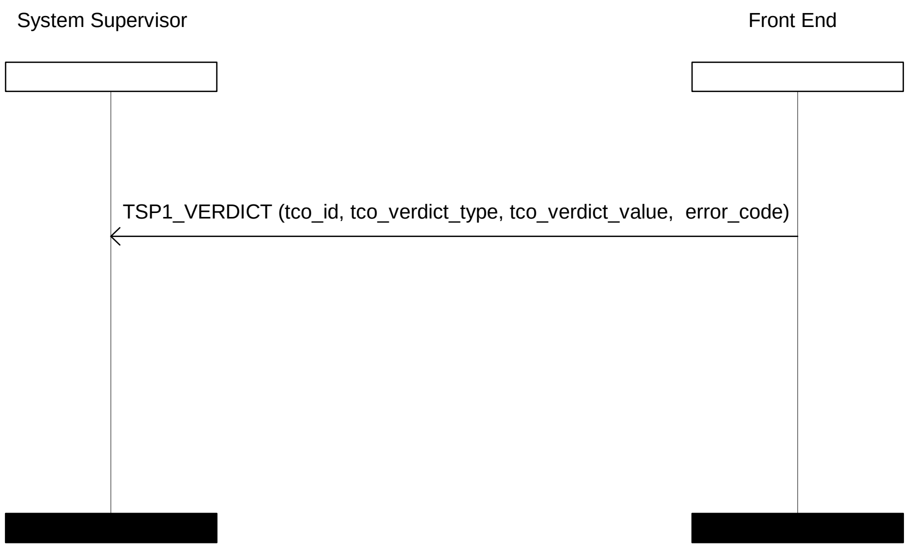

System Supervisor
Front End
TSP1_VERDICT (tco_id, tco_verdict_type, tco_verdict_value, error_code)

Figure 15: Example of normal operation of the TSP1_VERDICT PDU

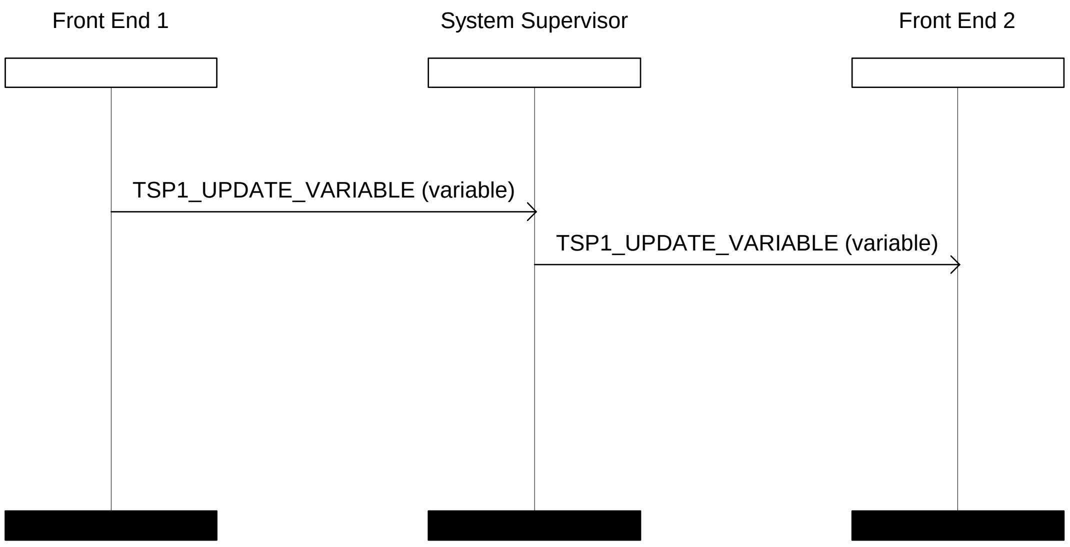

Front End 1
System Supervisor
Front End 2
TSP1_UPDATE_VARIABLE (variable)
TSP1_UPDATE_VARIABLE (variable)

Figure 16: Example of normal operation of the TSP1_UPDATE_VARIABLE PDU

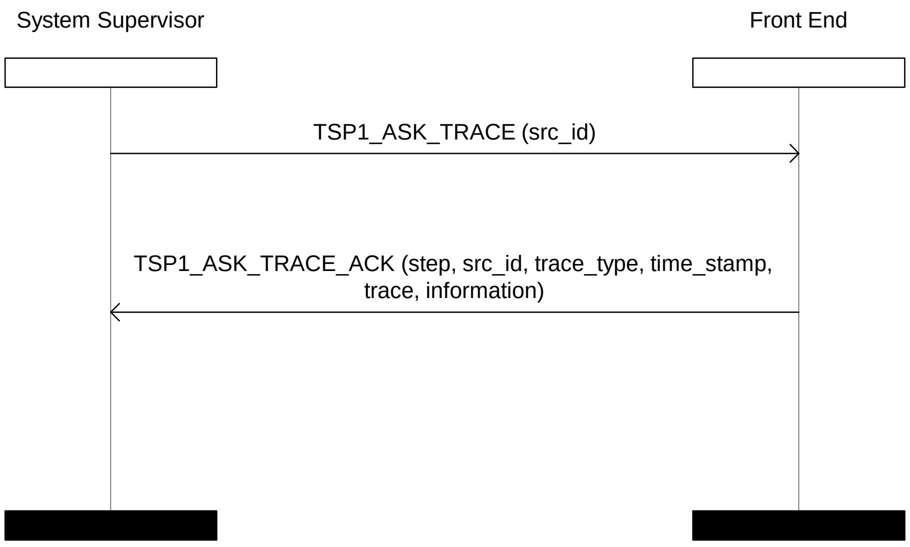

System Supervisor
Front End
TSP1_ASK_TRACE (src_id)
TSP1_ASK_TRACE_ACK (step, src_id, trace_type, time_stamp, trace, information)

Figure 17: Example of normal operation of the TSP1_ASK_TRACE PDU

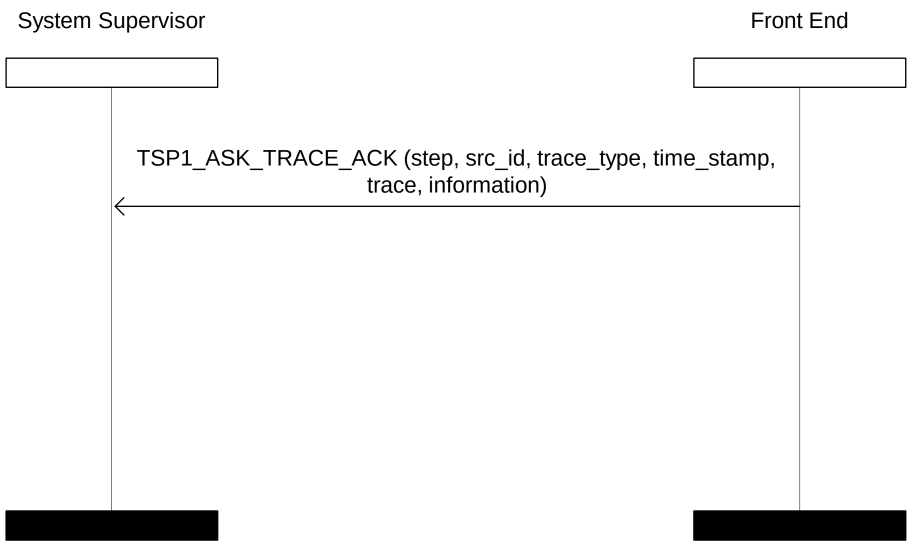

System Supervisor
Front End
TSP1_ASK_TRACE_ACK (step, src_id, trace_type, time_stamp, trace, information)

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

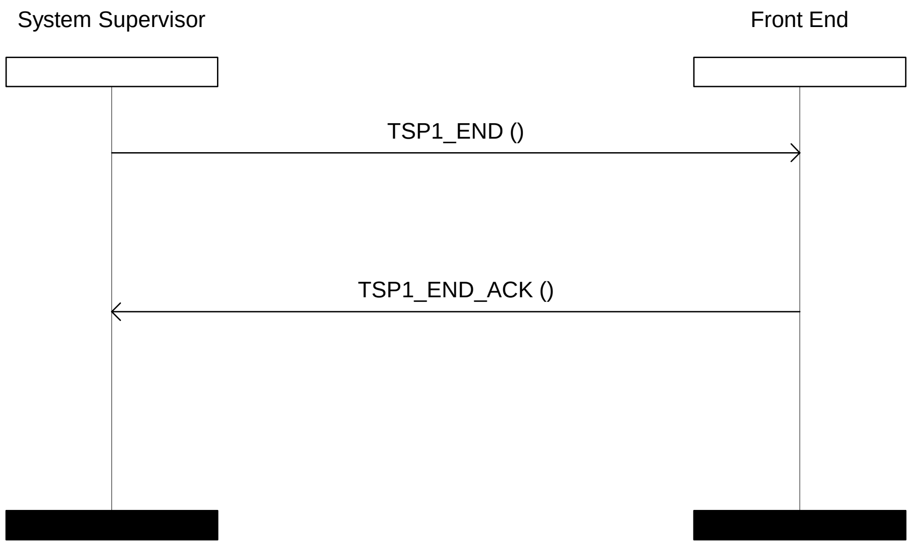


Figure 19: Example of normal operation of the TSP1_END PDU

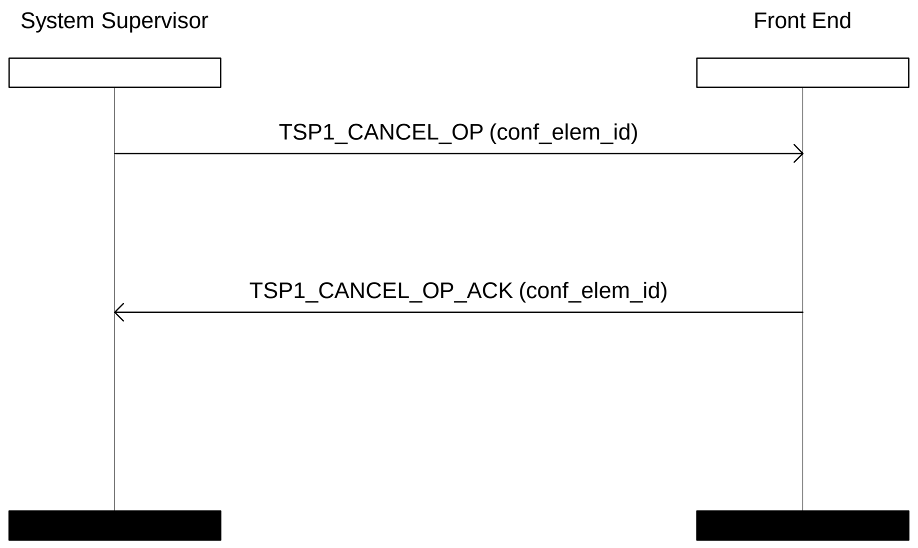

System Supervisor
Front End
TSP1_CANCEL_OP (conf_elem_id)
TSP1_CANCEL_OP_ACK (conf_elem_id)

Figure 20: Example of normal operation of the TSP1_CANCEL_OP PDU

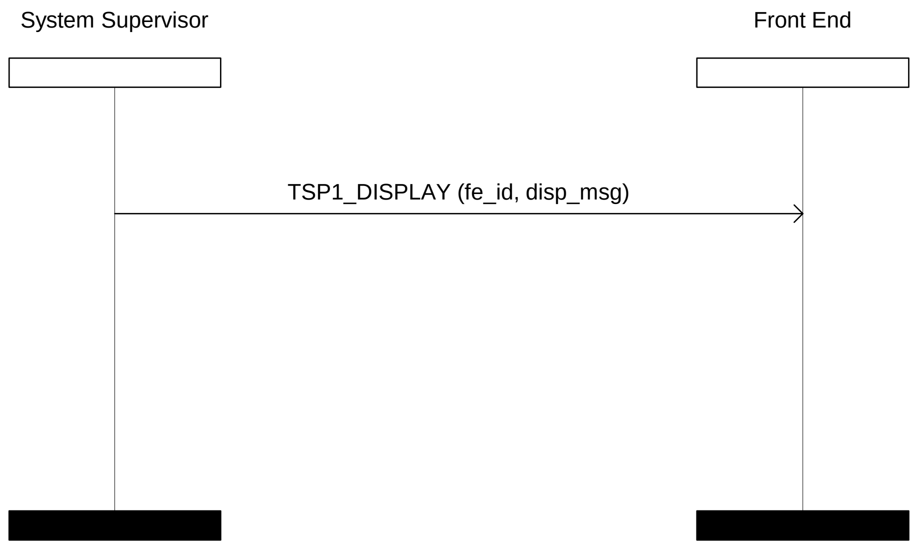


Figure 21: Example of normal operation of the TSP1_DISPLAY PDU to send display data from the System Supervisor to a Front-End

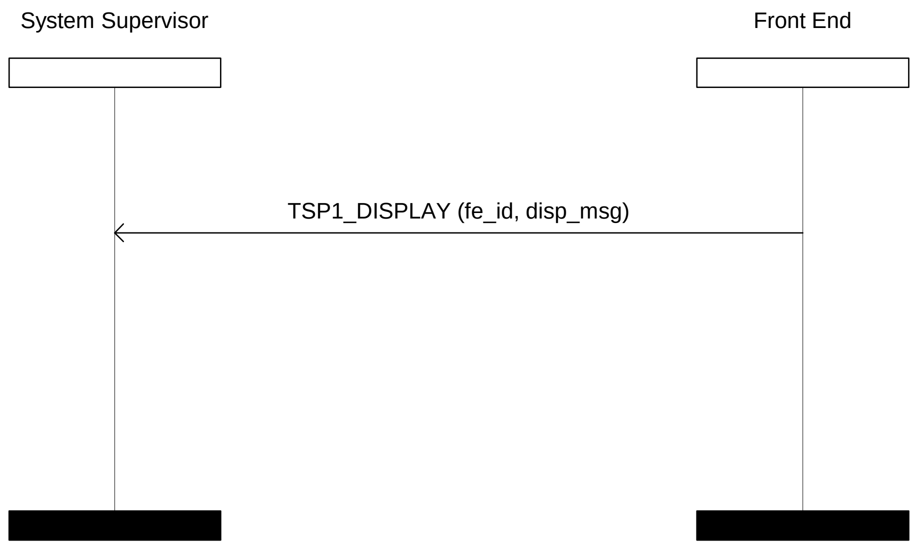


Figure 22: Example of normal operation of the TSP1_DISPLAY PDU to send display data from a Front-End to the System Supervisor

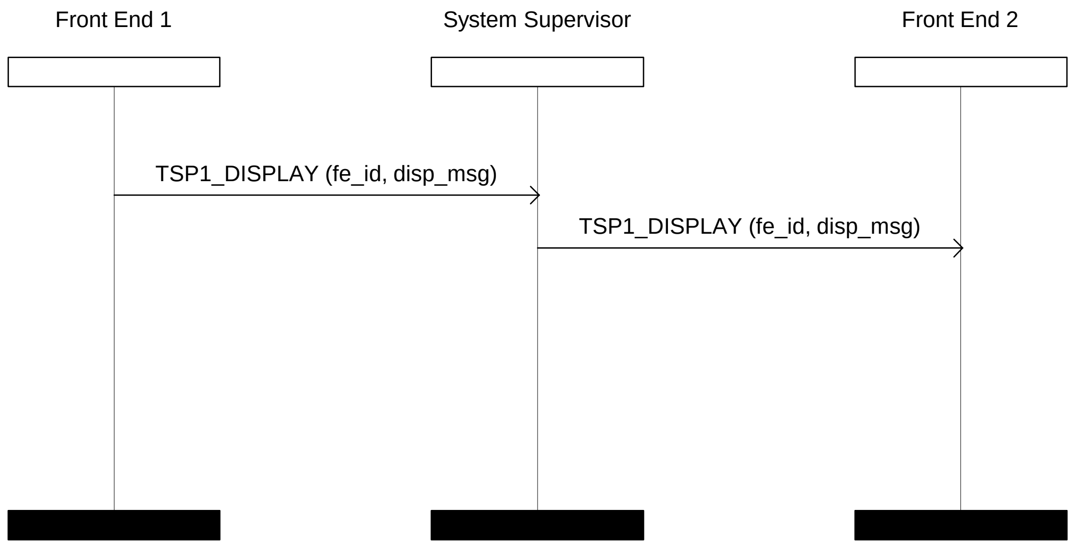

Front End 1
System Supervisor
Front End 2
TSP1_DISPLAY (fe_id, disp_msg)
TSP1_DISPLAY (fe_id, disp_msg)

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.

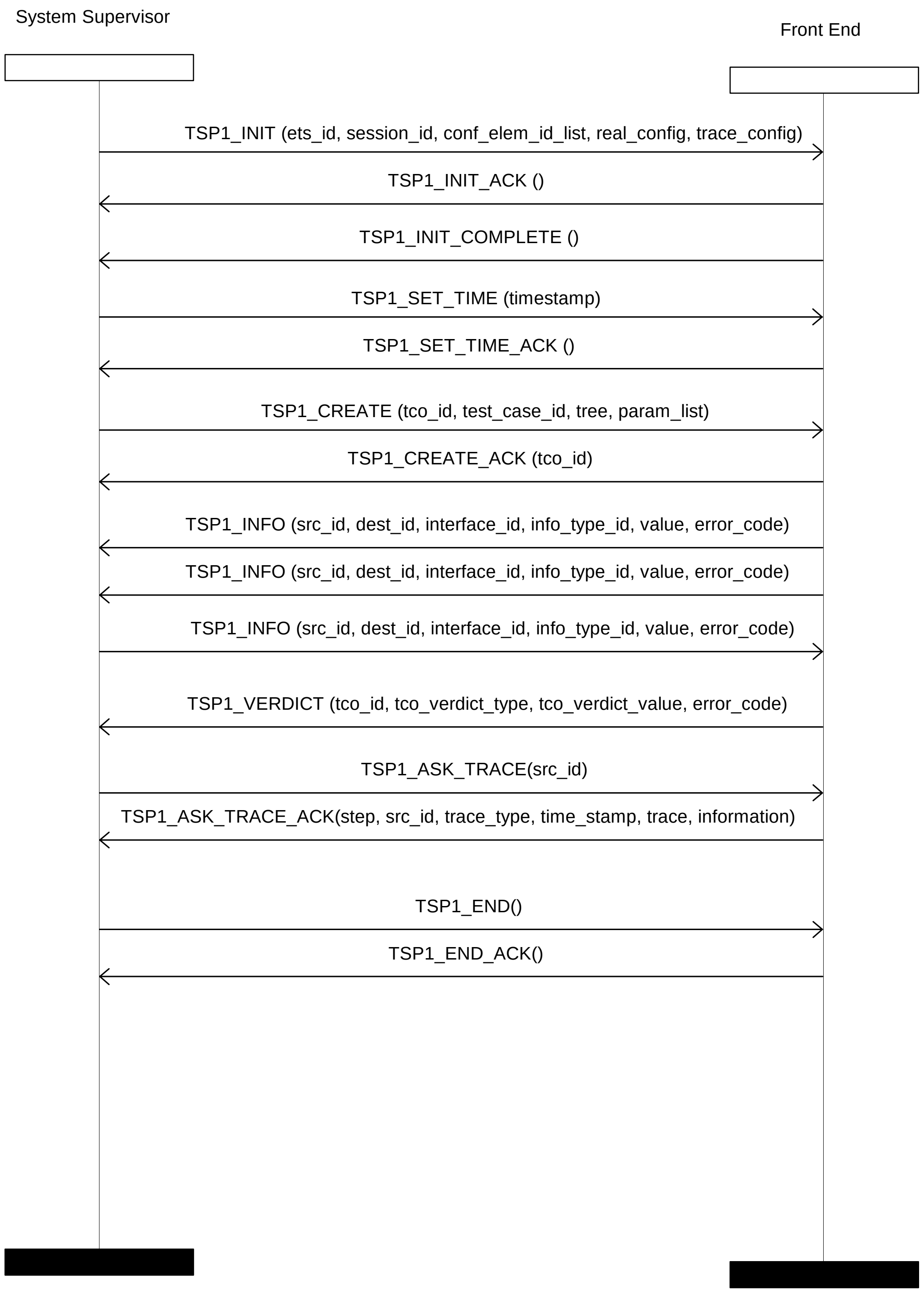

System Supervisor
Front End
TSP1_INIT (ets_id, session_id, conf_elem_id_list, real_config, trace_config)
TSP1_INIT_ACK ()
TSP1_INIT_COMPLETE ()
TSP1_SET_TIME (timestamp)
TSP1_SET_TIME_ACK ()
TSP1_CREATE (tco_id, test_case_id, tree, param_list)
TSP1_CREATE_ACK (tco_id)
TSP1_INFO (src_id, dest_id, interface_id, 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

The procedures for a Front-End are written in terms of the following conceptual states existing within its TSP1+ service control entity.

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 in the TSP1_INIT message is invalid, send TSP1_ERROR with the error type "Unknown ETS". If the Session identifier specified in the TSP1_INIT message is not 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 a TSP1_INIT pdu has not been received to open the testing session, send TSP1_ERROR with the error type "MOT not connected". If a TCE reports that is running an ETS which is not the ETS previously specified in the tsp1-INIT signal, send tsp1-ERROR with the error type set to "Unknown ETS". 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 to the 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 the responses from the TCEs indicate that the current test has been successfully closed in each, send TSP1_CANCEL_OP_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". 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.	

4.4.7.15 TSP1_TCO_FAILURE

In the event that a Front-End detects a fault condition in a TCO connected to it, or a TCO reports a fault condition to the Front-End, the Front-End shall send a TSP1_TCO_FAILURE message to the System Supervisor with an identification of the fault type. The fault type shall have one of the following values:

- Processing Failure;
- Communication Lost;
- Tester Crash.

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 (informative): 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 [2].

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 [2], 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 a 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 a 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 [2], 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: 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	Yes [ ] No [ ]
A2	Behaviour as a Front-End	4.2.2	o.1	Yes [ ] No [ ]

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	Yes [ ] No [ ]
B2	Signalling procedures at a Front-End	4.4.7	A2:m	Yes [ ] No [ ]

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	Yes [ ] No [ ]
C2	Receipt of TSP1-TCO-FAILURE	4.4.5.2	B1:m	Yes [ ] No [ ]
C3	Sending of TSP1-INIT and receipt of TSP1-INIT-ACK PDU	4.4.7.1	B1:m	Yes [ ] No [ ]
C4	Receipt of TSP1-INIT-COMPLETE PDU	4.4.7.2	B1:m	Yes [ ] No [ ]
C5	Sending of TSP1-CHK-CONF PDU and receipt of TSP1-CHK-CONF-ACK PDU	4.4.7.3	B1:m	Yes [ ] No [ ]
C6	Sending of TSP1-SET-PARAMETER PDU and receipt of TSP1-SET-PARAMETER PDU	4.4.7.4	B1:m	Yes [ ] No [ ]
C7	Sending of TSP1-SET-TIME PDU and receipt of TSP1-SET-TIME-ACK PDU	4.4.7.5	B1:m	Yes [ ] No [ ]
C8	Sending of TSP1-LIST-FE-SERVICES PDU and receipt of TSP1-LIST-FE-SERVICES-ACK PDU	4.4.7.6	B1:m	Yes [ ] No [ ]
C9	Sending of TSP1-CREATE PDU and receipt of TSP1-CREATE-ACK PDU	4.4.7.7	B1:m	Yes [ ] No [ ]
C10	Receipt of TSP1-INFO PDU	4.4.7.8	B1:m	Yes [ ] No [ ]
C11	Receipt of TSP1-VERDICT PDU	4.4.7.9	B1:m	Yes [ ] No [ ]
C12	Receipt of TSP1-UPDATE-VARIABLE PDU	4.4.7.10	B1:m	Yes [ ] No [ ]
C13	Sending of TSP1-ASK-TRACE PDU and receipt of TSP1-ASK-TRACE-ACK PDU	4.4.7.11	B1:m	Yes [ ] No [ ]
C14	Sending of TSP1-END PDU and receipt of TSP1-END-ACK PDU	4.4.7.12	B1:m	Yes [ ] No [ ]
C15	Sending of TSP1-CANCEL-OP PDU and receipt of TSP1-CANCEL-OP-ACK PDU	4.4.7.13	B1:m	Yes [ ] No [ ]
C16	Sending of TSP1-DISPLAY PDU	4.4.7.14	B1:m	Yes [ ] No [ ]
C17	Receipt of TSP1-DISPLAY PDU	4.4.7.14	B1:m	Yes [ ] No [ ]

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	Yes [ ] No [ ]
D2	Sending of TSP1-TCO-FAILURE	4.4.5.2	B2:m	Yes [ ] No [ ]
D3	Receipt of TSP1-INIT and sending of TSP1-INIT-ACK PDU	4.4.7.1	B2:m	Yes [ ] No [ ]
D4	Sending of TSP1-INIT-COMPLETE PDU	4.4.7.2	B2:m	Yes [ ] No [ ]
D5	Receipt of TSP1-CHK-CONF PDU and sending of TSP1-CHK-CONF-ACK PDU	4.4.7.3	B2:m	Yes [ ] No [ ]
D6	Receipt of TSP1-SET-PARAMETER PDU and sending of TSP1-SET-PARAMETER PDU	4.4.7.4	B2:m	Yes [ ] No [ ]
D7	Receipt of TSP1-SET-TIME PDU and sending of TSP1-SET-TIME-ACK PDU	4.4.7.5	B2:m	Yes [ ] No [ ]
D8	Receipt of TSP1-LIST-FE-SERVICES PDU and sending of TSP1-LIST-FE-SERVICES-ACK PDU	4.4.7.6	B2:m	Yes [ ] No [ ]
D9	Receipt of TSP1-CREATE PDU and sending of TSP1-CREATE-ACK PDU	4.4.7.7	B2:m	Yes [ ] No [ ]
D10	Sending of TSP1-INFO PDU	4.4.7.8	B2:m	Yes [ ] No [ ]
D11	Sending of TSP1-VERDICT PDU	4.4.7.9	B2:m	Yes [ ] No [ ]
D12	Sending of TSP1-UPDATE-VARIABLE PDU	4.4.7.10	B2:m	Yes [ ] No [ ]
D13	Receipt of TSP1-ASK-TRACE PDU and sending of TSP1-ASK-TRACE-ACK PDU	4.4.7.11	B2:m	Yes [ ] No [ ]
D14	Receipt of TSP1-END PDU and sending of TSP1-END-ACK PDU	4.4.7.12	B2:m	Yes [ ] No [ ]
D15	Receipt of TSP1-CANCEL-OP PDU and sending of TSP1-CANCEL-OP-ACK PDU	4.4.7.13	B2:m	Yes [ ] No [ ]
D16	Receipt of TSP1-DISPLAY PDU	4.4.7.14	B2:m	Yes [ ] No [ ]
D17	Sending of TSP1-DISPLAY PDU	4.4.7.14	B2:m	Yes [ ] No [ ]

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	Yes [ ] No [ ]
E2	TSP1-TCO-FAILURE	4.4.3	B1:m	Yes [ ] No [ ]
E3	TSP1-INIT	4.4.3	B1:m	Yes [ ] No [ ]
E4	TSP1-INIT-ACK	4.4.3	B1:m	Yes [ ] No [ ]
E5	TSP1-INIT-COMPLETE	4.4.3	B1:m	Yes [ ] No [ ]
E6	TSP1-CHK-CONF	4.4.3	B1:m	Yes [ ] No [ ]
E7	TSP1-CHK-CONF-ACK	4.4.3	B1:m	Yes [ ] No [ ]
E8	TSP1-SET-PARAMETER	4.4.3	B1:m	Yes [ ] No [ ]
E9	TSP1-SET-PARAMETER	4.4.3	B1:m	Yes [ ] No [ ]
E10	TSP1-SET-TIME	4.4.3	B1:m	Yes [ ] No [ ]
E11	TSP1-SET-TIME-ACK	4.4.3	B1:m	Yes [ ] No [ ]
E12	TSP1-LIST-FE-SERVICES	4.4.3	B1:m	Yes [ ] No [ ]
E13	TSP1-LIST-FE-SERVICES-ACK	4.4.3	B1:m	Yes [ ] No [ ]
E14	TSP1-CREATE	4.4.3	B1:m	Yes [ ] No [ ]
E15	TSP1-CREATE-ACK	4.4.3	B1:m	Yes [ ] No [ ]
E16	TSP1-INFO	4.4.3	B1:m	Yes [ ] No [ ]
E17	TSP1-VERDICT	4.4.3	B1:m	Yes [ ] No [ ]
E18	TSP1-UPDATE-VARIABLE	4.4.3	B1:m	Yes [ ] No [ ]
E19	TSP1-ASK-TRACE	4.4.3	B1:m	Yes [ ] No [ ]
E20	TSP1-ASK-TRACE-ACK	4.4.3	B1:m	Yes [ ] No [ ]
E21	TSP1-END	4.4.3	B1:m	Yes [ ] No [ ]
E22	TSP1-END-ACK	4.4.3	B1:m	Yes [ ] No [ ]
E23	TSP1-CANCEL-OP	4.4.3	B1:m	Yes [ ] No [ ]
E24	TSP1-CANCEL-OP-ACK	4.4.3	B1:m	Yes [ ] No [ ]
E25	TSP1-DISPLAY	4.4.3	B1:m	Yes [ ] No [ ]

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	Yes [ ] No [ ]
F2	TSP1-TCO-FAILURE	4.4.3	B2:m	Yes [ ] No [ ]
F3	TSP1-INIT	4.4.3	B2:m	Yes [ ] No [ ]
F4	TSP1-INIT-ACK	4.4.3	B2:m	Yes [ ] No [ ]
F5	TSP1-INIT-COMPLETE	4.4.3	B2:m	Yes [ ] No [ ]
F6	TSP1-CHK-CONF	4.4.3	B2:m	Yes [ ] No [ ]
F7	TSP1-CHK-CONF-ACK	4.4.3	B2:m	Yes [ ] No [ ]
F8	TSP1-SET-PARAMETER	4.4.3	B2:m	Yes [ ] No [ ]
F9	TSP1-SET-PARAMETER	4.4.3	B2:m	Yes [ ] No [ ]
F10	TSP1-SET-TIME	4.4.3	B2:m	Yes [ ] No [ ]
F11	TSP1-SET-TIME-ACK	4.4.3	B2:m	Yes [ ] No [ ]
F12	TSP1-LIST-FE-SERVICES	4.4.3	B2:m	Yes [ ] No [ ]
F13	TSP1-LIST-FE-SERVICES-ACK	4.4.3	B2:m	Yes [ ] No [ ]
F14	TSP1-CREATE	4.4.3	B2:m	Yes [ ] No [ ]
F15	TSP1-CREATE-ACK	4.4.3	B2:m	Yes [ ] No [ ]
F16	TSP1-INFO	4.4.3	B2:m	Yes [ ] No [ ]
F17	TSP1-VERDICT	4.4.3	B2:m	Yes [ ] No [ ]
F18	TSP1-UPDATE-VARIABLE	4.4.3	B2:m	Yes [ ] No [ ]
F19	TSP1-ASK-TRACE	4.4.3	B2:m	Yes [ ] No [ ]
F20	TSP1-ASK-TRACE-ACK	4.4.3	B2:m	Yes [ ] No [ ]
F21	TSP1-END	4.4.3	B2:m	Yes [ ] No [ ]
F22	TSP1-END-ACK	4.4.3	B2:m	Yes [ ] No [ ]
F23	TSP1-CANCEL-OP	4.4.3	B2:m	Yes [ ] No [ ]
F24	TSP1-CANCEL-OP-ACK	4.4.3	B2:m	Yes [ ] No [ ]
F25	TSP1-DISPLAY	4.4.3	B2:m	Yes [ ] No [ ]

Annex B (informative): Test Synchronization Architecture

B.1 Introduction

Figure B.1shows 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).

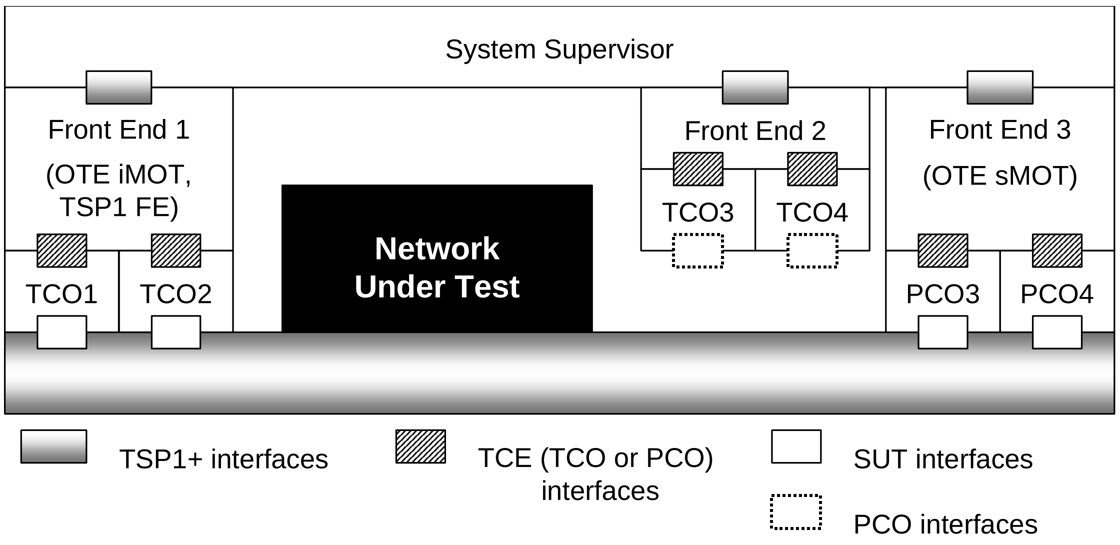


Figure B.1: Test Synchronization Architecture

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.

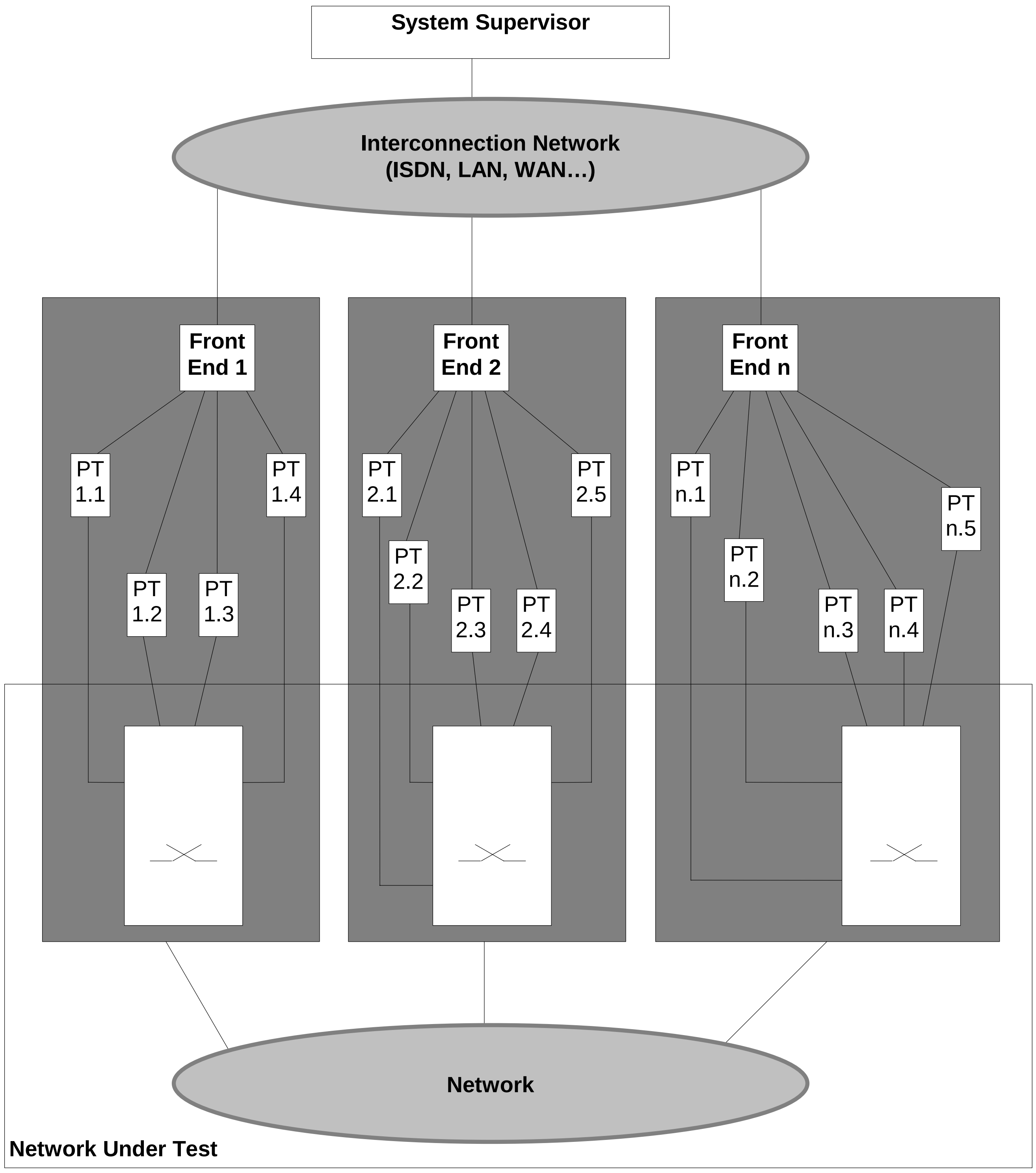


Figure B.2: Test Synchronization Architecture applied on NIT

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

The functionality of the different components is:

- 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 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.

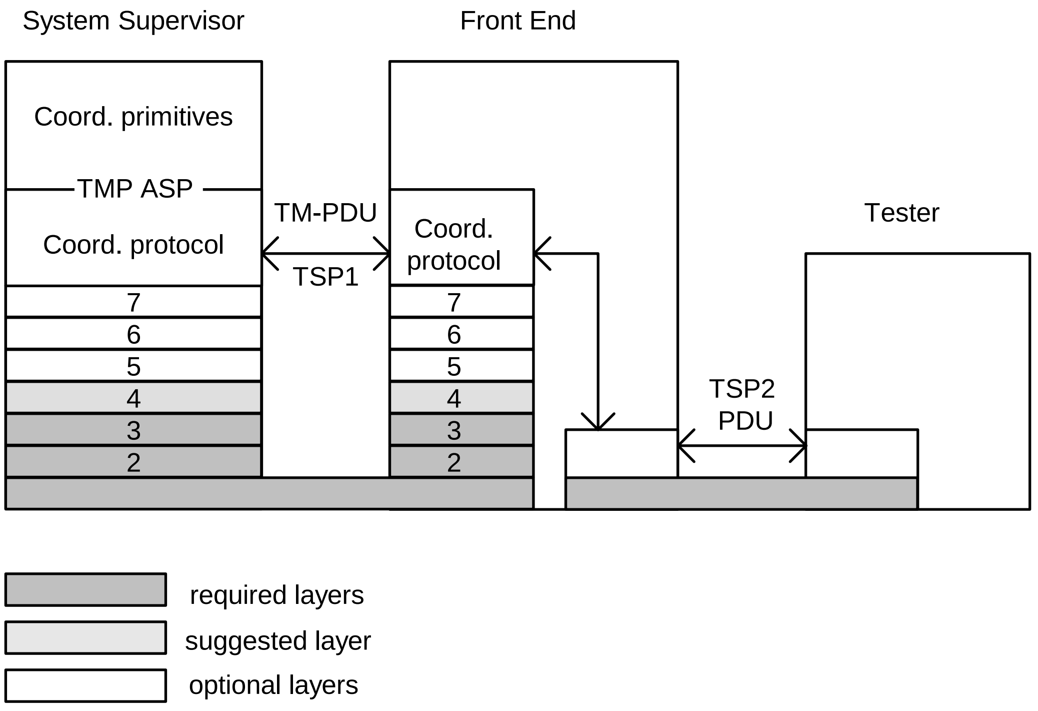


Figure B.3: TSP protocols

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.

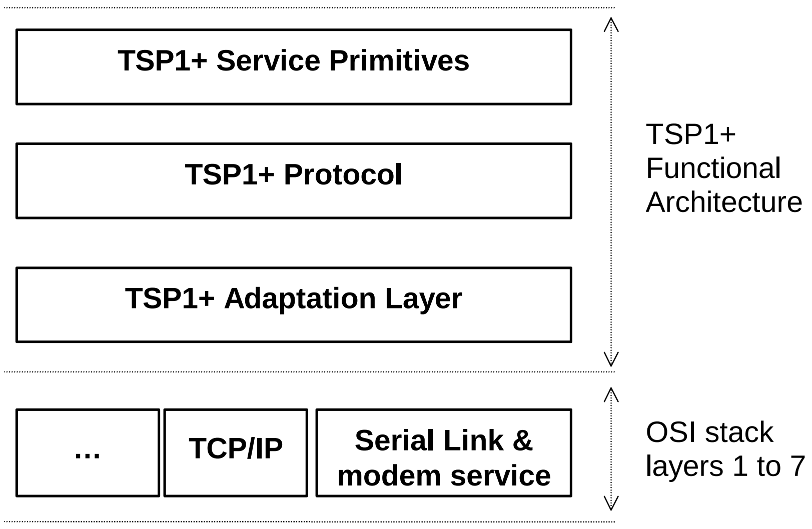


Figure B.4: TSP1+ functional architecture

Figure B.3shows 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 (figure 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 (figure 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.

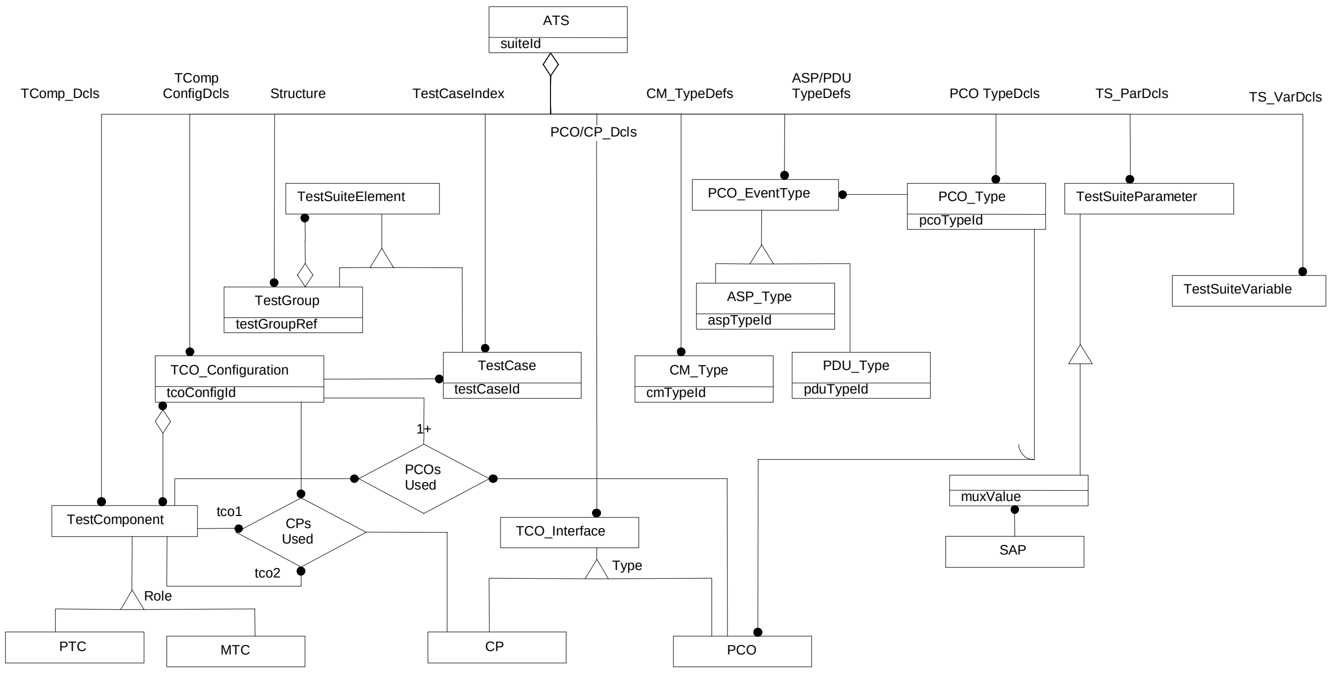


Figure C.1: Data declaration in TTCN model

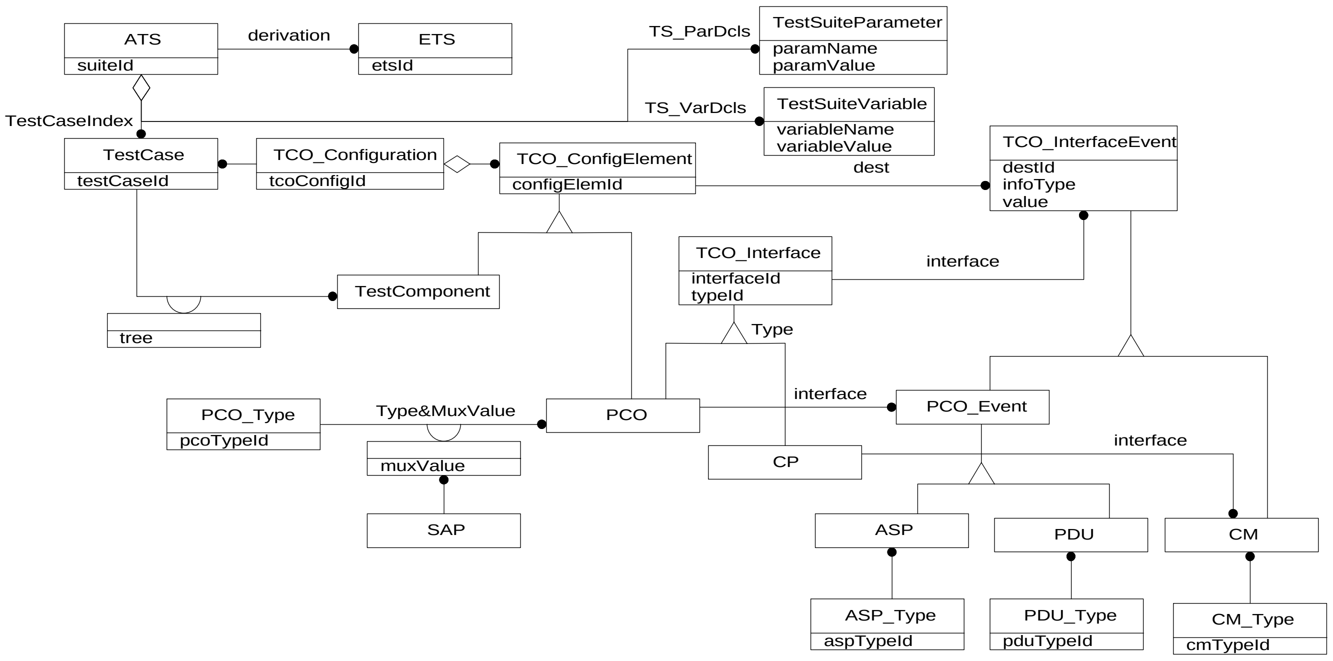


Figure C.2: Execution part in TTCN model

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.

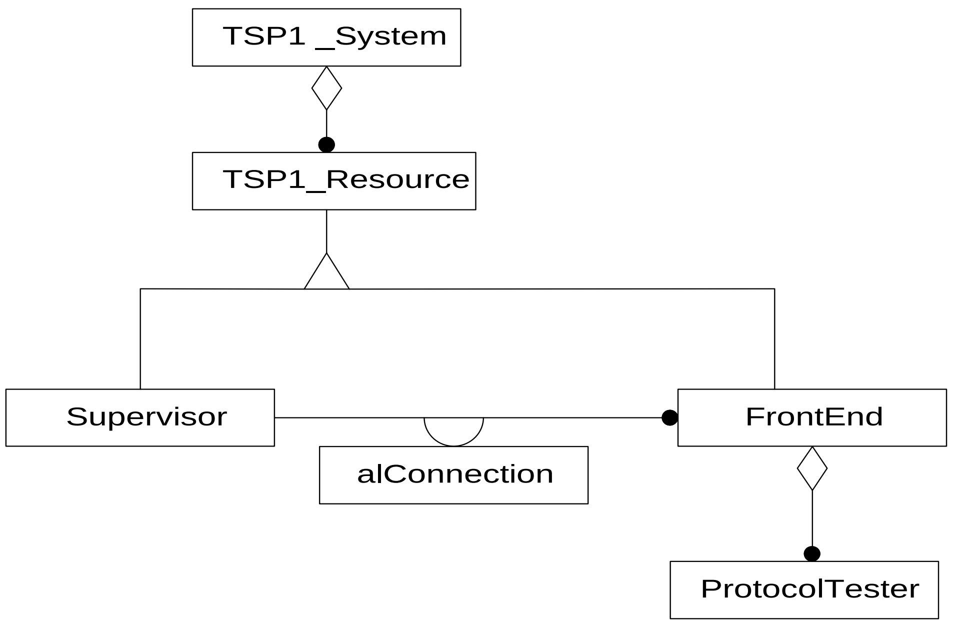


Figure C.3: System Model

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).

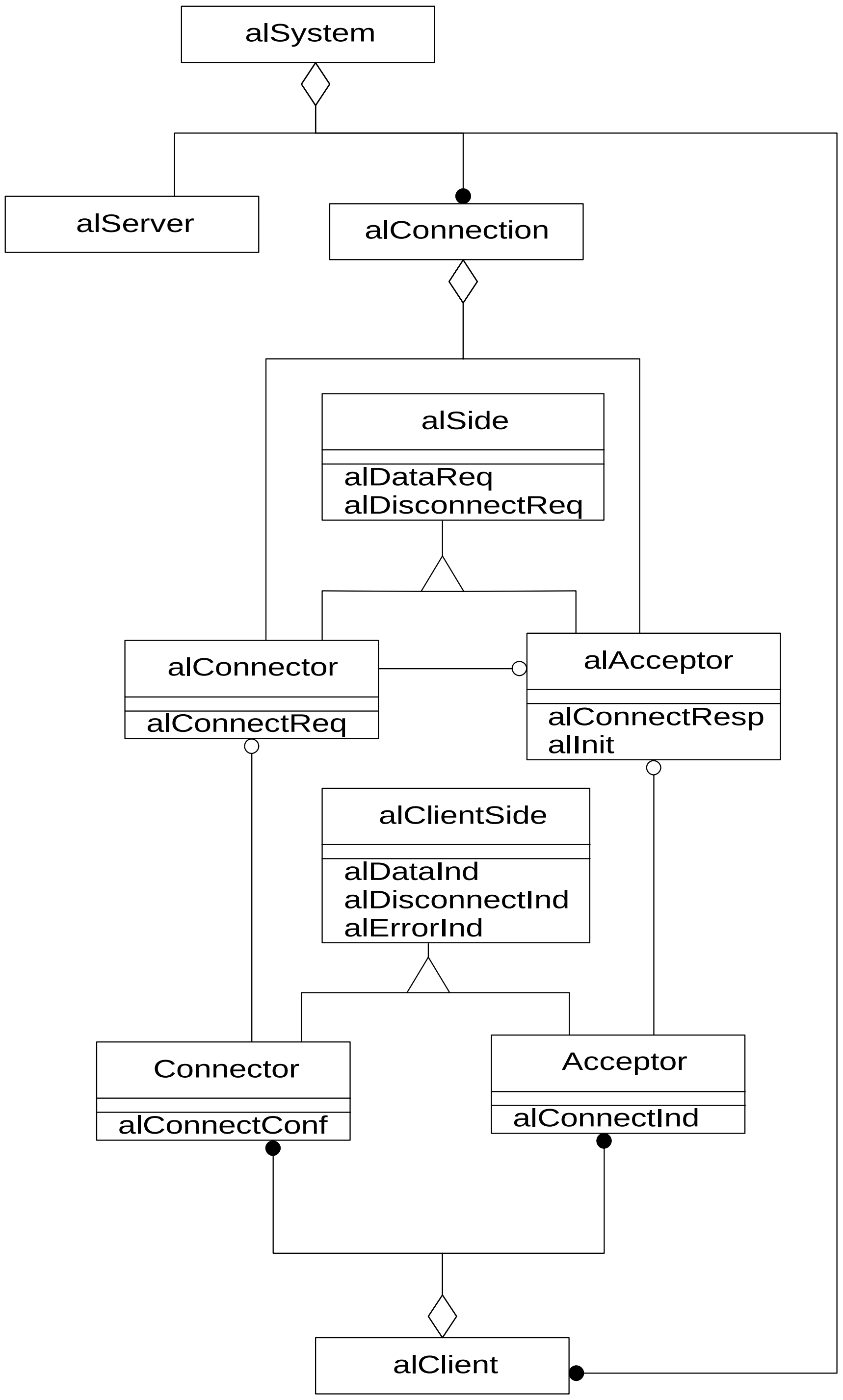


Figure C.4: Adaptation Layer Model

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:

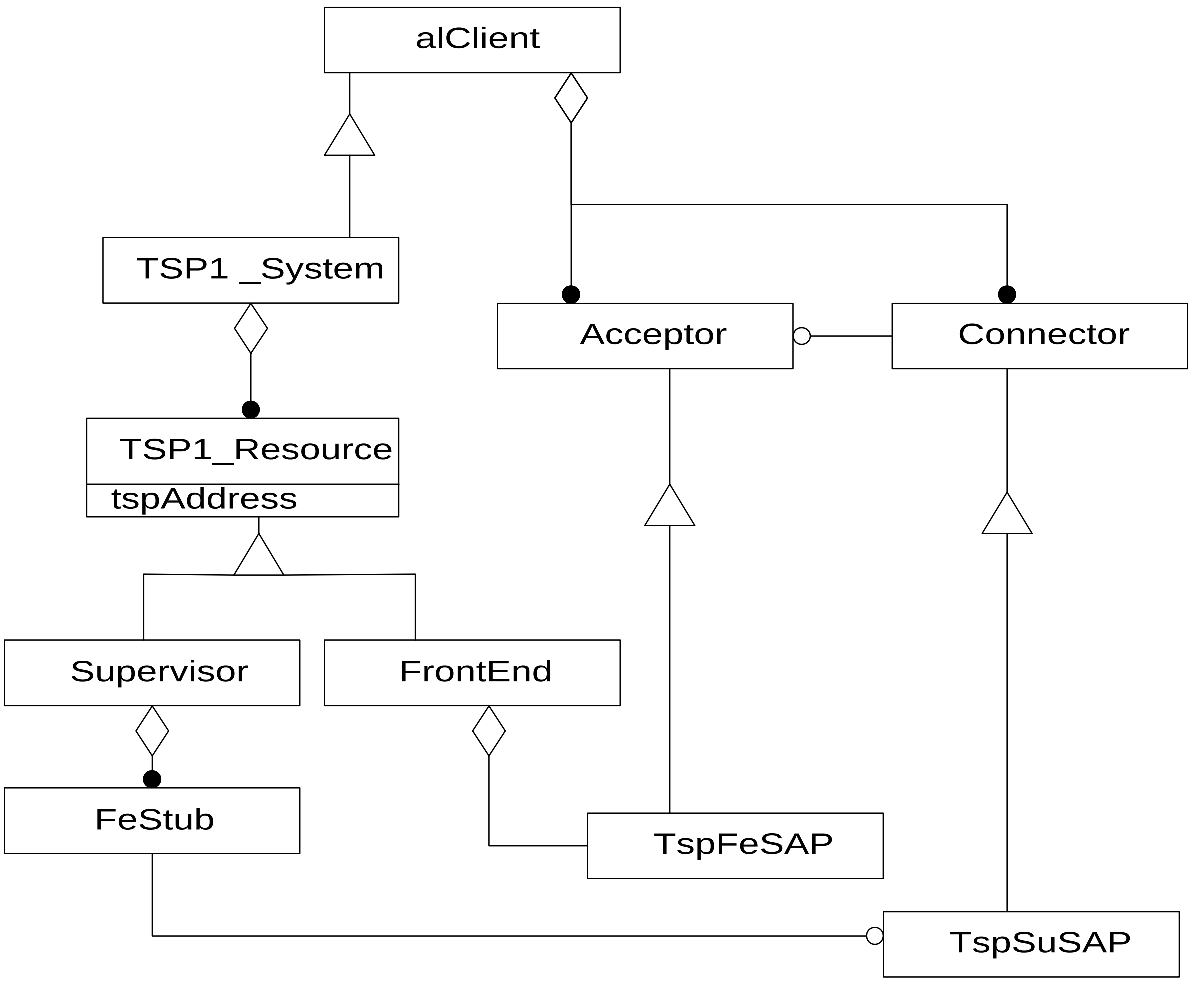


Figure C.5: TSP1+ system as an Adaptation Layer Client

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.

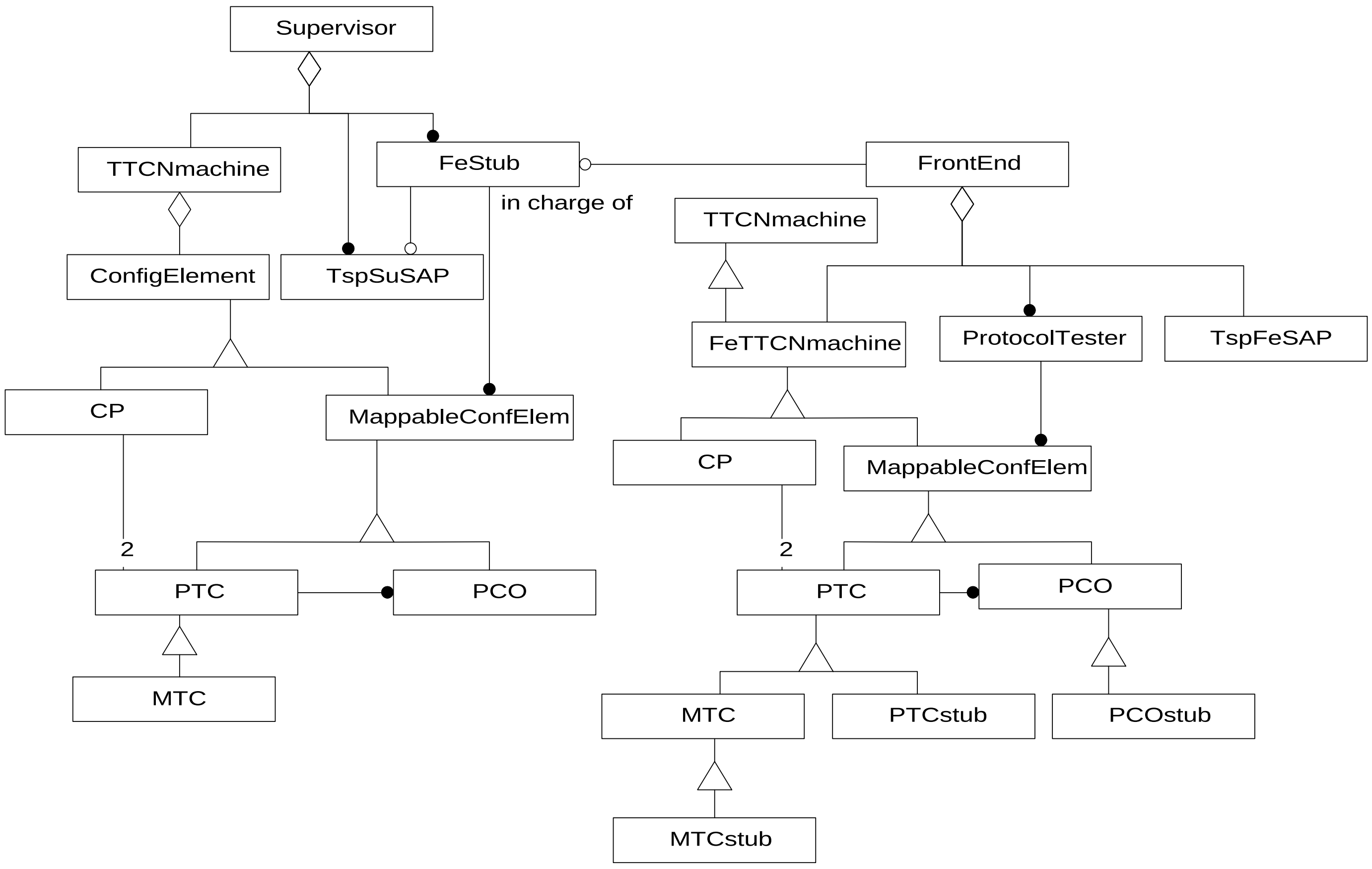


Figure C.6: Process Model

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.

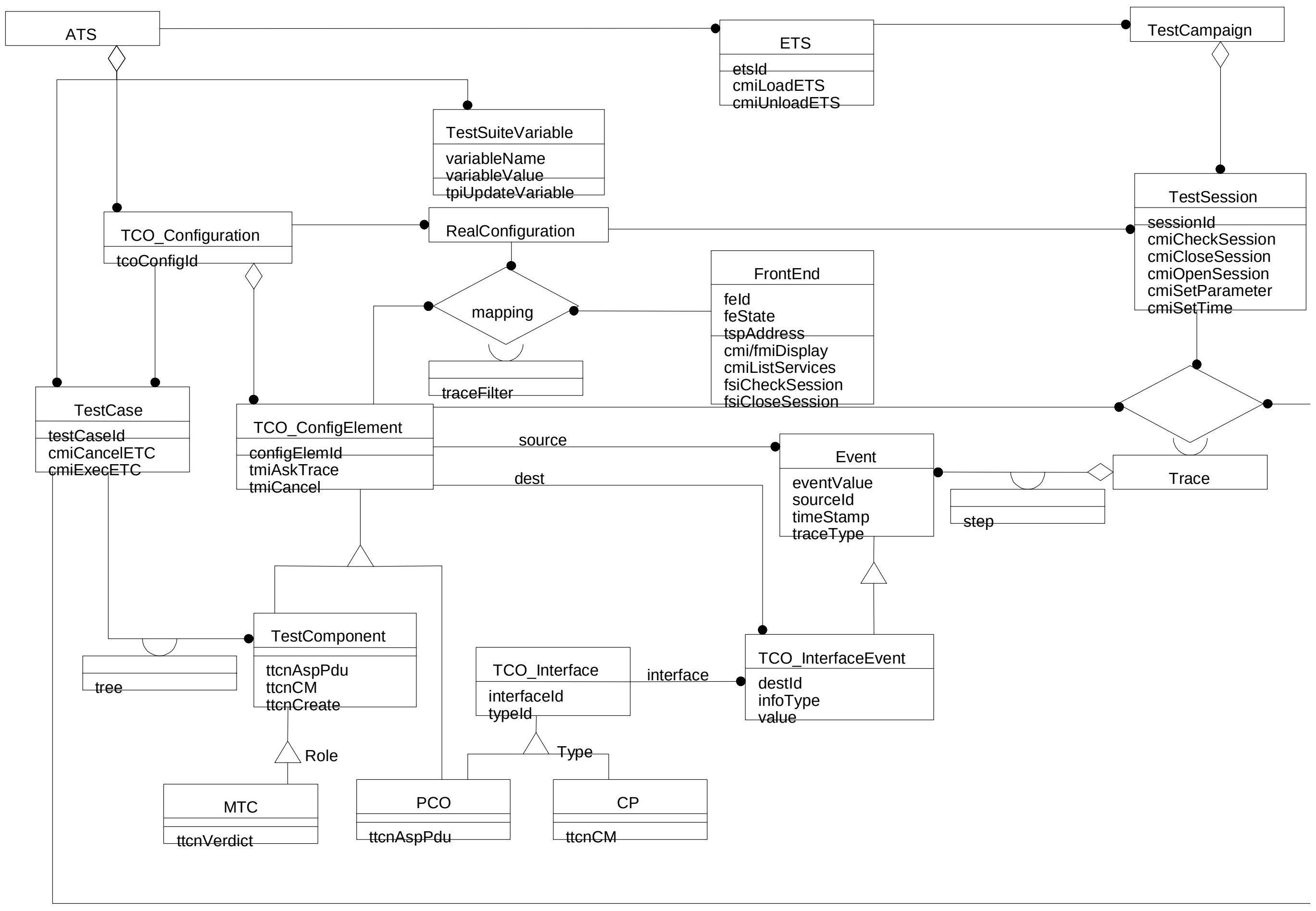


Figure C.7: Interface Model

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.

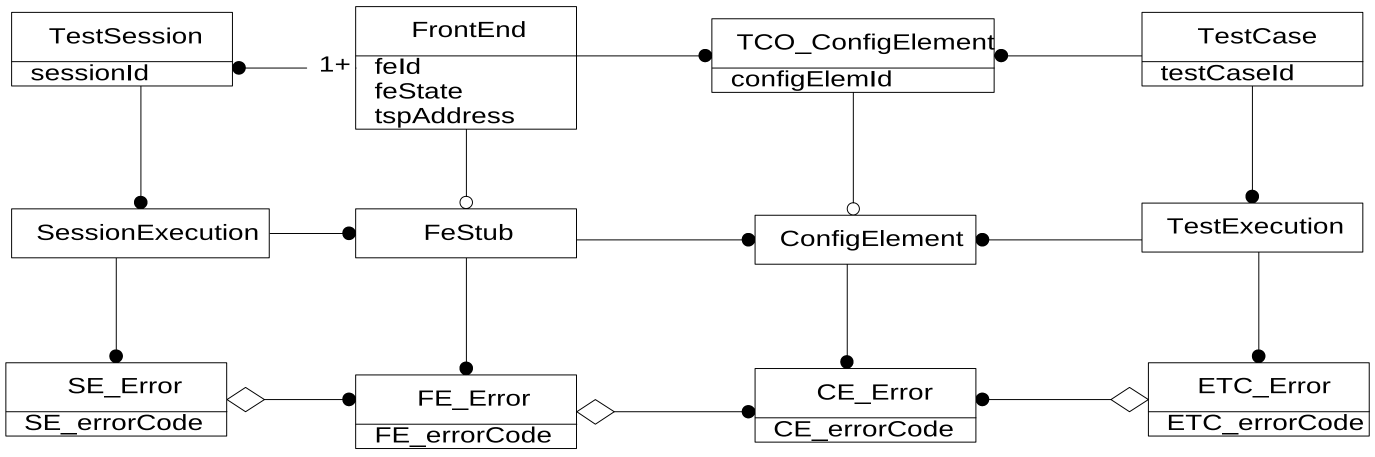


Figure C.8: Errors Model

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 the entire 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 a connection endpoint (like an OSI-SAP) where user processes can connect to from above. Table C-1 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 its 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 subclause 14.1).

C.8.2 Serial Port with modem

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

- Asynchronous mode, 8 bits, 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;
- Initialization 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 bits, no parity.

Annex D (informative): TSP1 SDL Model

tsp1_plus	rw	C:\SDL\TSP1v2sr\tsp1_plus.sdt
	rw	tsp1_plus.scu
	rw	C:\SDL\TSP1v2sr\
Signal and Data Descriptions		
TSPone	rw	TSP1plus.asn
TSP1_Messages	rw	tsp1_messages.sun
TSPone		
ASN1_Operators	rw	asn1_operators.sun
TSPone		
Environment_Messages	rw	environment_messages.sun
TSPone		
SDL System Structure		
TSP1_plus	rw	TSP1_plus.ssy
ASN1_Operators		
Environment_Messages		
TSP1_Messages		
TSPone		
SystemSupervisor	rw	system_supervisor.sbk
SystemSupervisor	rw	system_supervisor.spr
CloseCurrentSession	rw	closecurrentsession.spd
GetSessionParameters	rw	getsessionparameters.spd
GetTSP1LinkId	rw	gettsp1linkid.spd
GetFEforTCO	rw	getfefortco.spd
GetFEforConfigElement	rw	getfeforconfigelement.spd
Front_End	rw	front_end.sbk
Front_End	rw	front_end.spr
CheckForValidETS	rw	checkforvalidets.spd
CheckForValidSession	rw	checkforvalidsession.spd
CheckForValidTree	rw	checkforvalidtree.spd
CheckForValidTrace	rw	checkforvalidtrace.spd
CheckParameters	rw	checkparameters.spd
SetSessionParameters	rw	setsessionparameters.spd
SetFrontEndTime	rw	setfrontendtime.spd
CompileFEServiceList	rw	compilefeservicelist.spd
CheckMessageOrder	rw	checkmessageorder.spd
Destination_Front_End	rw	destination_front_end.sbk
Destination_Front_End	rw	destination_front_end.spr
CheckMessageSequence	rw	checkmessageorder1.spd
Macros		
Stop_Timer	rw	Stop_Timer.smc
Start_Timer	rw	Start_Timer.smc
Used Files		
Analysis Model		
Other Documents		

```
ASN.1 Text TSPone

TSPone DEFINITIONS ::=

 BEGIN

   EXPORTS TspAddress;


   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

   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                    ::= ENUMERATED  { first                   (1),
                                             current                 (0),
                                             last                    (2),
                                             first-last              (3) }

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

   TraceSet                ::= SEQUENCE    { conf-elem-id    ConfElemId,
                                             trace-filter    TraceFilter }
   TraceConfig             ::=SEQUENCE OF TraceSet
   TraceType               ::=PrintableString

   ServiceId               ::= ENUMERATED  { tsp1UnknownService      (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              ::= ENUMERATED  { intermediate-verdict    (0),
                                              final-verdict           (1) }
   VerdictValue             ::= ENUMERATED  { fail                    (0),
                                              inconc                  (1),
                                              pass                    (2) }


   Err-TSP1-INIT            ::= ENUMERATED  { errMOTNotReady          (1),
                                              errUnknownEts           (3),
                                              errUnknownSession       (4) }

   Err-TSP1-CHK-CONF        ::= ENUMERATED  { errMOTNotReady          (1),
                                              errMOTNotConnected      (2),
                                              errUnknownSession       (4) }

   Err-TSP1-SET-PARAMETER   ::= ENUMERATED  { errMOTNotReady          (1),
                                              errMOTNotConnected      (2),
                                              errInvalidVariable      (7) }

   Err-TSP1-SET-TIME        ::= ENUMERATED  { errMOTNotReady          (1),
                                              errMOTNotConnected      (2),
                                              errTimeNotAssigned      (5) }

   Err-TSP1-LIST-FE-SERVICES ::= ENUMERATED { errMOTNotConnected      (2) }

   Err-TSP1-CREATE          ::= ENUMERATED  { errMOTNotReady          (1),
                                              errMOTNotConnected      (2),
                                              errUnknownETS           (3),
                                              errTreeNotFound         (87) }

   Err-TSP1-INFO            ::= ENUMERATED  { errMOTNotConnected      (2),
                                              errOutOfSequenceMessage (86) }

   Err-TSP1-ASK-TRACE       ::= ENUMERATED  { errMOTNotReady          (1),
                                              errMOTNotConnected      (2),
                                              errTraceNotAvailable    (6) }

   Err-TSP1-END             ::= ENUMERATED  { errMOTNotReady          (1) }

   Err-TSP1-CANCEL-OP       ::= ENUMERATED  { errMOTNotReady          (1),
                                              errMOTNotConnected      (2) }

   Err-TSP1-TCO-FAILURE     ::= ENUMERATED  { errProcessingFailure  (170),
                                              errCommunicationLost  (200),
                                              errTesterCrash        (201) }

   ErrorCode                ::= CHOICE      { initErrs      [0] Err-TSP1-INIT,
                                              chkConfErrs   [1] Err-TSP1-CHK-CONF,
                                              setParsErrs   [2] Err-TSP1-SET-PARAMETER,
                                              setTimeErrs   [3] Err-TSP1-SET-TIME,
                                              listServErrs  [4] Err-TSP1-LIST-FE-SERVICES,
                                              createErrs    [5] Err-TSP1-CREATE,
                                              infoErrs      [6] Err-TSP1-INFO,
                                              askTraceErrs  [7] Err-TSP1-ASK-TRACE,
                                              endErrs       [8] Err-TSP1-END,
                                              cancelOpErrs  [9] Err-TSP1-CANCEL-OP }

   TSP1-ERR                 ::= SEQUENCE { service-id       ServiceId,
                                           err-code         ErrorCode,
                                           err-cause [0]    PrintableString OPTIONAL }
--                The content of 'err-cause' is manufacturer specific and therefore out --
--                of scope of this standard.                                            --

   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            Has no parameters

-- TSP1-INIT-COMPLETE       Has no parameters

-- TSP1-CHK-CONF            Has no parameters

-- TSP1-CHK-CONF-ACK        Has no parameters
```

```
   TSP1-SET-PARAMETER       ::= ParamList
```

```
-- TSP1-SET-PARAMETER-ACK   Has no parameters

   TSP1-SET-TIME            ::= GeneralizedTime

-- TSP1-SET-TIME-ACK        Has no parameters

-- TSP1-LIST-FE-SERVICES    Has no parameters

   TSP1-LIST-FE-SERVICES-ACK::= ServiceList

   TSP1-CREATE              ::= SEQUENCE { tco-id        [0] TcoId         OPTIONAL,
                                           test-case-id  [1] TestCaseId,
                                           tree              TreeId        OPTIONAL,
                                           param-list        ParamList     OPTIONAL }

   TSP1-CREATE-ACK          ::= 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           ::= 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 } OPTIONAL }

-- TSP1-END                 Has no parameters

-- TSP1-END-ACK             Has no parameters

   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 }

   TSP1-TCO-FAILURE         ::= SEQUENCE { fe-id             FeId,
                                           fault-code        Err-TSP1-TCO-FAILURE }

-- Definition of 'GeneralizedTime' which is not supported by Telelogic Tau.

  GeneralizedTime            ::= PrintableString

 END
```

```
Use TSPone;
```

PACKAGE TSP1_Messages 1(2)

```
/** TSP1+ Message Set (NORMATIVE) **/

SIGNAL tsp1_ERR (TSP1_ERR);
SIGNAL tsp1_TCO_FAILURE (TSP1_TCO_FAILURE);

SIGNAL tsp1_INIT (TSP1_INIT);
SIGNAL tsp1_INIT_ACK;
SIGNAL tsp1_INIT_COMPLETE;

SIGNAL tsp1_CHK_CONF;
SIGNAL tsp1_CHK_CONF_ACK;

SIGNAL tsp1_SET_PARAMETER (TSP1_SET_PARAMETER);
SIGNAL tsp1_SET_PARAMETER_ACK;

SIGNAL tsp1_SET_TIME (TSP1_SET_TIME);
SIGNAL tsp1_SET_TIME_ACK;

SIGNAL tsp1_LIST_FE_SERVICES;
SIGNAL tsp1_LIST_FE_SERVICES_ACK (TSP1_LIST_FE_SERVICES_ACK);

SIGNAL tsp1_CREATE (TSP1_CREATE);
SIGNAL tsp1_CREATE_ACK (TSP1_CREATE_ACK);

SIGNAL tsp1_INFO (TSP1_INFO);

SIGNAL tsp1_VERDICT (TSP1_UPDATE_VERDICT);
SIGNAL tsp1_UPDATE_VARIABLE (TSP1_UPDATE_VARIABLE);

SIGNAL tsp1_ASK_TRACE (TSP1_ASK_TRACE);
SIGNAL tsp1_ASK_TRACE_ACK (TSP1_ASK_TRACE_ACK);

SIGNAL tsp1_END;
SIGNAL tsp1_END_ACK;

SIGNAL tsp1_CANCEL_OP (TSP1_CANCEL_OP);
SIGNAL tsp1_CANCEL_OP_ACK (TSP1_CANCEL_OP_ACK);

SIGNAL tsp1_DISPLAY (TSP1_DISPLAY);
```

Use TSPone;

PACKAGE TSP1_Messages 2(2)

```
/** TSP1+ Signal Groupings : Front End to System Supervisor **/

SIGNALLIST FE_to_SS = tsp1_ERR,
                      tsp1_TCO_FAILURE,
                      tsp1_INIT_ACK,
                      tsp1_INIT_COMPLETE,
                      tsp1_CHK_CONF_ACK,
                      tsp1_SET_PARAMETER_ACK,
                      tsp1_SET_TIME_ACK,
                      tsp1_LIST_FE_SERVICES_ACK,
                      tsp1_CREATE_ACK,
                      tsp1_INFO,
                      tsp1_VERDICT,
                      tsp1_UPDATE_VARIABLE,
                      tsp1_ASK_TRACE_ACK,
                      tsp1_END_ACK,
                      tsp1_CANCEL_OP_ACK,
                      tsp1_DISPLAY;
```

```
/** TSP1+ Signal Groupings : System Supervisor to Front End **/

SIGNALLIST SS_to_FE = tsp1_INIT,
                      tsp1_CHK_CONF,
                      tsp1_SET_PARAMETER,
                      tsp1_SET_TIME,
                      tsp1_LIST_FE_SERVICES,
                      tsp1_CREATE,
                      tsp1_INFO,
                      tsp1_UPDATE_VARIABLE,
                      tsp1_ASK_TRACE,
                      tsp1_END,
                      tsp1_CANCEL_OP,
                      tsp1_DISPLAY;
```

USE TSPone;

PACKAGE ASN1_Operators 1(7)

```
/** This PACKAGE contains the SDL
    OPERATORS necessary for adding
    and extracting data elements to and
    from variables based on complex
    ASN.1 strctures  **/
```

```
USE TSPone;
```

PACKAGE ASN1_Operators 2(7)

```
/*** Definition of TSP1-ERR Operators  ***/

NEWTYPE TSP1_ERR_operators
  OPERATORS
      Add_service_id_From: ServiceId, TSP1_ERR -> TSP1_ERR;
      Add_err_code_From:   ErrorCode, TSP1_ERR -> TSP1_ERR;

     OPERATOR Add_service_id_From;
         FPAR     ServId     ServiceId,
                  TSP1ERR_I  TSP1_ERR;
         RETURNS  TSP1ERR_O  TSP1_ERR;
         START;
             TASK TSP1ERR_O := TSP1ERR_I;
             TASK TSP1ERR_O!service_id := ServId;
         RETURN;
      ENDOPERATOR Add_service_id_From;

     OPERATOR Add_err_code_From;
         FPAR     ErrCode    ErrorCode,
                  TSP1ERR_I  TSP1_ERR;
         RETURNS  TSP1ERR_O  TSP1_ERR;
         START;
             TASK TSP1ERR_O := TSP1ERR_I;
             TASK TSP1ERR_O!err_code := ErrCode;
         RETURN;
      ENDOPERATOR Add_err_code_From;

ENDNEWTYPE TSP1_ERR_operators;
```

```
/*** Definition of TSP1-TCO-FAILURE Operators ***/

NEWTYPE TSP1_TCO_FAILURE_operators
  OPERATORS
      Add_fe_id_From:      FeId, TSP1_TCO_FAILURE -> TSP1_TCO_FAILURE;
      Add_fault_code_From: Err_TSP1_TCO_FAILURE, TSP1_TCO_FAILURE -> TSP1_TCO_FAILURE;

      OPERATOR Add_fe_id_From;
         FPAR     FE         FeId,
                  TcoFail_I  TSP1_TCO_FAILURE;
         RETURNS  TcoFail_O  TSP1_TCO_FAILURE;
         START;
             TASK TcoFail_O := TcoFail_I;
             TASK TcoFail_O!fe_id := FE;
         RETURN;
      ENDOPERATOR Add_fe_id_From;

      OPERATOR Add_fault_code_From;
         FPAR     FCode      Err_TSP1_TCO_FAILURE,
                  TcoFail_I  TSP1_TCO_FAILURE;
         RETURNS  TcoFail_O  TSP1_TCO_FAILURE;
         START;
             TASK TcoFail_O := TcoFail_I;
             TASK TcoFail_O!fault_code := FCode;
         RETURN;
      ENDOPERATOR Add_fault_code_From;

ENDNEWTYPE TSP1_TCO_FAILURE_operators;
```

```
USE TSPone;
```

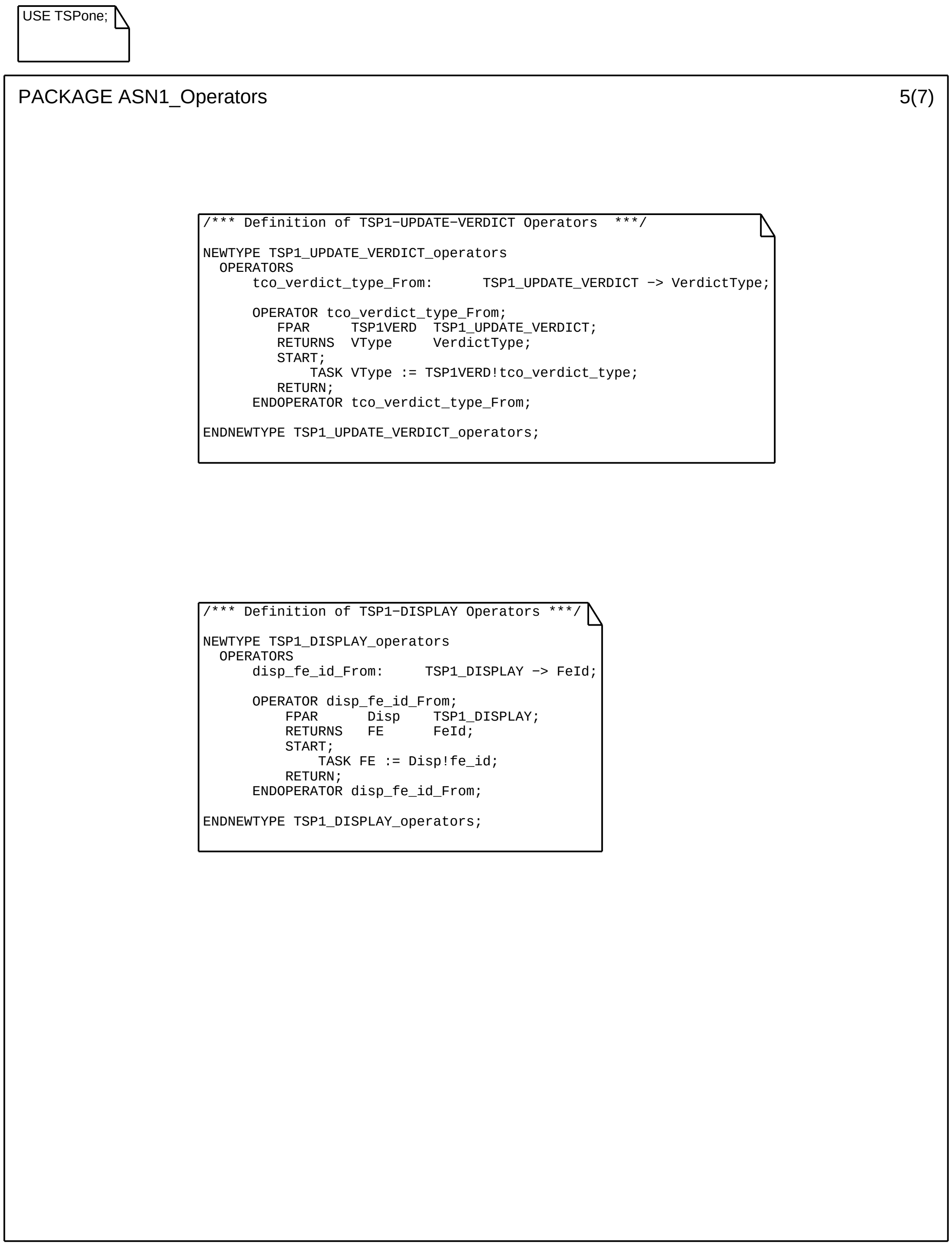
PACKAGE ASN1_Operators 3(7)

```
/*** Definition of TSP1-INIT Operators (1) ***/

NEWTYPE TSP1_INIT_out_operators
  OPERATORS
      ets_id_From:            TSP1_INIT -> EtsId;
      session_id_From:        TSP1_INIT -> SessionId;
      real_config_From:       TSP1_INIT -> RealConfig;
      trace_config_From:      TSP1_INIT -> TraceConfig;

      OPERATOR ets_id_From;
         FPAR     TSP1INIT  TSP1_INIT;
         RETURNS  ETS   EtsId;
         START;
             TASK ETS := TSP1INIT!ets_id;
         RETURN;
      ENDOPERATOR ets_id_From;

      OPERATOR session_id_From;
         FPAR     TSP1INIT  TSP1_INIT;
         RETURNS  Sess  SessionId;
         START;
             TASK Sess := TSP1INIT!session_id;
         RETURN;
      ENDOPERATOR session_id_From;

      OPERATOR real_config_From;
         FPAR     TSP1INIT  TSP1_INIT;
         RETURNS  RConf RealConfig;
         START;
             TASK RConf := TSP1INIT!real_config;
         RETURN;
      ENDOPERATOR real_config_From;

      OPERATOR trace_config_From;
         FPAR     TSP1INIT  TSP1_INIT;
         RETURNS  TConf TraceConfig;
         START;
             TASK TConf := TSP1INIT!trace_config;
         RETURN;
      ENDOPERATOR trace_config_From;

ENDNEWTYPE TSP1_INIT_out_operators;
```

USE TSPone;

PACKAGE ASN1_Operators 4(7)

```
/*** Definition of TSP1−INIT Operators (2) ***/

NEWTYPE TSP1_INIT_in_operators
  OPERATORS
      Add_ets_id_From:            EtsId, TSP1_INIT −> TSP1_INIT;
      Add_session_id_From:        SessionId, TSP1_INIT −> TSP1_INIT;
      Add_conf_elem_id_list_From: ConfElemIdList, TSP1_INIT −> TSP1_INIT;
      Add_real_config_From:       RealConfig, TSP1_INIT −> TSP1_INIT;
      Add_trace_config_From:      TraceConfig, TSP1_INIT −> TSP1_INIT;

     OPERATOR Add_ets_id_From;
         FPAR     ETS        EtsId,
                  TSP1INIT_I TSP1_INIT;
         RETURNS  TSP1INIT_O TSP1_INIT;
         START;
             TASK TSP1INIT_O := TSP1INIT_I;
             TASK TSP1INIT_O!ets_id := ETS;
         RETURN;
      ENDOPERATOR Add_ets_id_From;

     OPERATOR Add_session_id_From;
         FPAR     Sess       SessionId,
                  TSP1INIT_I TSP1_INIT;
         RETURNS  TSP1INIT_O TSP1_INIT;
         START;
             TASK TSP1INIT_O := TSP1INIT_I;
             TASK TSP1INIT_O!session_id := Sess;
         RETURN;
      ENDOPERATOR Add_session_id_From;

     OPERATOR Add_conf_elem_id_list_From;
         FPAR     CEIlist     ConfElemIdList,
                  TSP1INIT_I TSP1_INIT;
         RETURNS  TSP1INIT_O TSP1_INIT;
         START;
             TASK TSP1INIT_O := TSP1INIT_I;
             TASK TSP1INIT_O!conf_elem_id_list := CEIlist;
         RETURN;
      ENDOPERATOR Add_conf_elem_id_list_From;

     OPERATOR Add_real_config_From;
         FPAR     RConf      RealConfig,
                  TSP1INIT_I TSP1_INIT;
         RETURNS  TSP1INIT_O TSP1_INIT;
         START;
             TASK TSP1INIT_O := TSP1INIT_I;
             TASK TSP1INIT_O!real_config := RConf;
         RETURN;
      ENDOPERATOR Add_real_config_From;

     OPERATOR Add_trace_config_From;
         FPAR     TConf      TraceConfig,
                  TSP1INIT_I TSP1_INIT;
         RETURNS  TSP1INIT_O TSP1_INIT;
         START;
             TASK TSP1INIT_O := TSP1INIT_I;
             TASK TSP1INIT_O!trace_config := TConf;
         RETURN;
      ENDOPERATOR Add_trace_config_From;

ENDNEWTYPE TSP1_INIT_in_operators;
```



USE TSPone;

PACKAGE ASN1_Operators 5(7)

```
/*** Definition of TSP1-UPDATE-VERDICT Operators  ***/

NEWTYPE TSP1_UPDATE_VERDICT_operators
  OPERATORS
      tco_verdict_type_From:      TSP1_UPDATE_VERDICT -> VerdictType;

      OPERATOR tco_verdict_type_From;
         FPAR     TSP1VERD  TSP1_UPDATE_VERDICT;
         RETURNS  VType     VerdictType;
         START;
             TASK VType := TSP1VERD!tco_verdict_type;
         RETURN;
      ENDOPERATOR tco_verdict_type_From;

ENDNEWTYPE TSP1_UPDATE_VERDICT_operators;
```

```
/*** Definition of TSP1-DISPLAY Operators ***/

NEWTYPE TSP1_DISPLAY_operators
  OPERATORS
      disp_fe_id_From:     TSP1_DISPLAY -> FeId;

      OPERATOR disp_fe_id_From;
          FPAR      Disp    TSP1_DISPLAY;
          RETURNS   FE      FeId;
          START;
              TASK FE := Disp!fe_id;
          RETURN;
      ENDOPERATOR disp_fe_id_From;

ENDNEWTYPE TSP1_DISPLAY_operators;
```

USE TSPone;

PACKAGE ASN1_Operators 6(7)

```
/*** Definition of TSP1−CANCEL−OP Operators ***/

NEWTYPE TSP1_CANCEL_OP_operators
  OPERATORS
     conf_elem_id_From:       TSP1_CANCEL_OP −> ConfElemId;

OPERATOR conf_elem_id_From;
        FPAR        CANC    TSP1_CANCEL_OP;
        RETURNS     Co      ConfElemId;
        START;
            TASK Co := CANC!conf_elem_id;
        RETURN;
     ENDOPERATOR conf_elem_id_From;

ENDNEWTYPE TSP1_CANCEL_OP_operators;
```

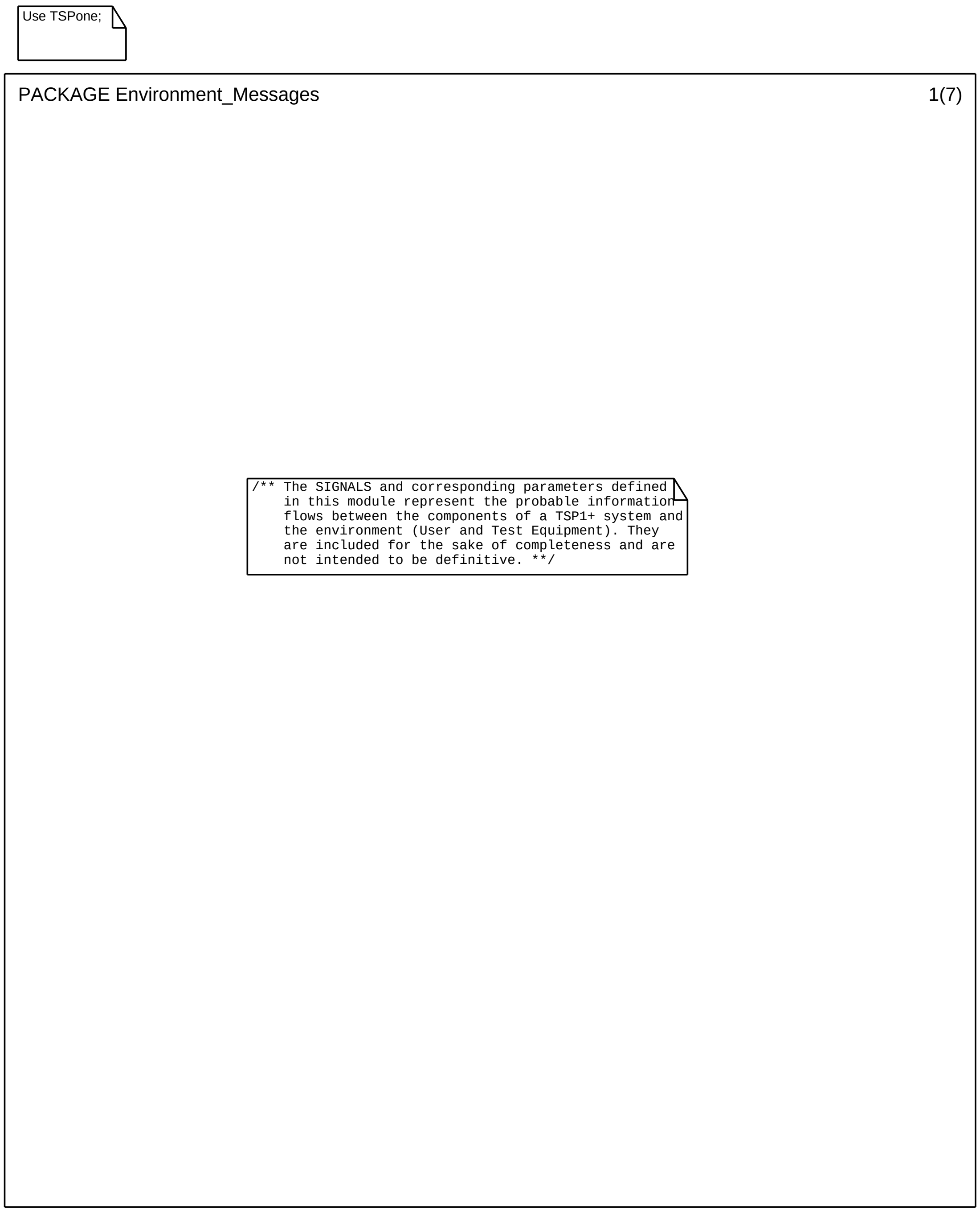
```
/*** Definition of TSP1−CANCEL−OP−ACK Operators ***/

NEWTYPE TSP1_CANCEL_OP_ACK_operators
  OPERATORS
     Add_conf_elem_id_ack_From: ConfElemId, TSP1_CANCEL_OP_ACK −> TSP1_CANCEL_OP_ACK;

OPERATOR Add_conf_elem_id_ack_From;
        FPAR        Co      ConfElemId,
                    CANC_I  TSP1_CANCEL_OP_ACK;
        RETURNS     CANC_O  TSP1_CANCEL_OP_ACK;
        START;
            TASK CANC_O := CANC_I;
            TASK CANC_O!conf_elem_id := Co;
        RETURN;
     ENDOPERATOR Add_conf_elem_id_ack_From;

ENDNEWTYPE TSP1_CANCEL_OP_ACK_operators;
```

USE TSPone;

PACKAGE ASN1_Operators 7(7)

```
/*** Definition of TSP1-INFO Operators (1) ***/

NEWTYPE TSP1_INFO_out_operators
  OPERATORS
     dest_id_From:          TSP1_INFO -> ConfElemId;

OPERATOR dest_id_From;
        FPAR        INFO    TSP1_INFO;
        RETURNS     Co      ConfElemId;
        START;
            TASK Co := INFO!dest_id;
        RETURN;
     ENDOPERATOR dest_id_From;

ENDNEWTYPE TSP1_INFO_out_operators;
```

```
/*** Definition of TSP1-CREATE Operators (1) ***/

NEWTYPE TSP1_CREATE_out_operators
  OPERATORS
     tco_id_From:            TSP1_CREATE -> TcoId;
     tree_From:              TSP1_CREATE -> TreeId;

OPERATOR tco_id_From;
        FPAR        CREA    TSP1_CREATE;
        RETURNS     Tc      TcoId;
        START;
            TASK Tc := CREA!tco_id;
        RETURN;
     ENDOPERATOR tco_id_From;

OPERATOR tree_From;
        FPAR        CREA    TSP1_CREATE;
        RETURNS     Tr      TreeId;
        START;
            TASK Tr := CREA!tree;
        RETURN;
     ENDOPERATOR tree_From;

ENDNEWTYPE TSP1_CREATE_out_operators;
```



Use TSPone;

PACKAGE Environment_Messages 1(7)

```
/** The SIGNALS and corresponding parameters defined
    in this module represent the probable information
    flows between the components of a TSP1+ system and
    the environment (User and Test Equipment). They
    are included for the sake of completeness and are
    not intended to be definitive. **/
```

```
Use TSPone;
```

PACKAGE Environment_Messages 2(7)

```
/** Messages between the User and the System Supervisor **/
/**                      (Non-Normative)                 **/

SIGNAL start_session (StartSession);
SIGNAL session_started;

SIGNAL verify_config;
SIGNAL verify_config_ack;
SIGNAL change_parameter (TSP1_SET_PARAMETER);
SIGNAL parameter_changed;

SIGNAL set_clock (TSP1_SET_TIME);
SIGNAL clock_set;

SIGNAL list_services (FeId);
SIGNAL service_list (TSP1_LIST_FE_SERVICES_ACK);

SIGNAL start_tco (TSP1_CREATE);
SIGNAL tco_started;

SIGNAL test_verdict (TSP1_UPDATE_VERDICT);

SIGNAL modified_variable (TSP1_UPDATE_VARIABLE);

SIGNAL provide_trace (TSP1_ASK_TRACE);
SIGNAL provide_trace_ack (TSP1_ASK_TRACE_ACK);

SIGNAL close_session;
SIGNAL session_closed;

SIGNAL stop_test (TSP1_CANCEL_OP);
SIGNAL test_stopped;

SIGNAL display_info (TSP1_DISPLAY);

SIGNAL error_report (TSP1_ERR);
SIGNAL system_fault (TSP1_TCO_FAILURE);
```

Use TSPone;

PACKAGE Environment_Messages 3(7)

```
/** Messages between the Front-end and Test Components **/
/**                    (Non-Normative)                 **/

SIGNAL initialize_tester (InitializeTester);
SIGNAL tester_initialized;

SIGNAL verify_config_status;
SIGNAL verify_config_status_ack (EtsId);

SIGNAL modify_parameter (TSP1_SET_PARAMETER);
SIGNAL parameter_modified;

SIGNAL set_time (TSP1_SET_TIME);
SIGNAL time_set (BOOLEAN);

SIGNAL start_tester;
SIGNAL tester_started (BOOLEAN);

SIGNAL send_info (TSP1_INFO);
SIGNAL info_sent;

SIGNAL send_verdict (TSP1_UPDATE_VERDICT);

SIGNAL variable_modified (TSP1_UPDATE_VARIABLE);

SIGNAL request_trace;
SIGNAL request_trace_ack (TSP1_ASK_TRACE_ACK);

SIGNAL terminate_session;
SIGNAL session_terminated;

SIGNAL terminate_test;
SIGNAL test_terminated;

SIGNAL tester_fault (TSP1_TCO_FAILURE);
```

```
Use TSPone;
```

PACKAGE Environment_Messages 4(7)

```
/** Signal groupings for messages passing between **/
/** the User and the System Supervisor            **/

SIGNALLIST User_to_SS =     start_session,
                            verify_config,
                            change_parameter,
                            set_clock,
                            list_services,
                            start_tco,
                            provide_trace,
                            close_session,
                            stop_test,
                            display_info;

SIGNALLIST SS_to_User =     session_started,
                            verify_config_ack,
                            parameter_changed,
                            clock_set,
                            service_list,
                            tco_started,
                            test_verdict,
                            modified_variable,
                            provide_trace_ack,
                            session_closed,
                            test_stopped,
                            display_info,
                            error_report,
                            system_fault;
```

Use TSPone;

PACKAGE Environment_Messages 5(7)

```
/** Signal groupings for messages passing between **/
/** a Front-End and a TCO                         **/

SIGNALLIST FE_to_TCO =      initialize_tester,
                            verify_config_status,
                            modify_parameter,
                            set_time,
                            start_tester,
                            send_info,
                            info_sent,
                            request_trace,
                            terminate_session,
                            terminate_test;

SIGNALLIST TCO_to_FE =      tester_initialized,
                            verify_config_status_ack,
                            parameter_modified,
                            time_set,
                            tester_started,
                            send_info,
                            send_verdict,
                            variable_modified,
                            request_trace_ack,
                            session_terminated,
                            test_terminated,
                            tester_fault;
```

```
Use TSPone;
```

PACKAGE Environment_Messages 6(7)

```
/** General Data Sort Descriptions **/

SYNTYPE
  Integer8 = INTEGER CONSTANTS 0:255
ENDSYNTYPE Integer8;
```

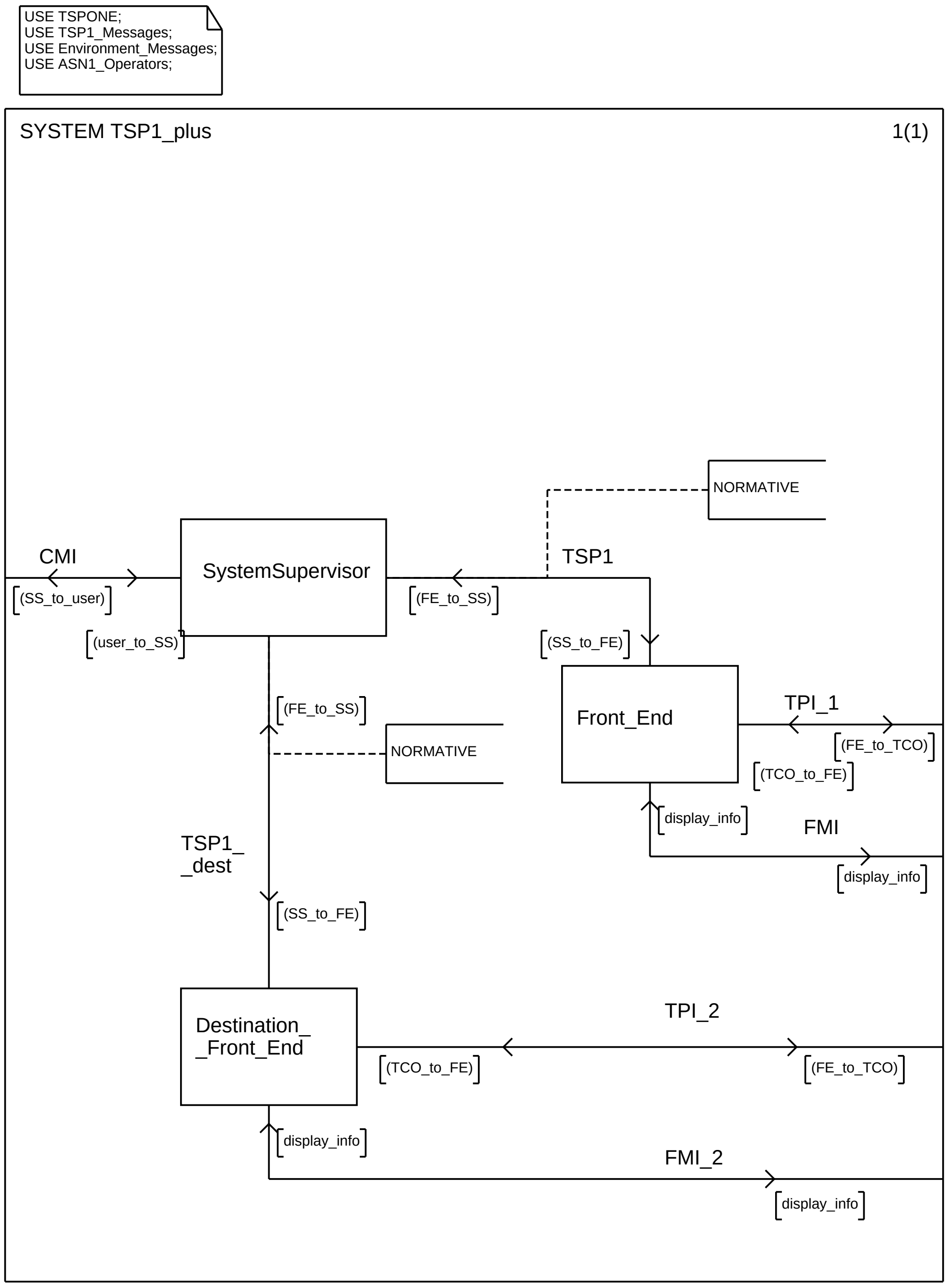
```
/* The basic time units for the timers in
 * this system are seconds */

SYNONYM
  sec Duration = 1.0;
```

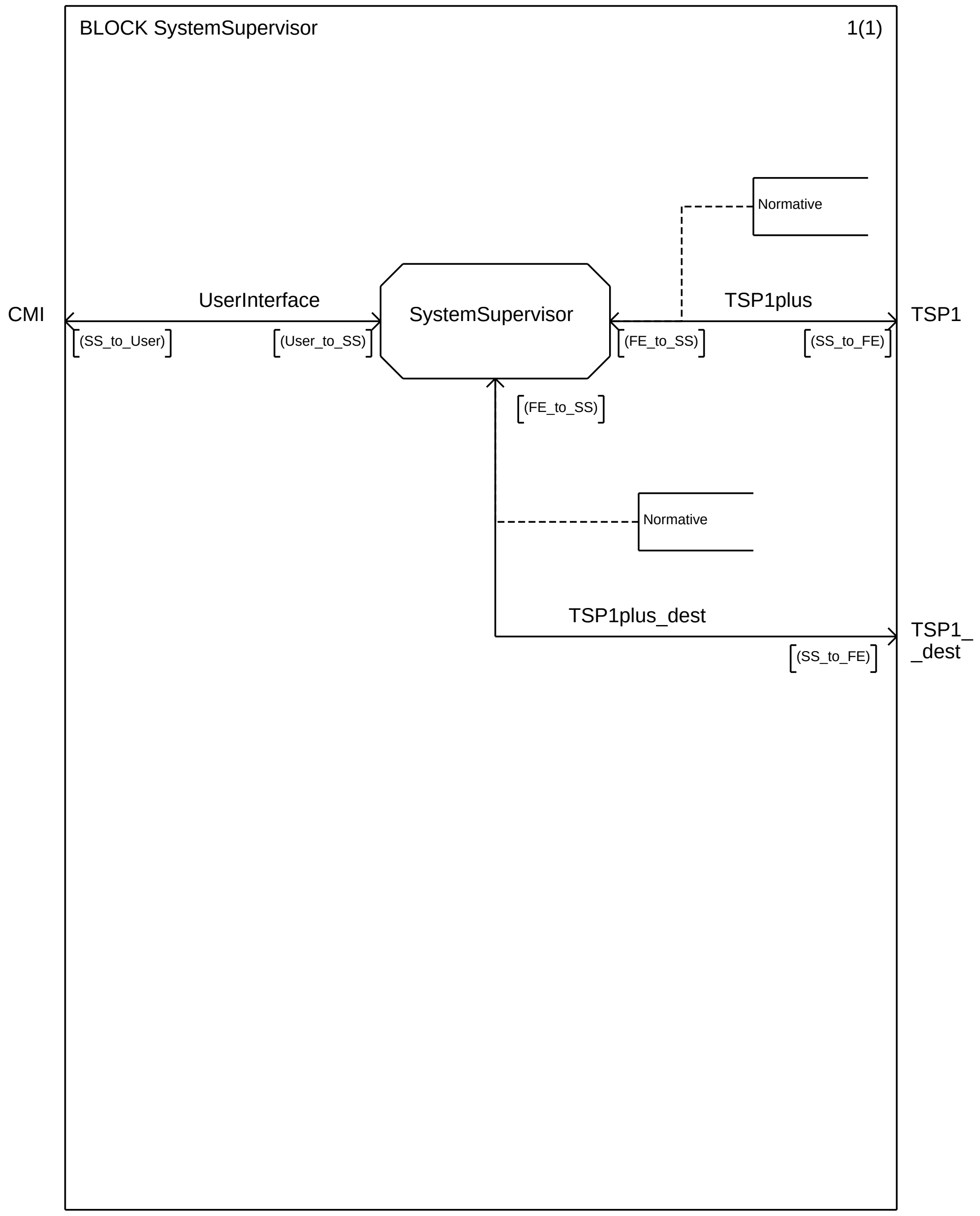
```
/** Message parameters for User<>SS signals **/

SYNTYPE
  StartSession = SessionId
ENDSYNTYPE STartSession;
```

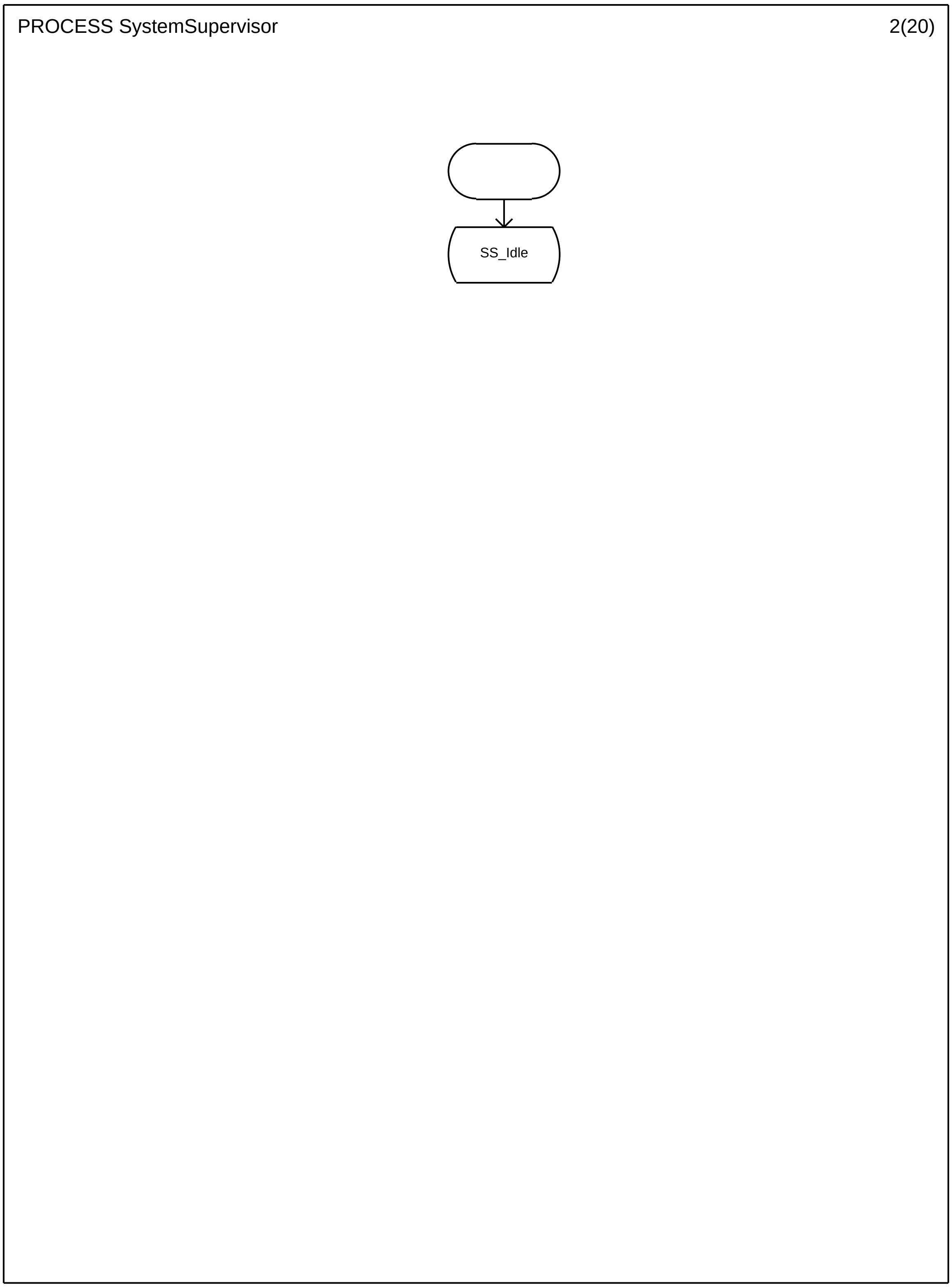
Use TSPone;

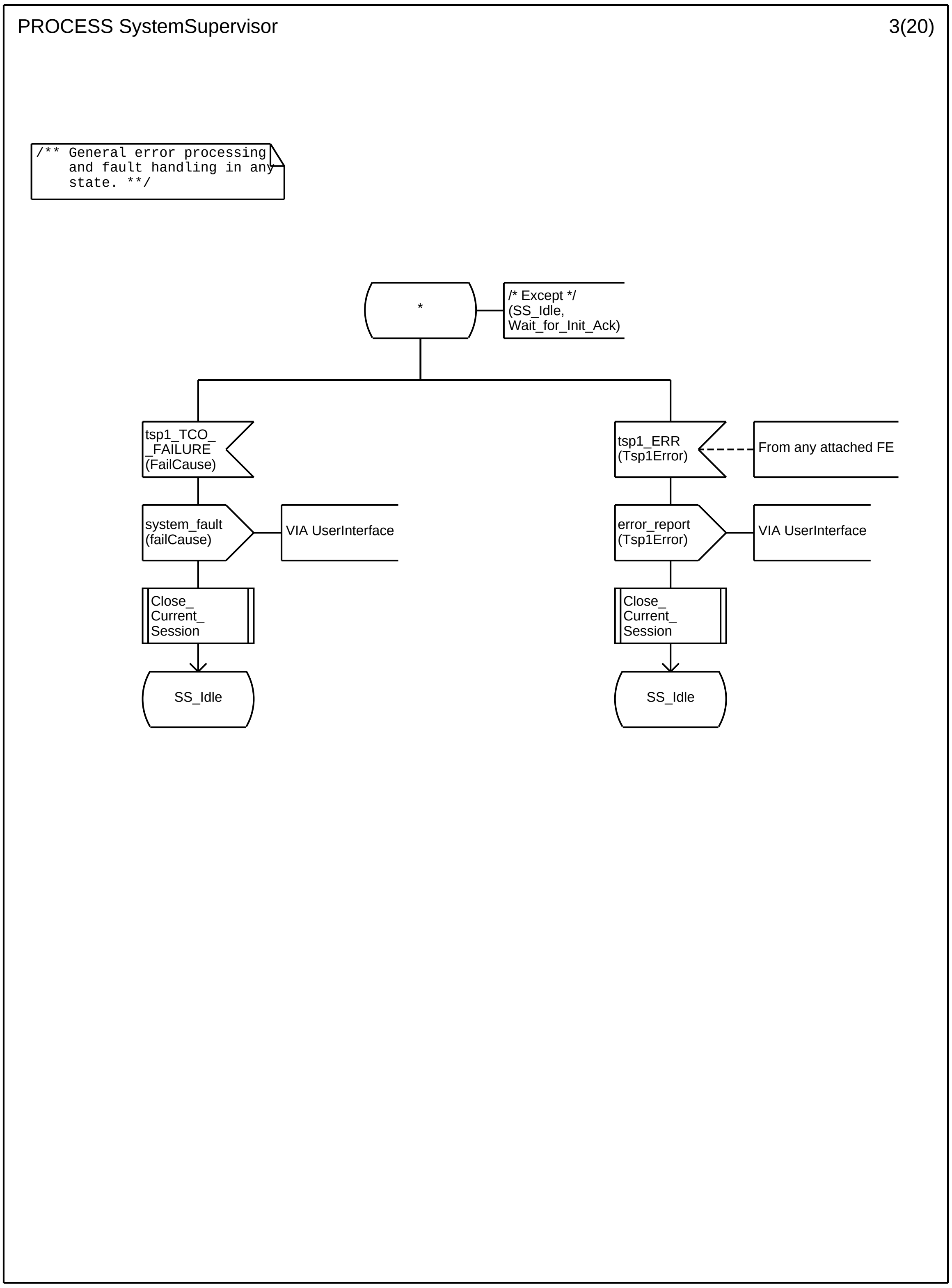
PACKAGE Environment_Messages 7(7)

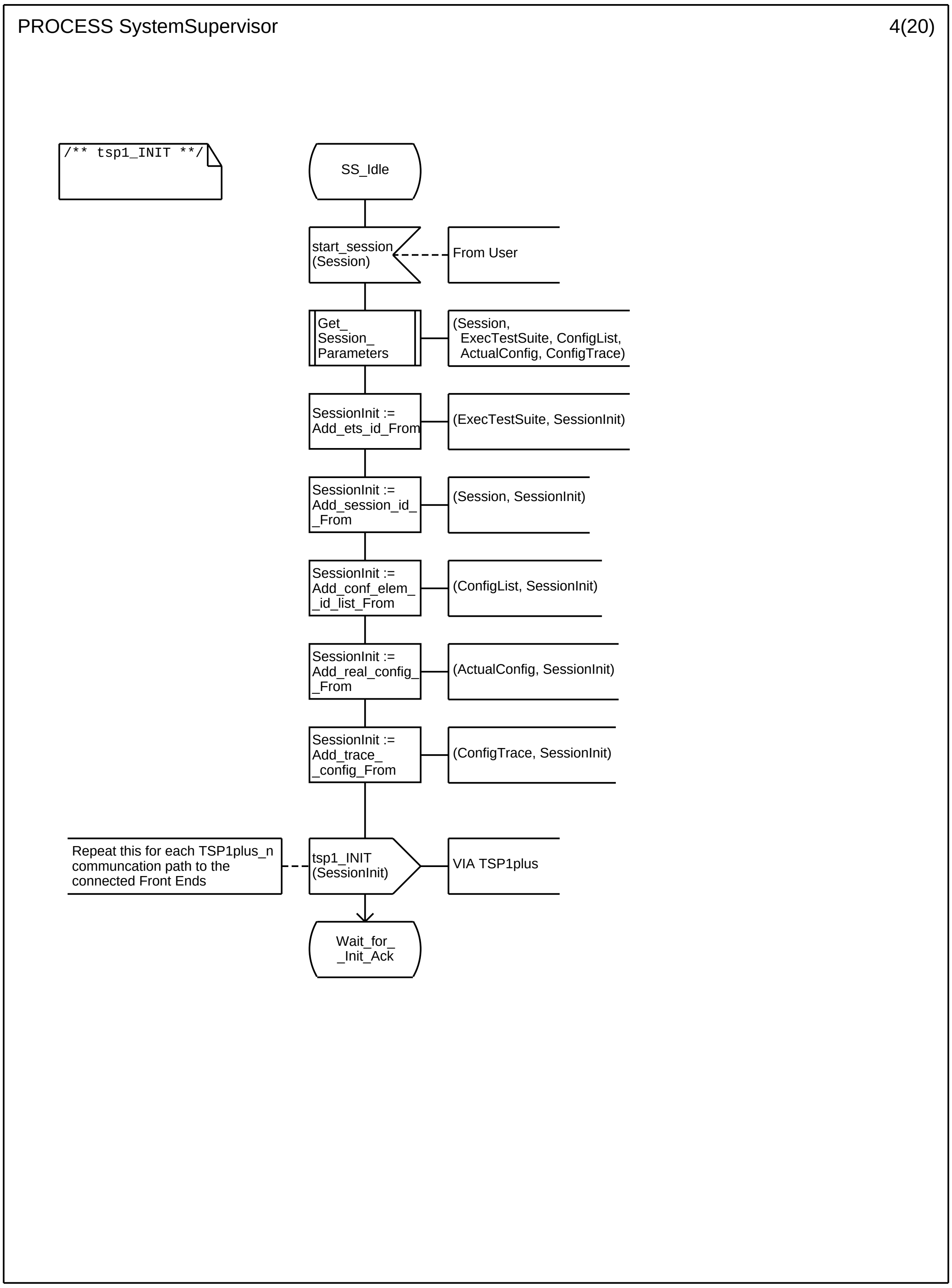
```
/** Message parameters for FE<>TCO Signals (1)**/

NEWTYPE InitializeTester
  STRUCT
     ets_ident           EtsId;
     real_configuration  RealConfig;
     trace_configuration TraceConfig;
  OPERATORS
     Add_ets_ident_From:            EtsId, Initializetester -> Initializetester;
     Add_real_configuration_From:   RealConfig, Initializetester -> Initializetester;
     Add_trace_configuration_From:  TraceConfig, Initializetester -> Initializetester;

     OPERATOR Add_ets_ident_From;
        FPAR        Et      EtsId,
                    init_I  InitializeTester;
        RETURNS     init_O  InitializeTester;
        START;
            TASK init_O := init_I;
            TASK init_O!ets_ident := Et;
        RETURN;
     ENDOPERATOR Add_ets_ident_From;

     OPERATOR Add_real_configuration_From;
        FPAR        Re      RealConfig,
                    init_I  InitializeTester;
        RETURNS     init_O  InitializeTester;
        START;
            TASK init_O := init_I;
            TASK init_O!real_configuration := Re;
        RETURN;
     ENDOPERATOR Add_real_configuration_From;

     OPERATOR Add_trace_configuration_From;
        FPAR        Tr      TraceConfig,
                    init_I  InitializeTester;
        RETURNS     init_O  InitializeTester;
        START;
            TASK init_O := init_I;
            TASK init_O!trace_configuration := Tr;
        RETURN;
     ENDOPERATOR Add_trace_configuration_From;

ENDNEWTYPE InitializeTester;
```

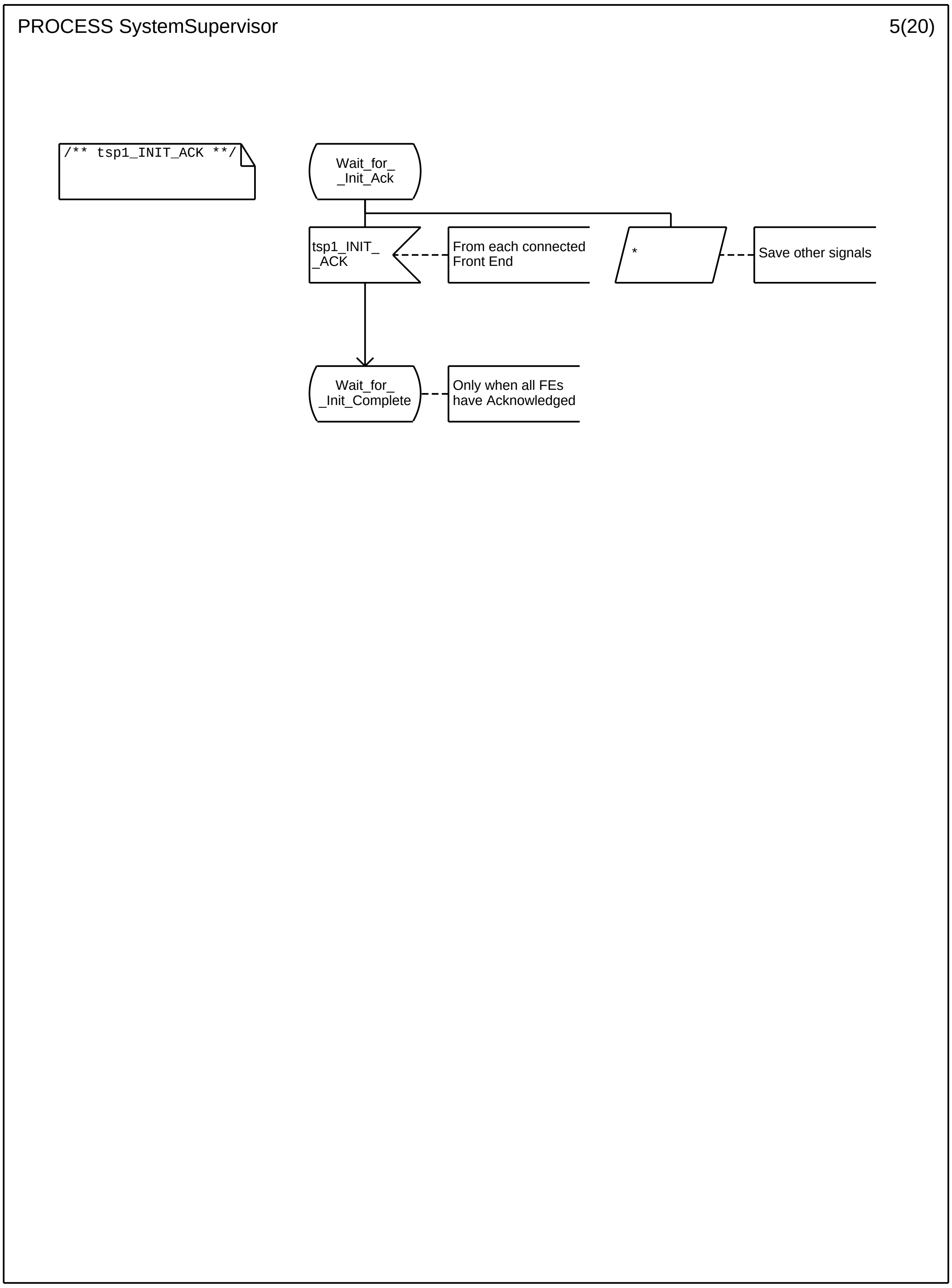

USE TSPONE;
USE TSP1_Messages;
USE Environment_Messages;
USE ASN1_Operators;
SYSTEM TSP1_plus
1(1)
NORMATIVE
CMI
SystemSupervisor
TSP1
(SS_to_user)
(FE_to_SS)
(user_to_SS)
(SS_to_FE)
(FE_to_SS)
Front_End
TPI_1
NORMATIVE
(FE_to_TCO)
(TCO_to_FE)
display_info
FMI
TSP1_
_dest
display_info
(SS_to_FE)
Destination_
_Front_End
TPI_2
(TCO_to_FE)
(FE_to_TCO)
display_info
FMI_2
display_info

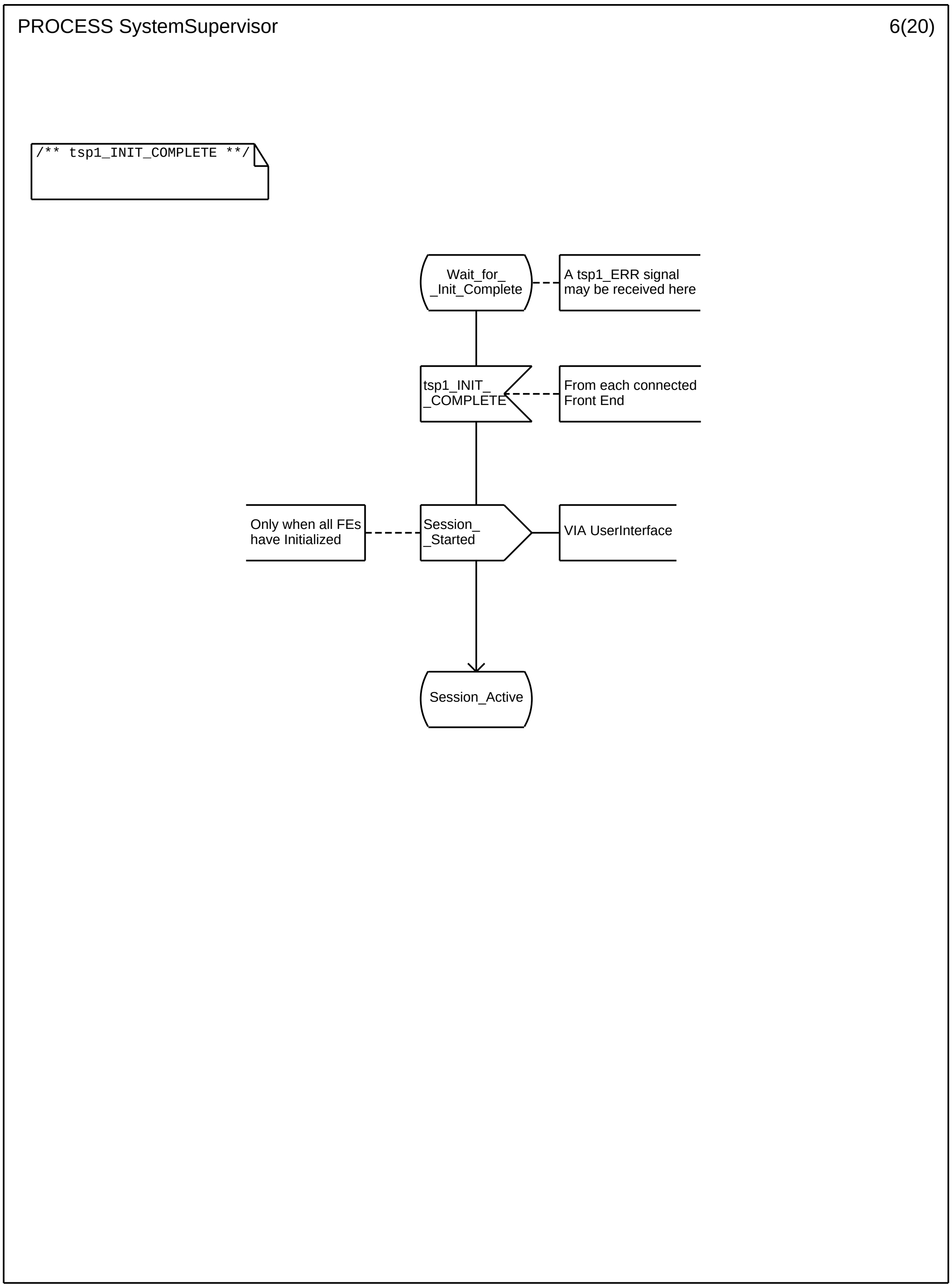

BLOCK SystemSupervisor
1(1)
Normative
CMI
UserInterface
SystemSupervisor
TSP1plus
TSP1
(SS_to_User)
(User_to_SS)
(FE_to_SS)
(SS_to_FE)
(FE_to_SS)
Normative
TSP1plus_dest
TSP1_
_dest
(SS_to_FE)

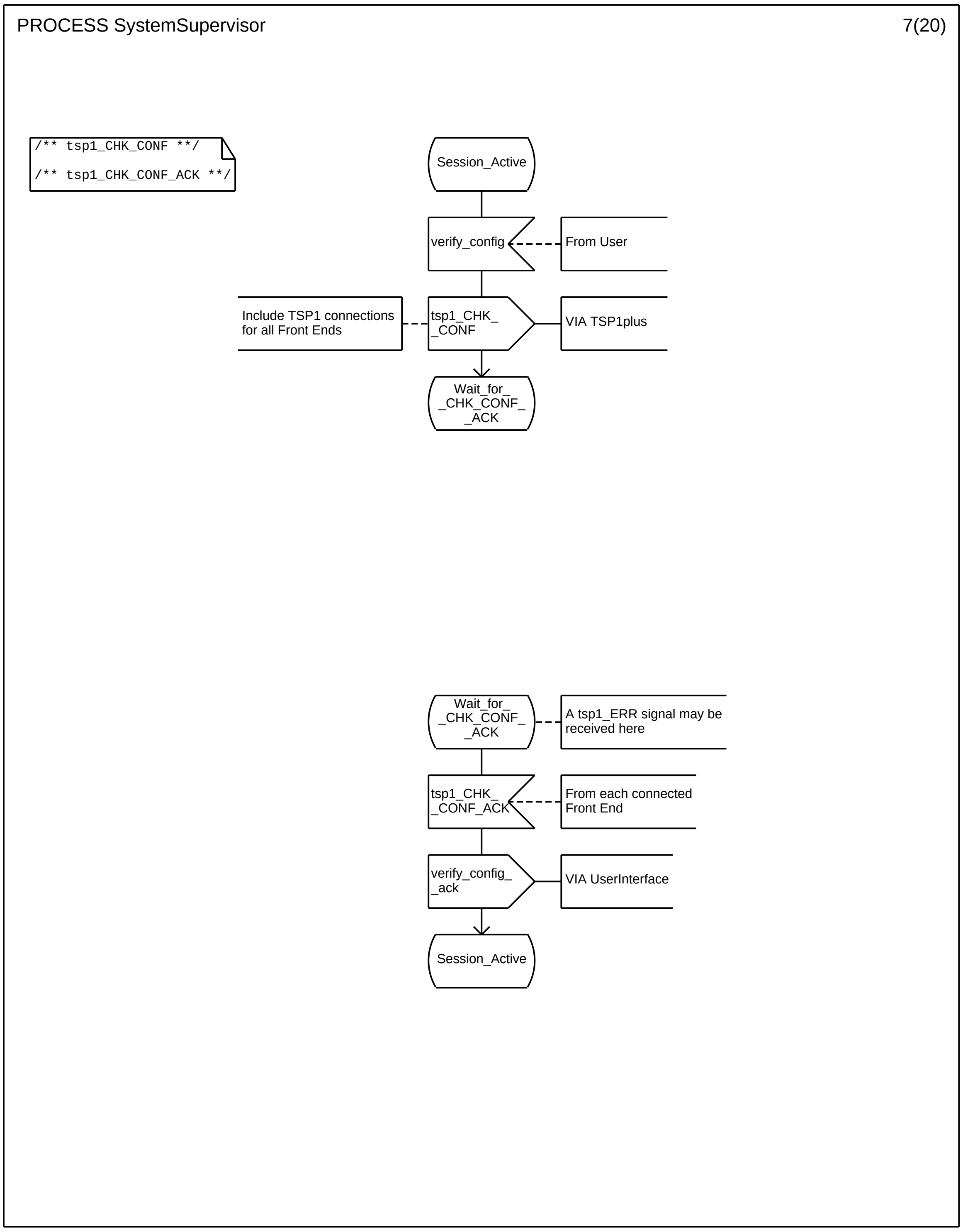

PROCESS SystemSupervisor
2(20)
SS_Idle

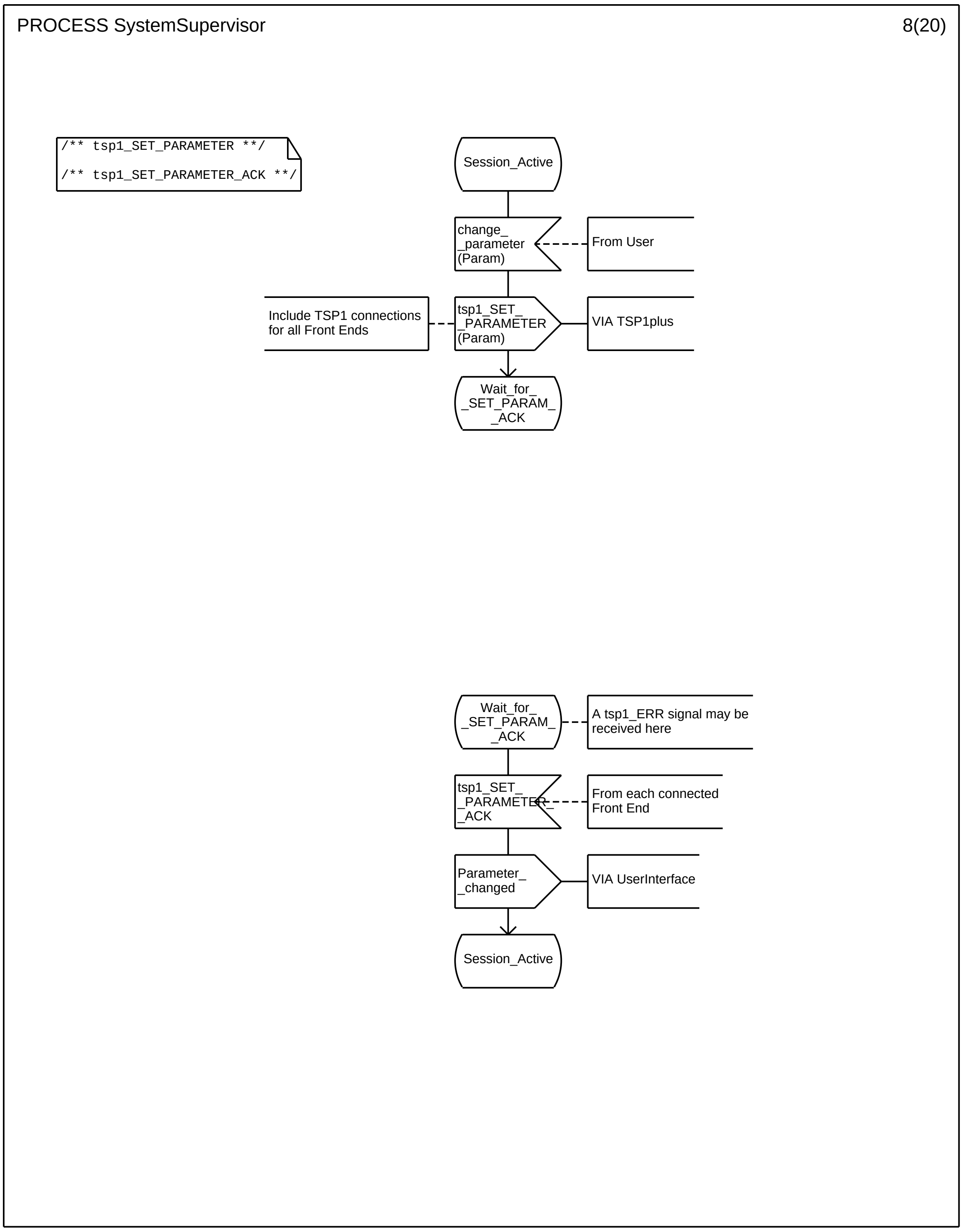

PROCESS SystemSupervisor
3(20)
/** General error processing and fault handling in any state. **/
*
/* Except */ (SS_Idle, Wait_for_Init_Ack)
tsp1_TCO_ _FAILURE (FailCause)
system_fault (failCause)
VIA UserInterface
Close_ Current_ Session
SS_Idle
tsp1_ERR (Tsp1Error)
From any attached FE
error_report (Tsp1Error)
VIA UserInterface
Close_ Current_ Session
SS_Idle

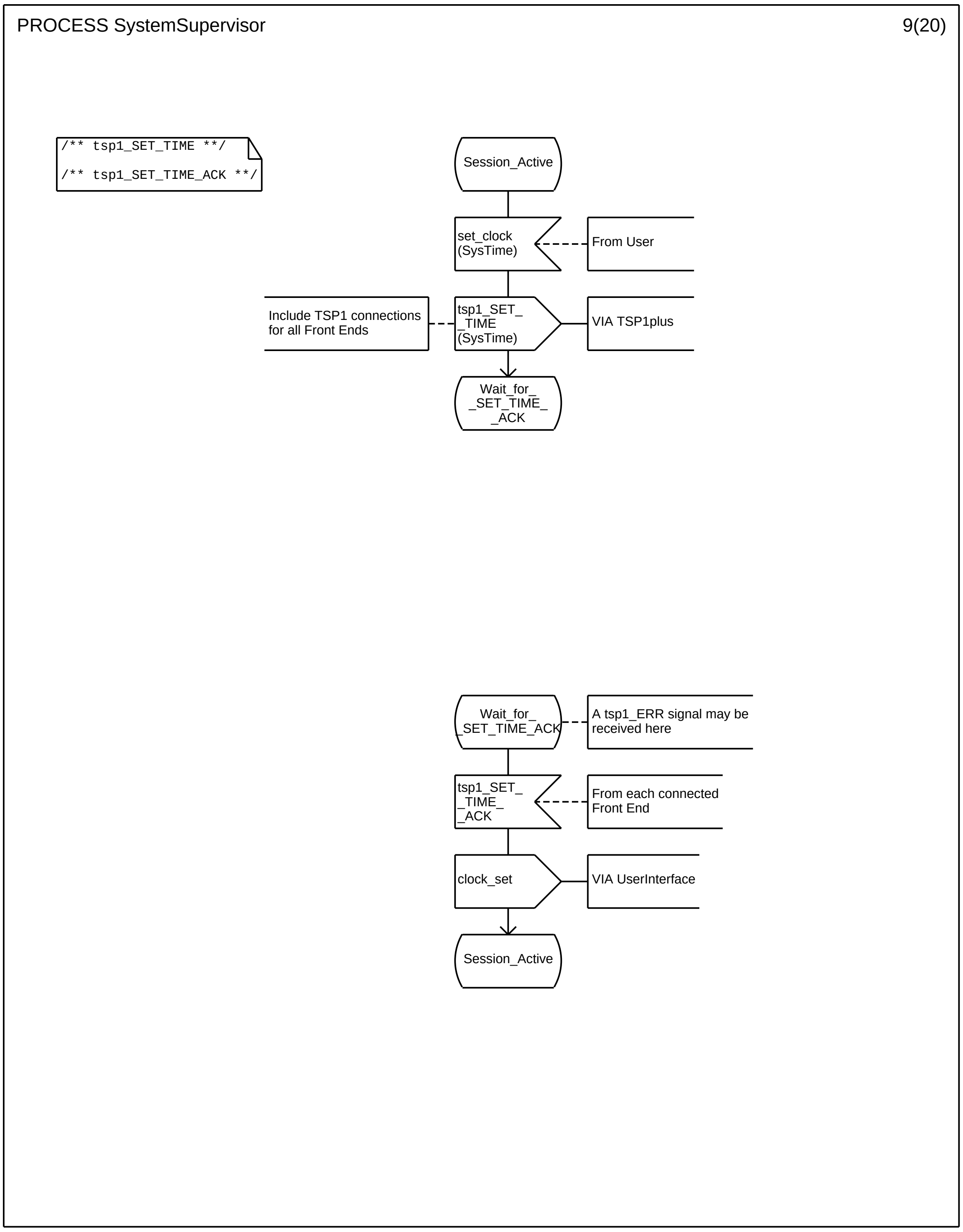

PROCESS SystemSupervisor
4(20)
/** tsp1_INIT **/
SS_Idle
start_session (Session)
From User
Get_ Session_ Parameters
(Session, ExecTestSuite, ConfigList, ActualConfig, ConfigTrace)
SessionInit := Add_ets_id_From
(ExecTestSuite, SessionInit)
SessionInit := Add_session_id_ _From
(Session, SessionInit)
SessionInit := Add_conf_elem_ _id_list_From
(ConfigList, SessionInit)
SessionInit := Add_real_config_ _From
(ActualConfig, SessionInit)
SessionInit := Add_trace_ _config_From
(ConfigTrace, SessionInit)
Repeat this for each TSP1plus_n communcation path to the connected Front Ends
tsp1_INIT (SessionInit)
VIA TSP1plus
Wait_for_ _Init_Ack

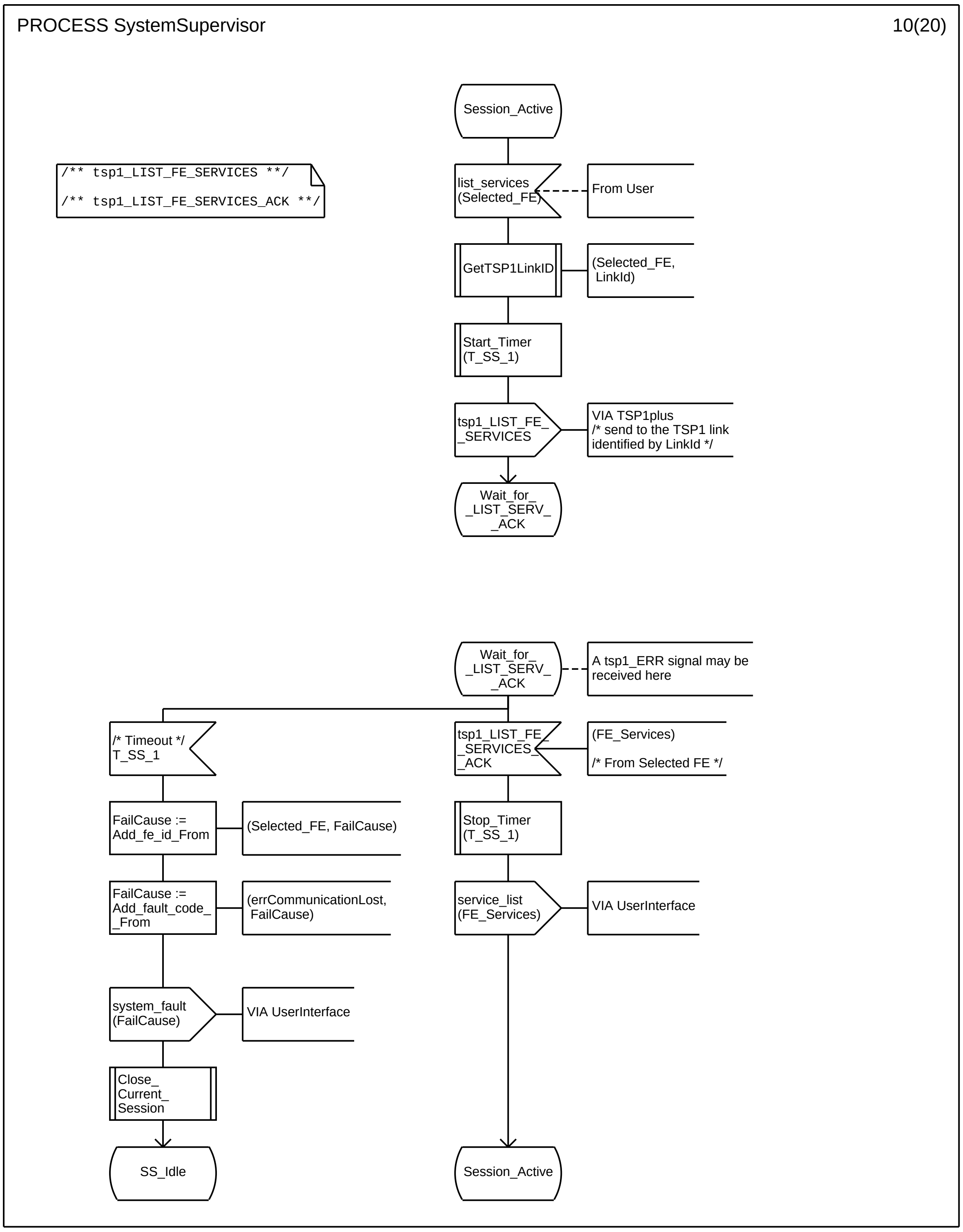

PROCESS SystemSupervisor
5(20)
/** tsp1_INIT_ACK **/
Wait_for_ _Init_Ack
tsp1_INIT_ _ACK
From each connected Front End
*
Save other signals
Wait_for_ _Init_Complete
Only when all FEs have Acknowledged

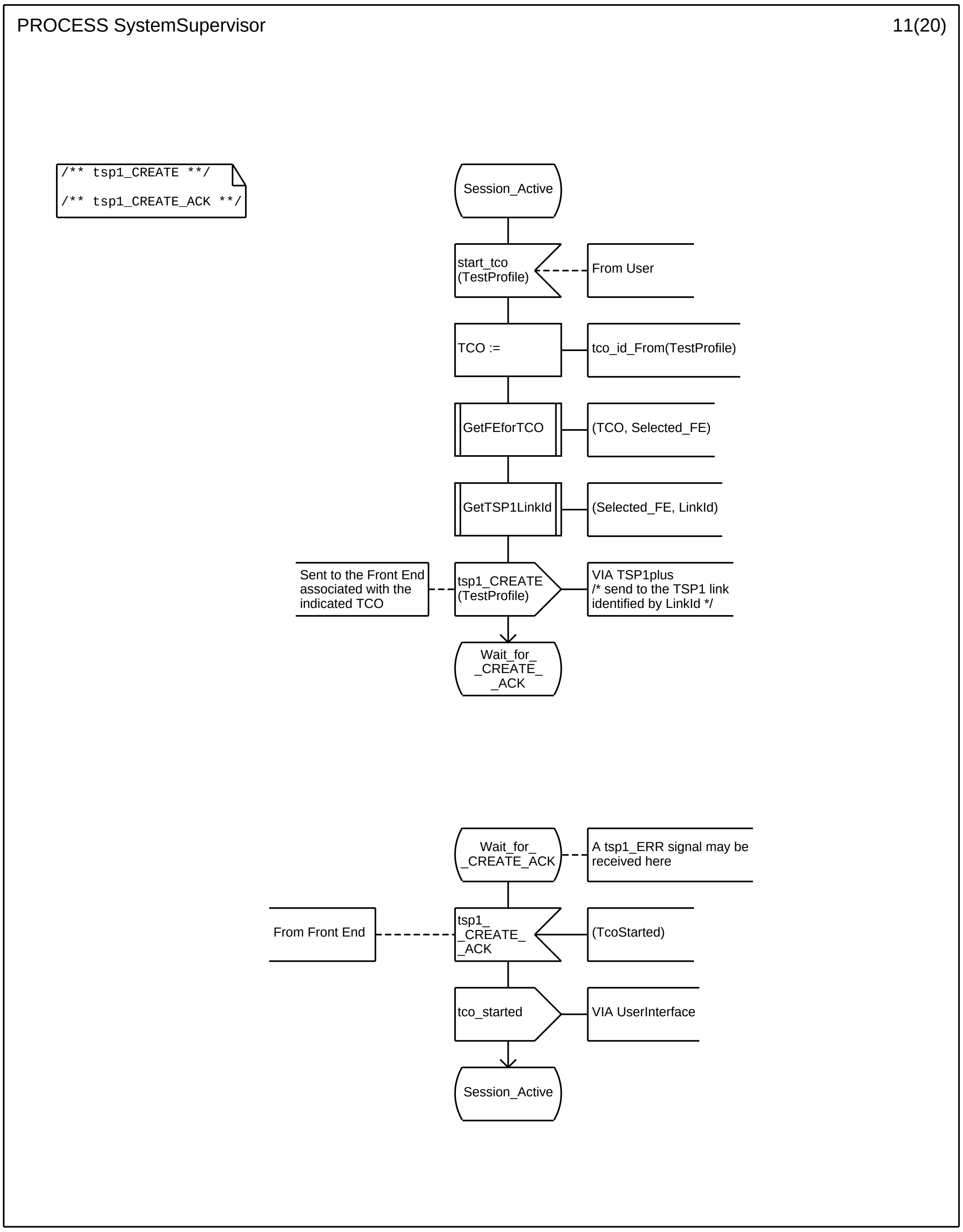

PROCESS SystemSupervisor
6(20)
/** tsp1_INIT_COMPLETE **/
Wait_for_ _Init_Complete
A tsp1_ERR signal may be received here
tsp1_INIT_ _COMPLETE
From each connected Front End
Only when all FEs have Initialized
Session_ _Started
VIA UserInterface
Session_Active

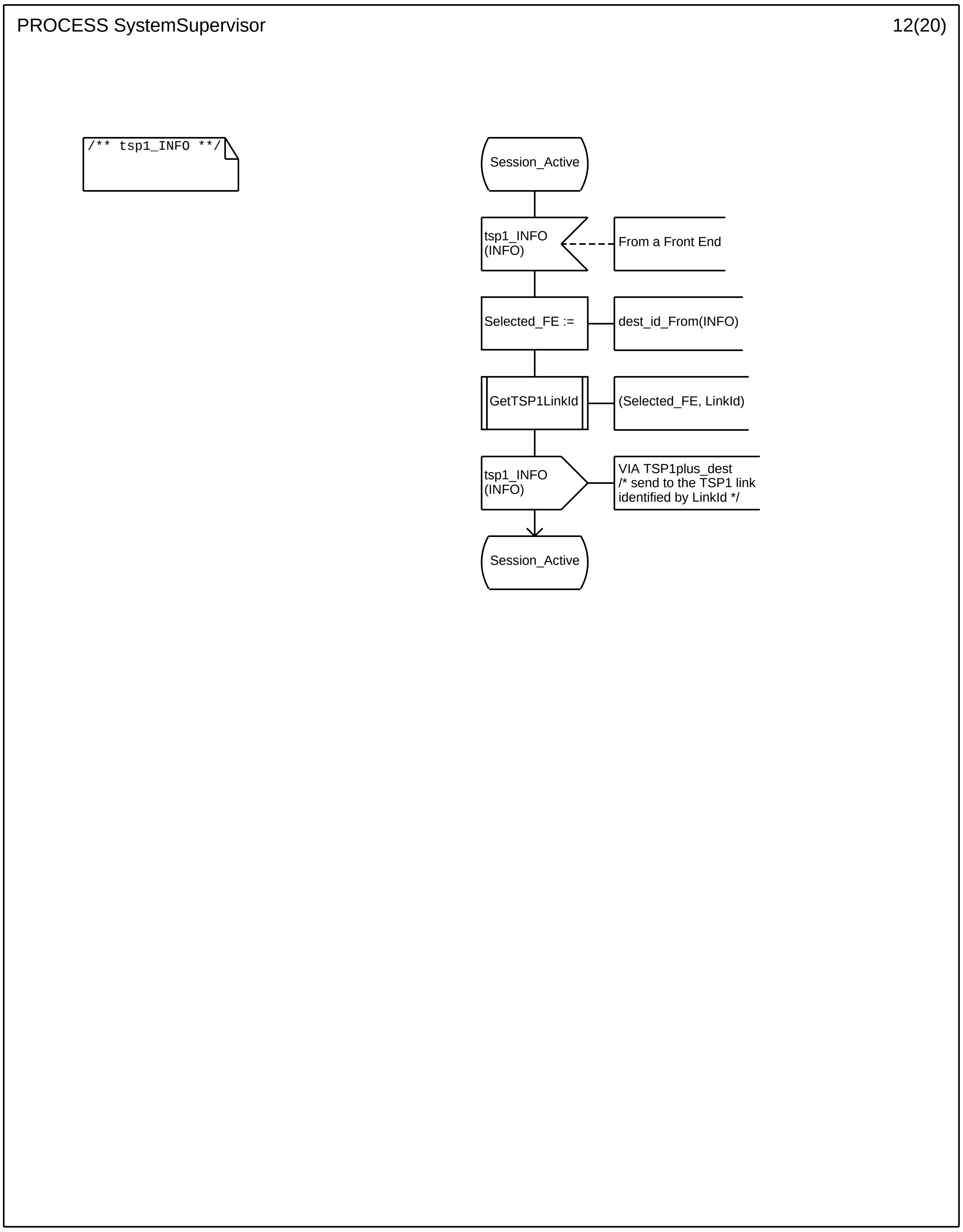

PROCESS SystemSupervisor
7(20)
/** tsp1_CHK_CONF **/
/** tsp1_CHK_CONF_ACK **/
Session_Active
verify_config
From User
Include TSP1 connections for all Front Ends
tsp1_CHK_ _CONF
VIA TSP1plus
Wait_for_ _CHK_CONF_ _ACK
Wait_for_ _CHK_CONF_ _ACK
A tsp1_ERR signal may be received here
tsp1_CHK_ _CONF_ACK
From each connected Front End
verify_config_ _ack
VIA UserInterface
Session_Active

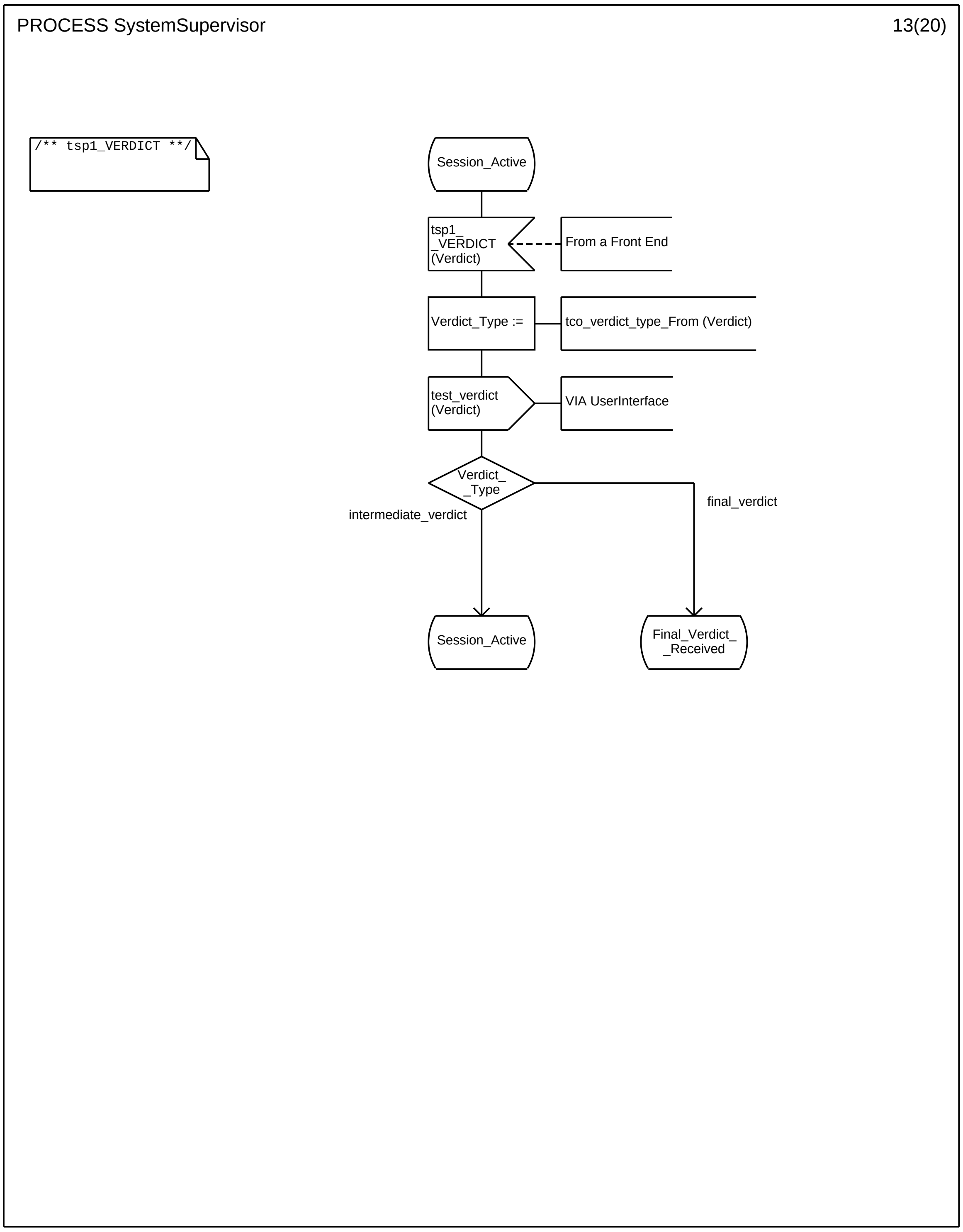

PROCESS SystemSupervisor
8(20)
/** tsp1_SET_PARAMETER **/
/** tsp1_SET_PARAMETER_ACK **/
Session_Active
change_
_parameter
(Param)
From User
Include TSP1 connections for all Front Ends
tsp1_SET_
_PARAMETER
(Param)
VIA TSP1plus
Wait_for_
_SET_PARAM_
_ACK
Wait_for_
_SET_PARAM_
_ACK
A tsp1_ERR signal may be received here
tsp1_SET_
PARAMETER
_ACK
From each connected Front End
Parameter_
_changed
VIA UserInterface
Session_Active

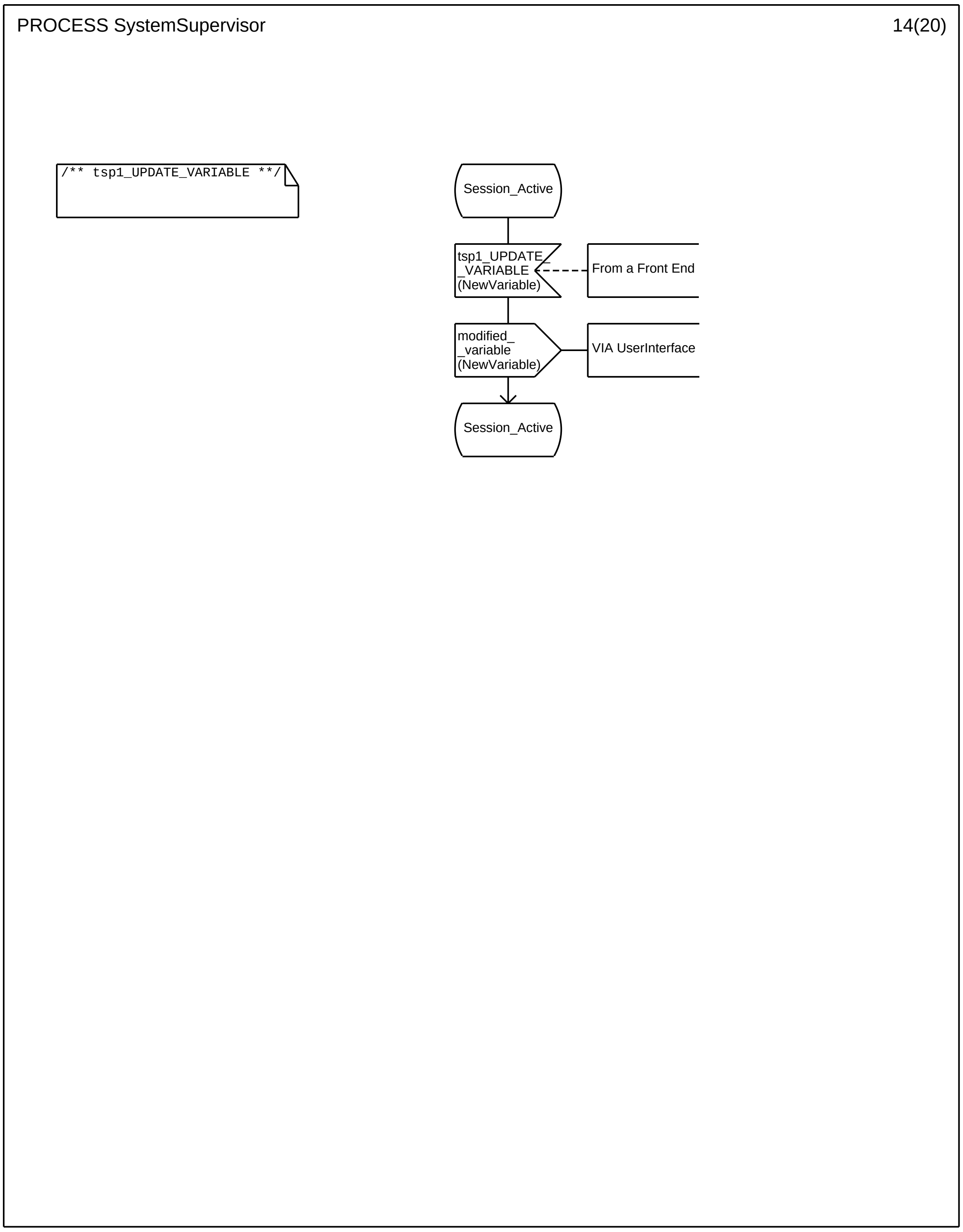

PROCESS SystemSupervisor
9(20)
/** tsp1_SET_TIME **/
/** tsp1_SET_TIME_ACK **/
Session_Active
set_clock (SysTime)
From User
Include TSP1 connections for all Front Ends
tsp1_SET_ _TIME (SysTime)
VIA TSP1plus
Wait_for_ _SET_TIME_ _ACK
Wait_for_ _SET_TIME_ACK
A tsp1_ERR signal may be received here
tsp1_SET_ _TIME_ _ACK
From each connected Front End
clock_set
VIA UserInterface
Session_Active

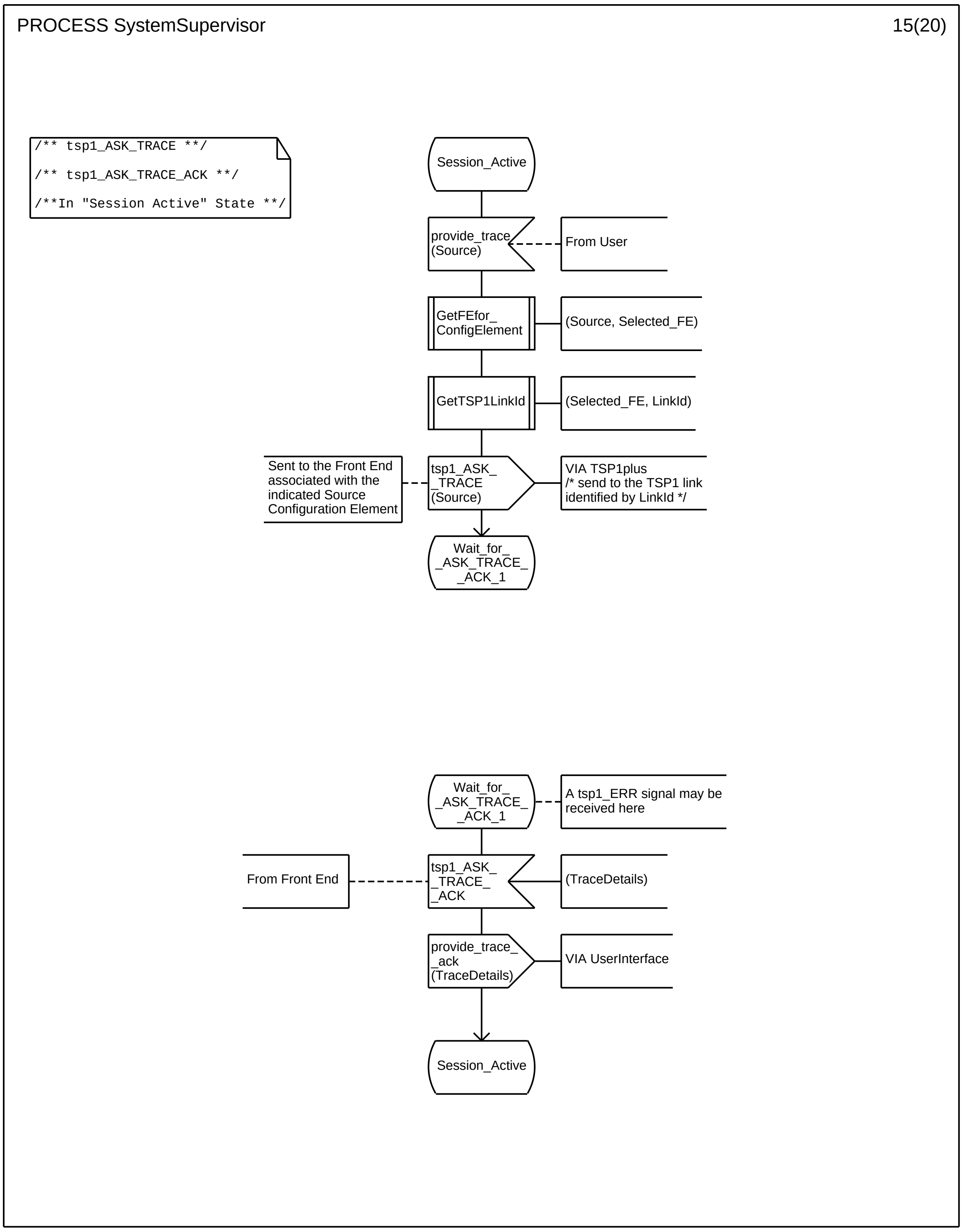

PROCESS SystemSupervisor
10(20)
/** tsp1_LIST_FE_SERVICES **/
/** tsp1_LIST_FE_SERVICES_ACK **/
Session_Active
list_services (Selected_FE)
From User
GetTSP1LinkID
(Selected_FE, LinkId)
Start_Timer (T_SS_1)
tsp1_LIST_FE_ _SERVICES
VIA TSP1plus /* send to the TSP1 link identified by LinkId */
Wait_for_ _LIST_SERV_ _ACK
Wait_for_ _LIST_SERV_ _ACK
A tsp1_ERR signal may be received here
/* Timeout */ T_SS_1
FailCause := Add_fe_id_From
(Selected_FE, FailCause)
FailCause := Add_fault_code_ _From
(errCommunicationLost, FailCause)
system_fault (FailCause)
VIA UserInterface
Close_ Current_ Session
SS_Idle
tsp1_LIST_FE_ _SERVICES_ _ACK
(FE_Services)
/* From Selected FE */
Stop_Timer (T_SS_1)
service_list (FE_Services)
VIA UserInterface
Session_Active

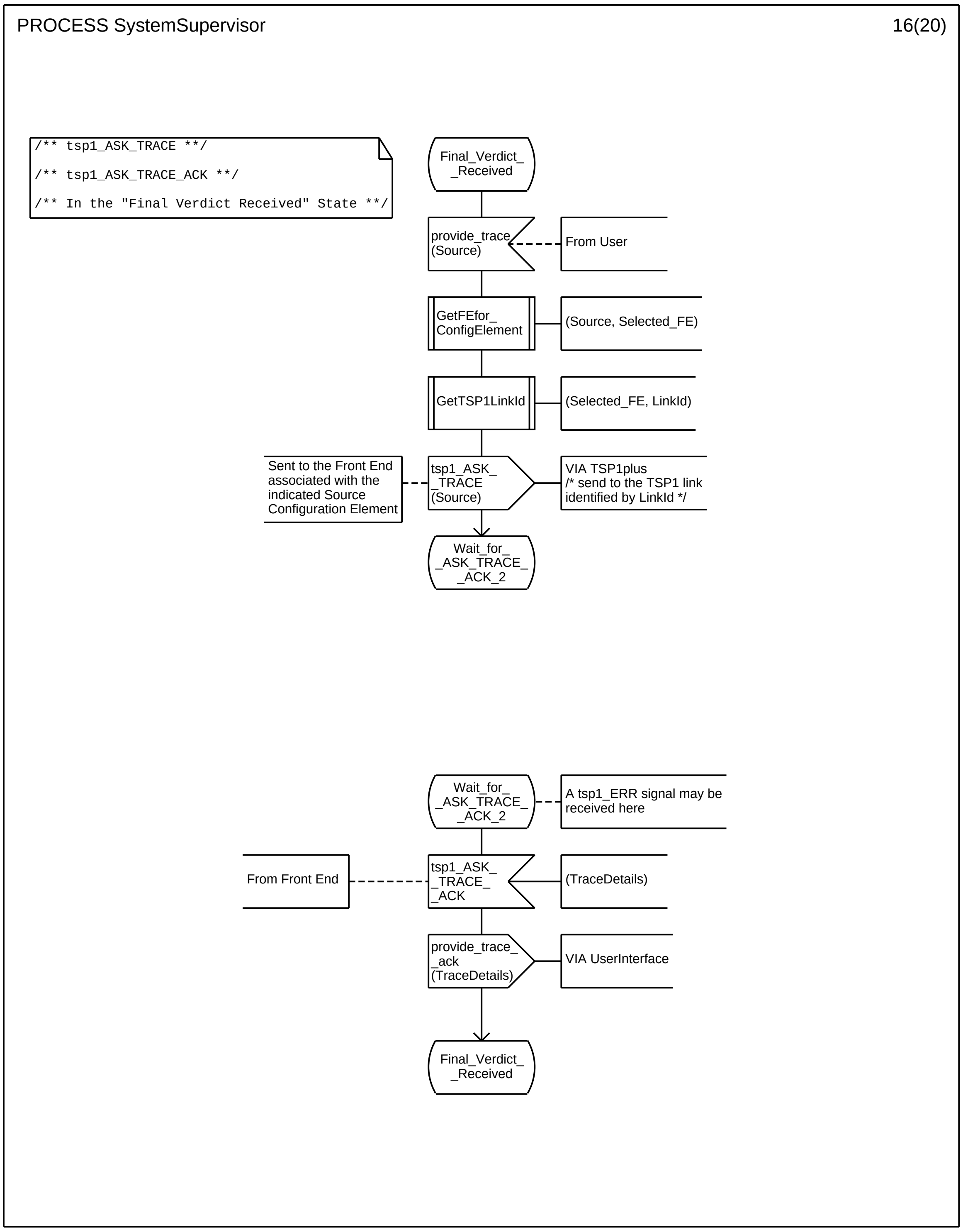

PROCESS SystemSupervisor
11(20)
/** tsp1_CREATE **/
/** tsp1_CREATE_ACK **/
Session_Active
start_tco (TestProfile)
From User
TCO :=
tco_id_From(TestProfile)
GetFEforTCO
(TCO, Selected_FE)
GetTSP1LinkId
(Selected_FE, LinkId)
Sent to the Front End associated with the indicated TCO
tsp1_CREATE (TestProfile)
VIA TSP1plus /* send to the TSP1 link identified by LinkId */
Wait_for_ _CREATE_ _ACK
Wait_for_ _CREATE_ACK
A tsp1_ERR signal may be received here
From Front End
tsp1_ _CREATE_ _ACK
(TcoStarted)
tco_started
VIA UserInterface
Session_Active

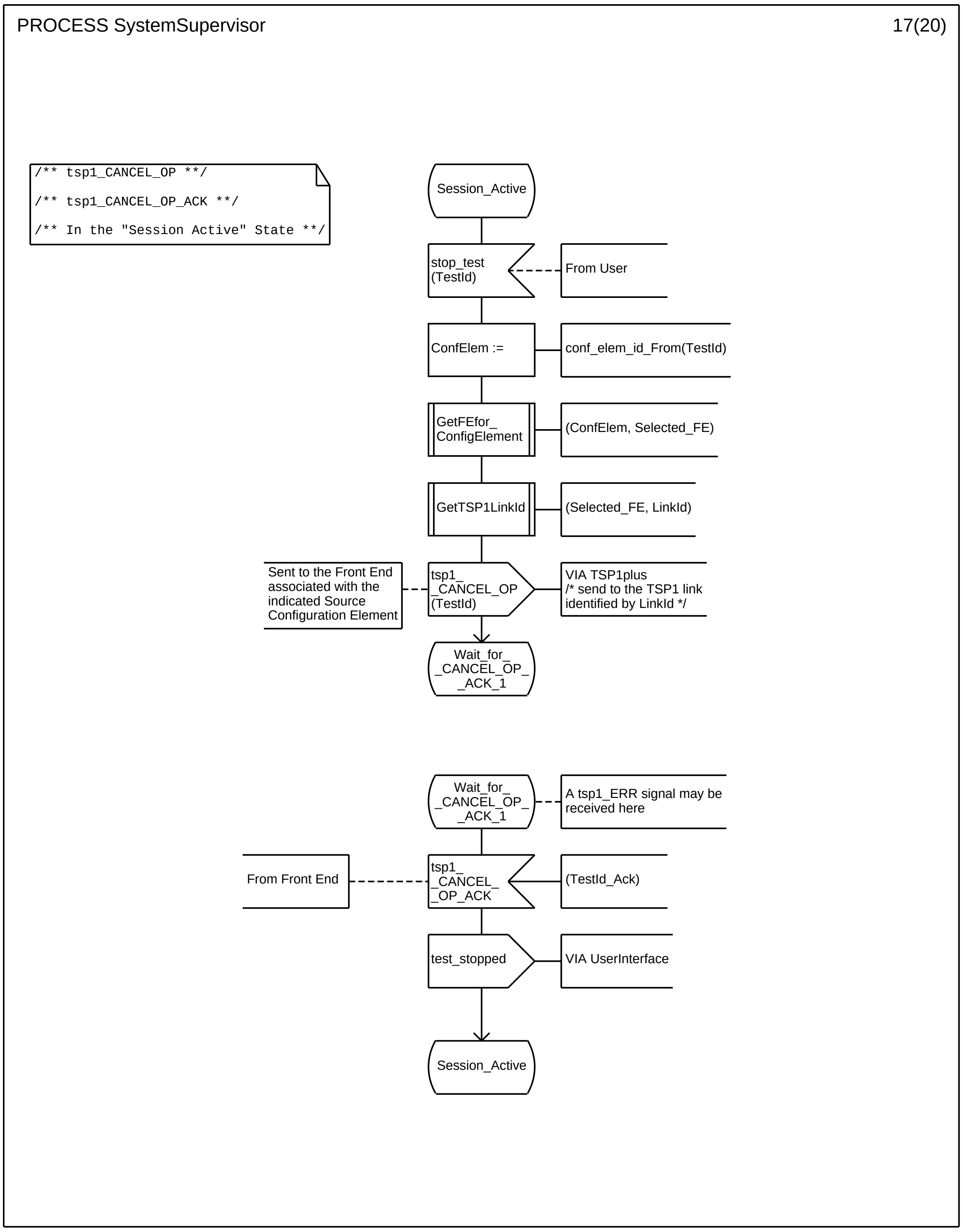

PROCESS SystemSupervisor
12(20)
/** tsp1_INFO **/
Session_Active
tsp1_INFO (INFO)
From a Front End
Selected_FE :=
dest_id_From(INFO)
GetTSP1LinkId
(Selected_FE, LinkId)
tsp1_INFO (INFO)
VIA TSP1plus_dest /* send to the TSP1 link identified by LinkId */
Session_Active

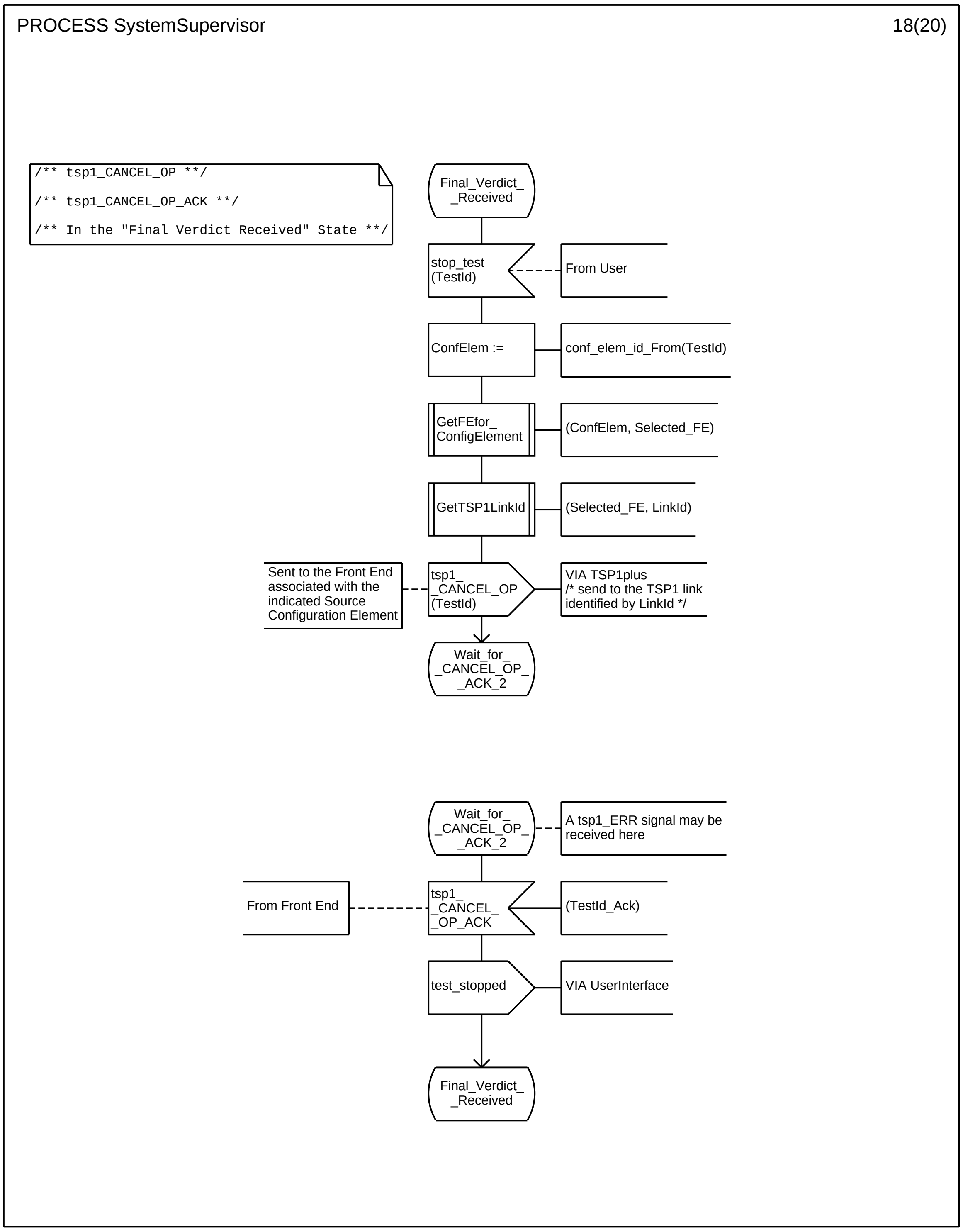

PROCESS SystemSupervisor
13(20)
/** tsp1_VERDICT **/
Session_Active
tsp1_
_VERDICT
(Verdict)
From a Front End
Verdict_Type :=
tco_verdict_type_From (Verdict)
test_verdict
(Verdict)
VIA UserInterface
Verdict_
_Type
intermediate_verdict
final_verdict
Session_Active
Final_Verdict_
_Received

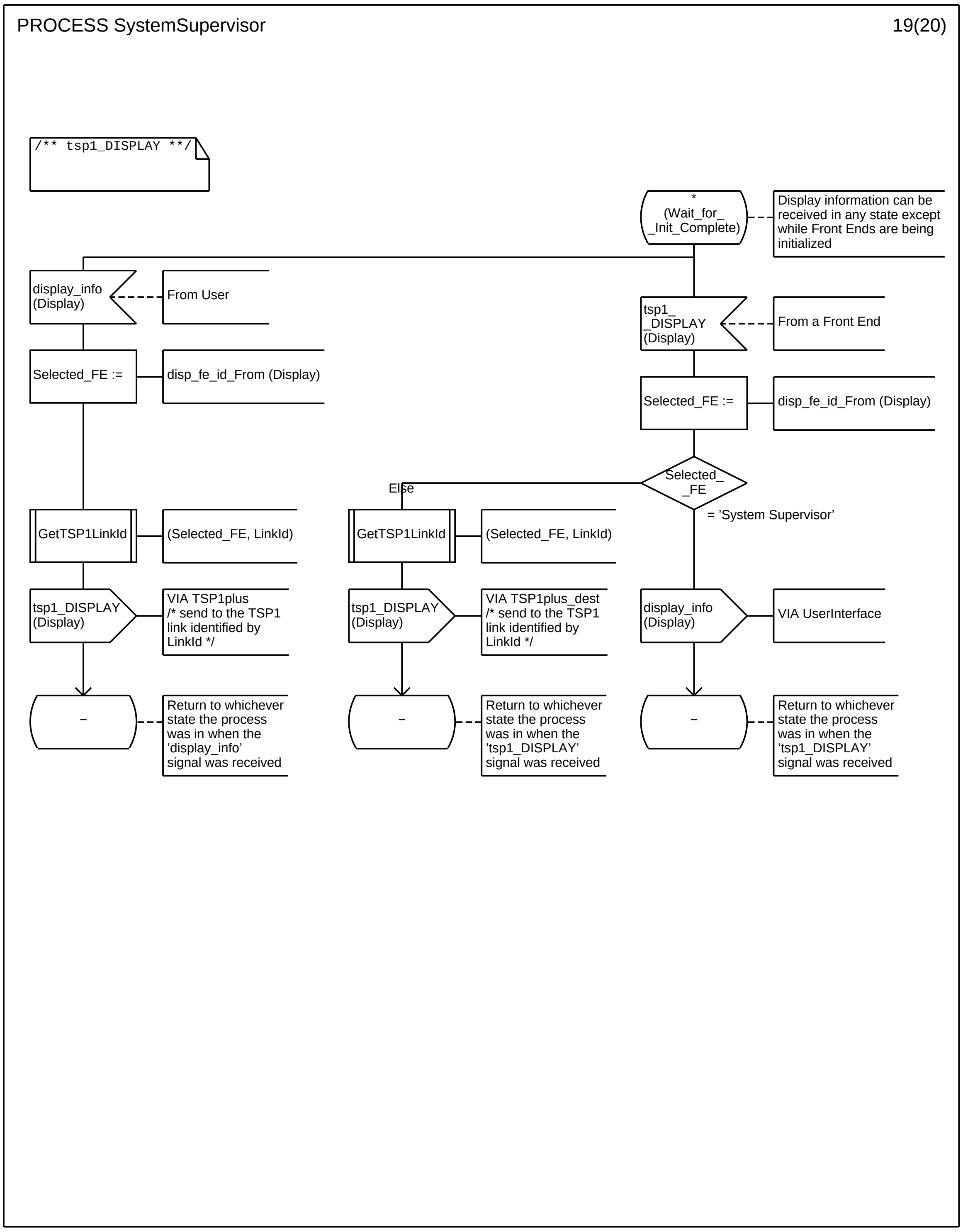

PROCESS SystemSupervisor
14(20)
/** tsp1_UPDATE_VARIABLE **/
Session_Active
tsp1_UPDATE_
_VARIABLE
(NewVariable)
From a Front End
modified_
_variable
(NewVariable)
VIA UserInterface
Session_Active

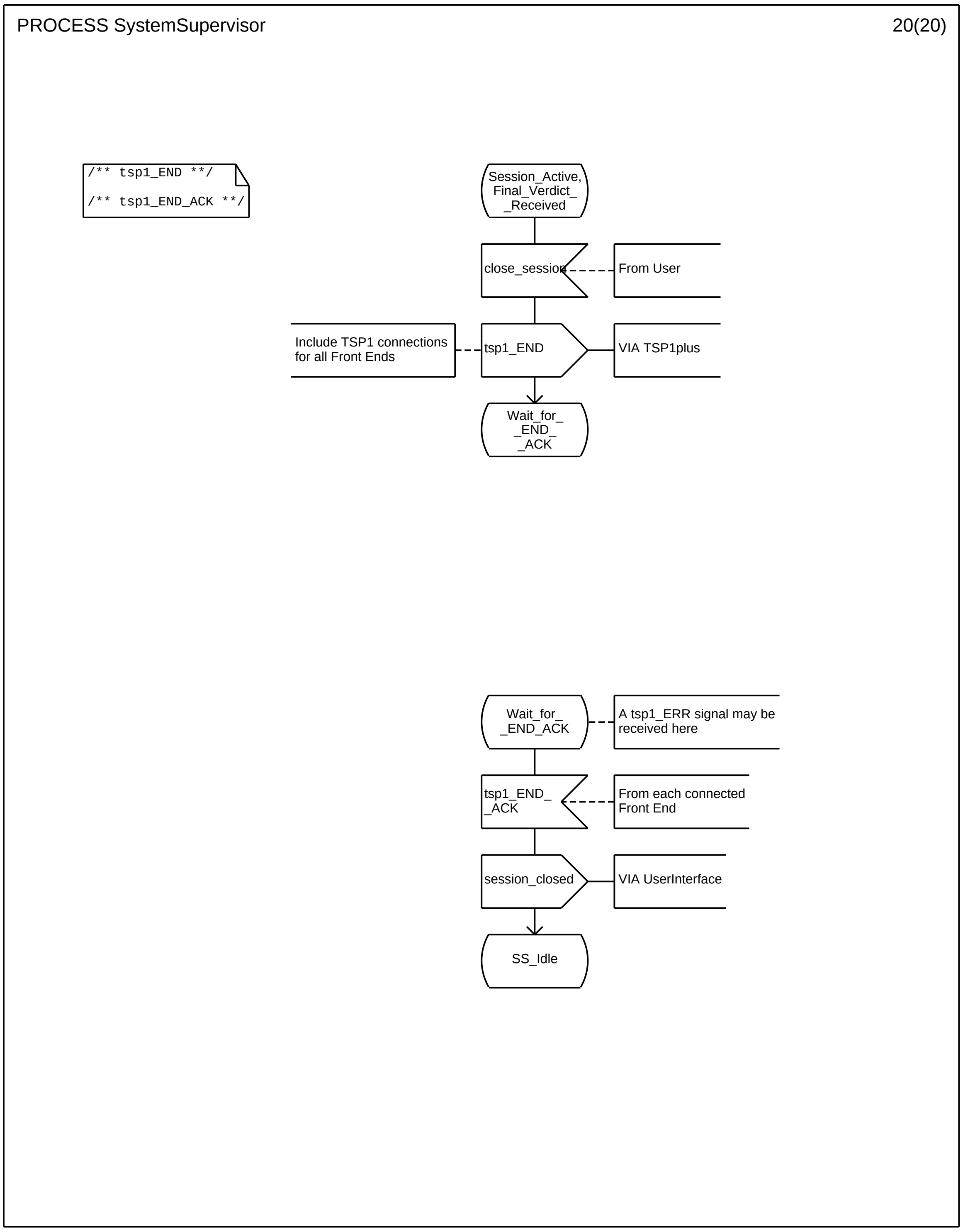

PROCESS SystemSupervisor
15(20)
/** tsp1_ASK_TRACE **/
/** tsp1_ASK_TRACE_ACK **/
/**In "Session Active" State **/
Session_Active
provide_trace (Source)
From User
GetFEfor_ ConfigElement
(Source, Selected_FE)
GetTSP1LinkId
(Selected_FE, LinkId)
Sent to the Front End associated with the indicated Source Configuration Element
tsp1_ASK_ _TRACE (Source)
VIA TSP1plus /* send to the TSP1 link identified by LinkId */
Wait_for_ _ASK_TRACE_ _ACK_1
Wait_for_ _ASK_TRACE_ _ACK_1
A tsp1_ERR signal may be received here
From Front End
tsp1_ASK_ _TRACE_ _ACK
(TraceDetails)
provide_trace_ _ack (TraceDetails)
VIA UserInterface
Session_Active

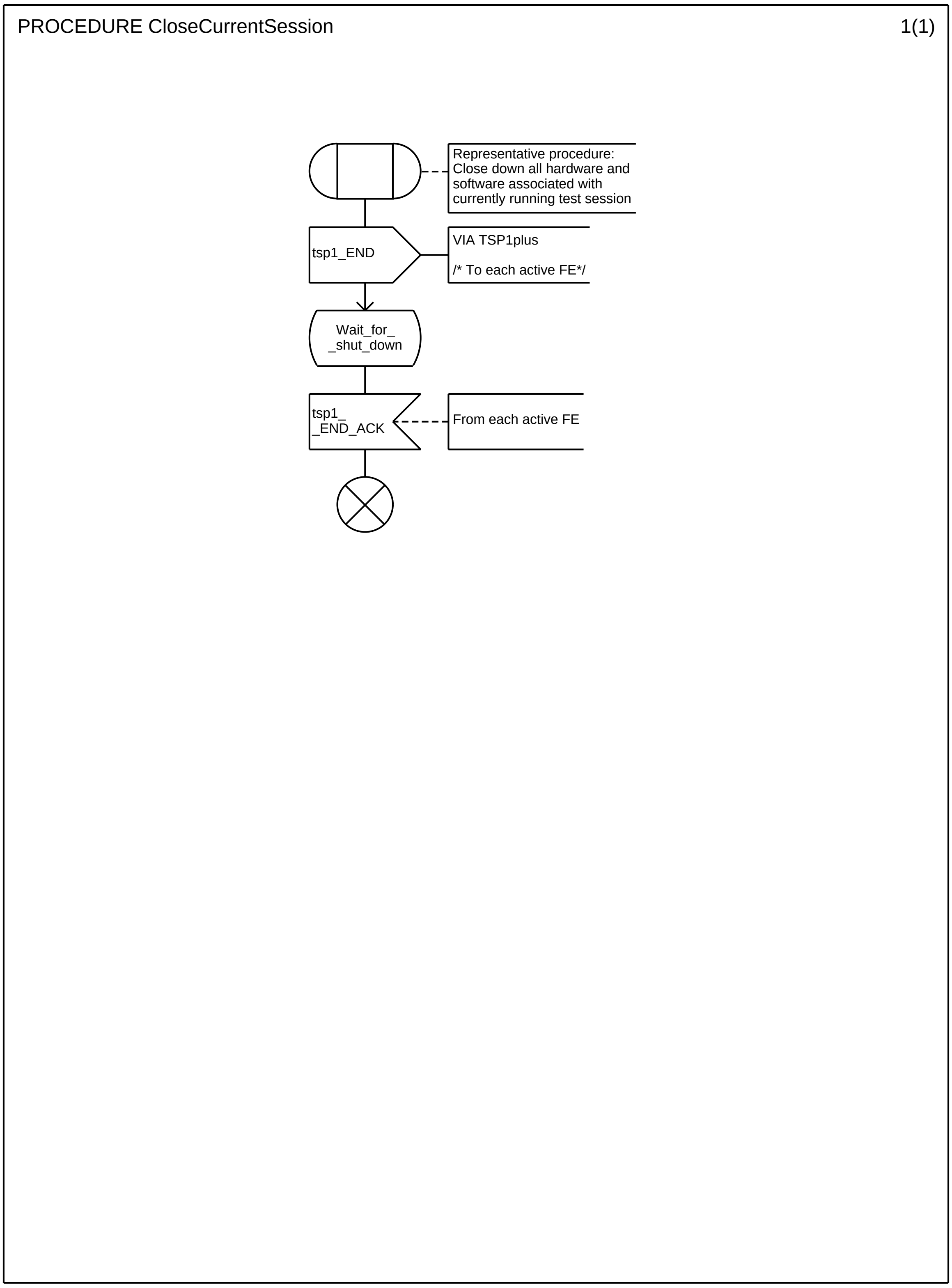

PROCESS SystemSupervisor
16(20)
/** tsp1_ASK_TRACE **/
/** tsp1_ASK_TRACE_ACK **/
/** In the "Final Verdict Received" State **/
Final_Verdict_ _Received
provide_trace (Source)
From User
GetFEfor_ ConfigElement
(Source, Selected_FE)
GetTSP1LinkId
(Selected_FE, LinkId)
Sent to the Front End associated with the indicated Source Configuration Element
tsp1_ASK_ _TRACE (Source)
VIA TSP1plus /* send to the TSP1 link identified by LinkId */
Wait_for_ _ASK_TRACE_ _ACK_2
Wait_for_ _ASK_TRACE_ _ACK_2
A tsp1_ERR signal may be received here
From Front End
tsp1_ASK_ _TRACE_ _ACK
(TraceDetails)
provide_trace_ _ack (TraceDetails)
VIA UserInterface
Final_Verdict_ _Received

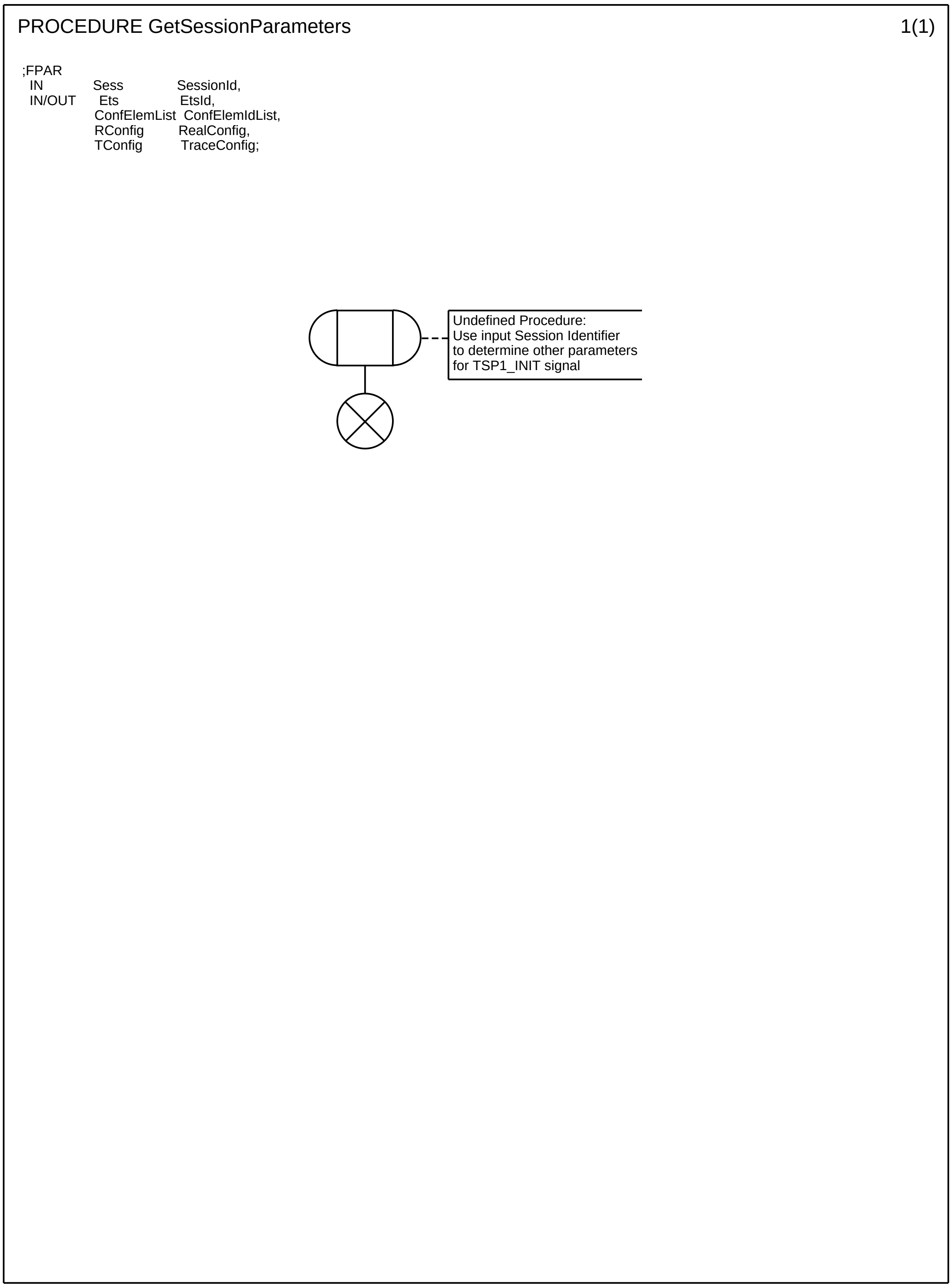

PROCESS SystemSupervisor
17(20)
/** tsp1_CANCEL_OP **/
/** tsp1_CANCEL_OP_ACK **/
/** In the "Session Active" State **/
Session_Active
stop_test (TestId)
From User
ConfElem :=
conf_elem_id_From(TestId)
GetFEfor_ ConfigElement
(ConfElem, Selected_FE)
GetTSP1LinkId
(Selected_FE, LinkId)
Sent to the Front End associated with the indicated Source Configuration Element
tsp1_ _CANCEL_OP (TestId)
VIA TSP1plus /* send to the TSP1 link identified by LinkId */
Wait_for_ _CANCEL_OP_ _ACK_1
Wait_for_ _CANCEL_OP_ _ACK_1
A tsp1_ERR signal may be received here
From Front End
tsp1_ _CANCEL_ _OP_ACK
(TestId_Ack)
test_stopped
VIA UserInterface
Session_Active

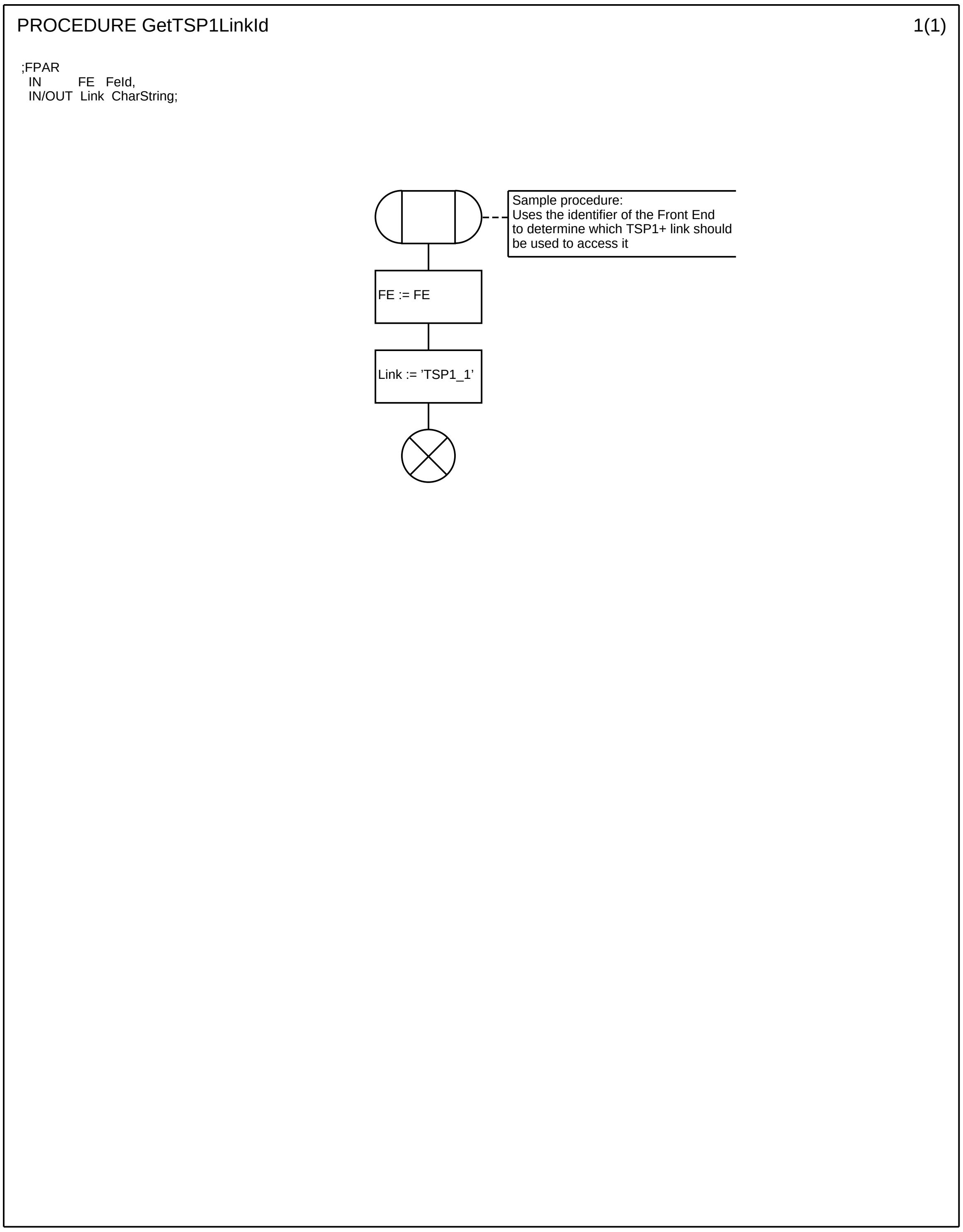

PROCESS SystemSupervisor
18(20)
/** tsp1_CANCEL_OP **/
/** tsp1_CANCEL_OP_ACK **/
/** In the "Final Verdict Received" State **/
Final_Verdict_ _Received
stop_test (TestId)
From User
ConfElem :=
conf_elem_id_From(TestId)
GetFEfor_ ConfigElement
(ConfElem, Selected_FE)
GetTSP1LinkId
(Selected_FE, LinkId)
Sent to the Front End associated with the indicated Source Configuration Element
tsp1_ _CANCEL_OP (TestId)
VIA TSP1plus /* send to the TSP1 link identified by LinkId */
Wait_for_ _CANCEL_OP_ _ACK_2
Wait_for_ _CANCEL_OP_ _ACK_2
A tsp1_ERR signal may be received here
From Front End
tsp1_ _CANCEL_ _OP_ACK
(TestId_Ack)
test_stopped
VIA UserInterface
Final_Verdict_ _Received

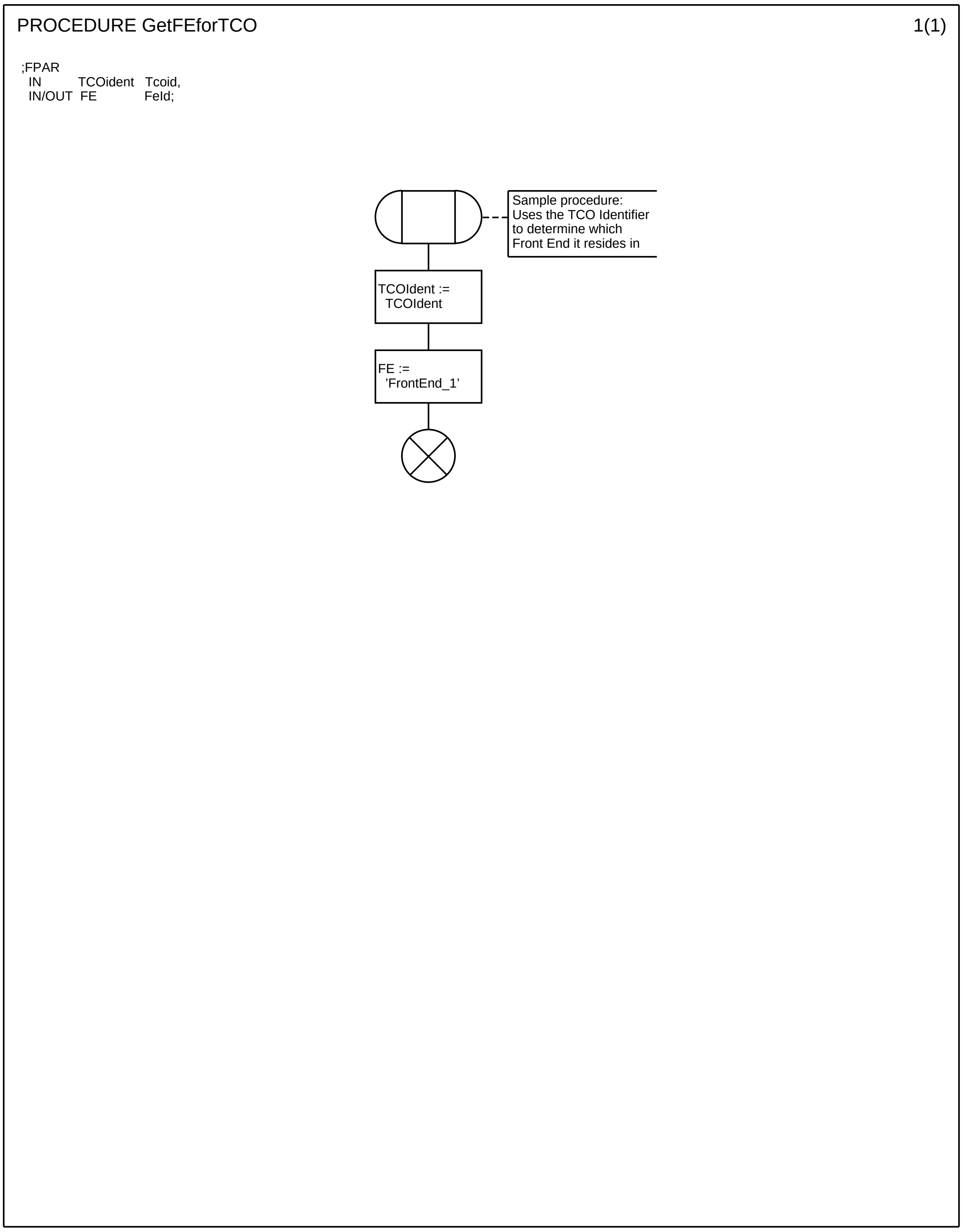

PROCESS SystemSupervisor
19(20)
/** tsp1_DISPLAY **/
*
(Wait_for_
_Init_Complete)
Display information can be received in any state except while Front Ends are being initialized
display_info
(Display)
From User
tsp1_
_DISPLAY
(Display)
From a Front End
Selected_FE :=
disp_fe_id_From (Display)
Selected_FE :=
disp_fe_id_From (Display)
Selected_
_FE
Else
= 'System Supervisor'
GetTSP1LinkId
(Selected_FE, LinkId)
GetTSP1LinkId
(Selected_FE, LinkId)
tsp1_DISPLAY
(Display)
VIA TSP1plus
/* send to the TSP1 link identified by LinkId */
tsp1_DISPLAY
(Display)
VIA TSP1plus_dest
/* send to the TSP1 link identified by LinkId */
display_info
(Display)
VIA UserInterface
–
Return to whichever state the process was in when the 'display_info' signal was received
–
Return to whichever state the process was in when the 'tsp1_DISPLAY' signal was received
–
Return to whichever state the process was in when the 'tsp1_DISPLAY' signal was received

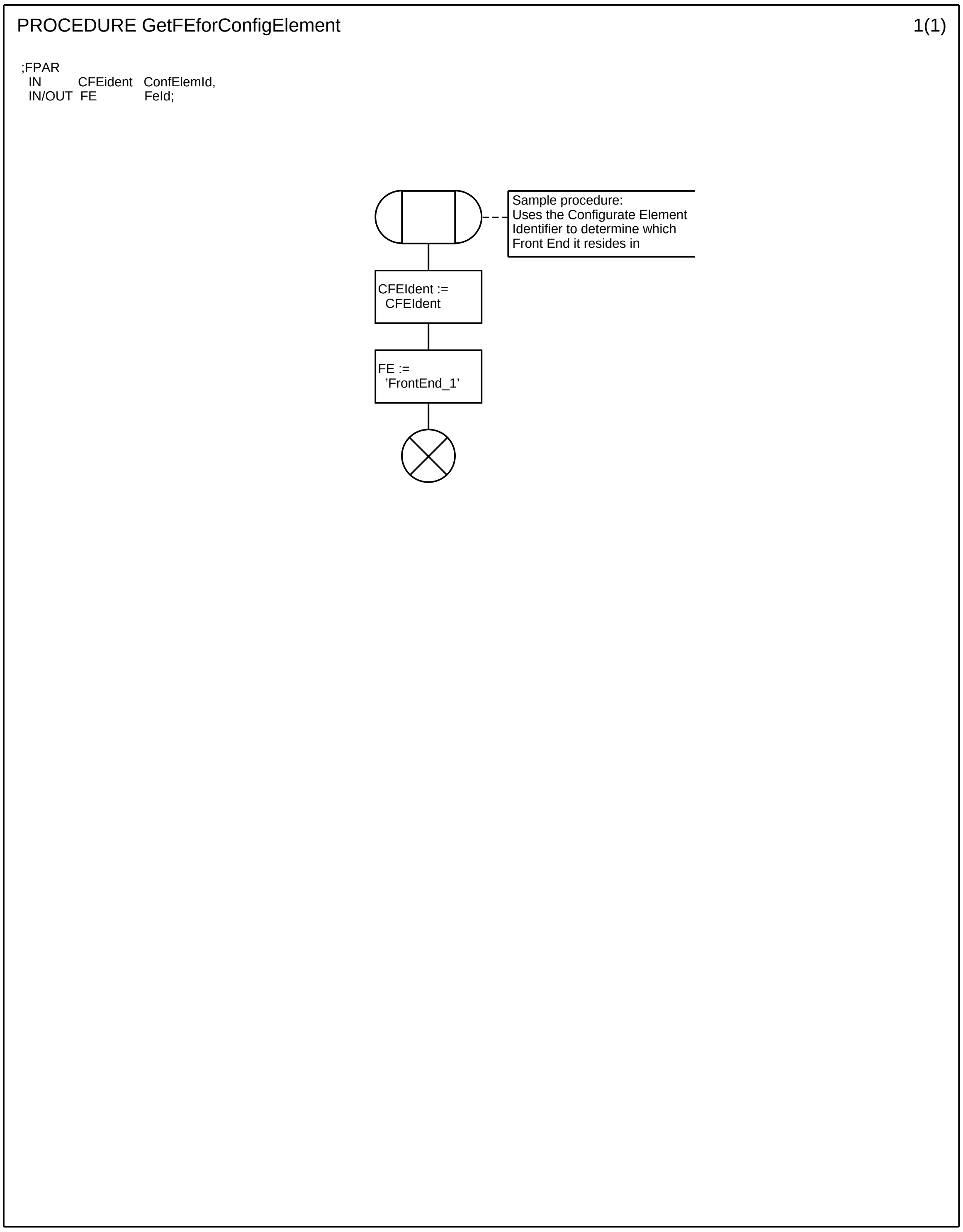

PROCESS SystemSupervisor
20(20)
/** tsp1_END **/
/** tsp1_END_ACK **/
Session_Active, Final_Verdict_ _Received
close_session
From User
Include TSP1 connections for all Front Ends
tsp1_END
VIA TSP1plus
Wait_for_ _END_ _ACK
Wait_for_ _END_ACK
A tsp1_ERR signal may be received here
tsp1_END_ _ACK
From each connected Front End
session_closed
VIA UserInterface
SS_Idle

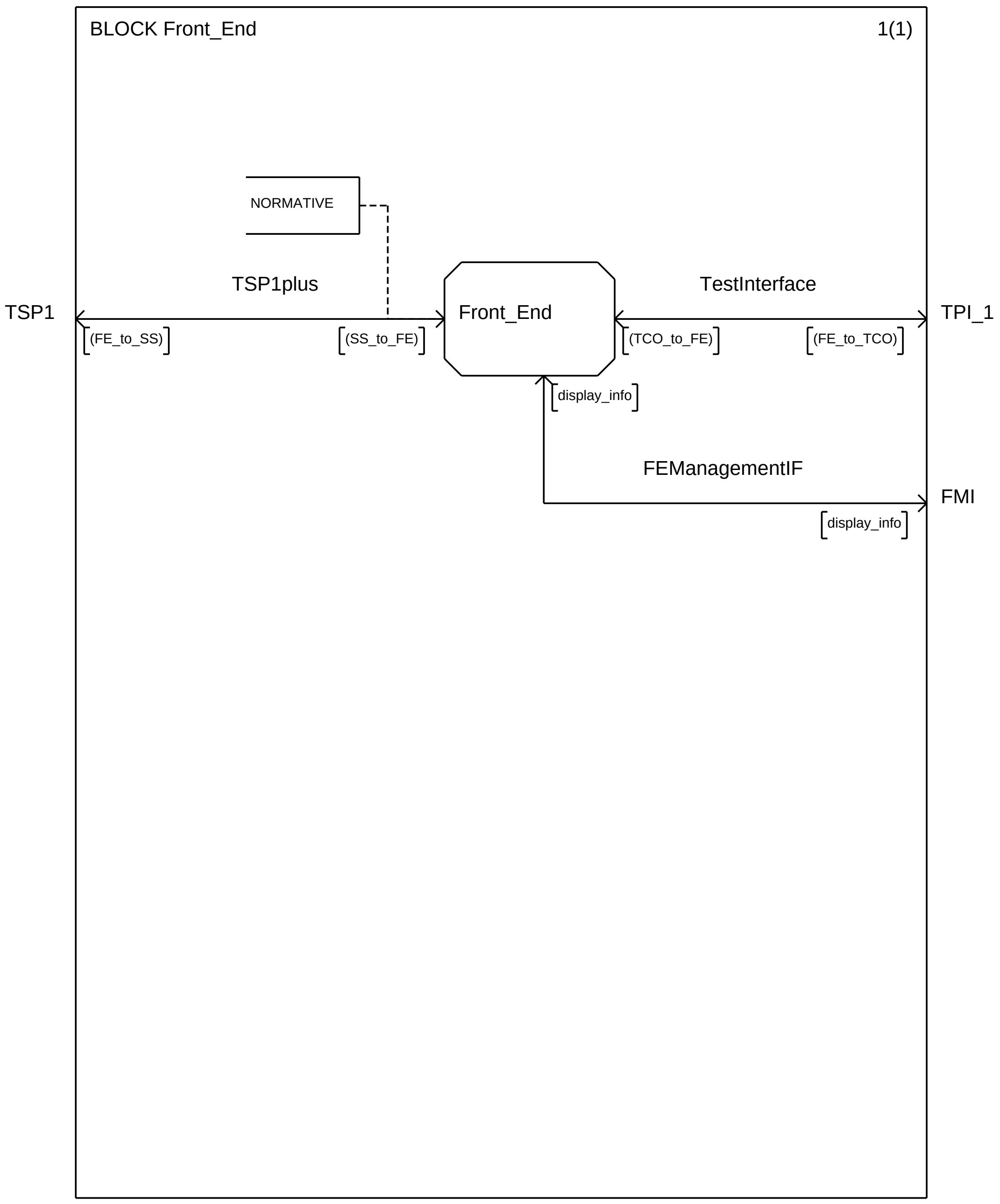

PROCEDURE CloseCurrentSession
1(1)
Representative procedure: Close down all hardware and software associated with currently running test session
tsp1_END
VIA TSP1plus
/* To each active FE*/
Wait_for_ _shut_down
tsp1_ _END_ACK
From each active FE

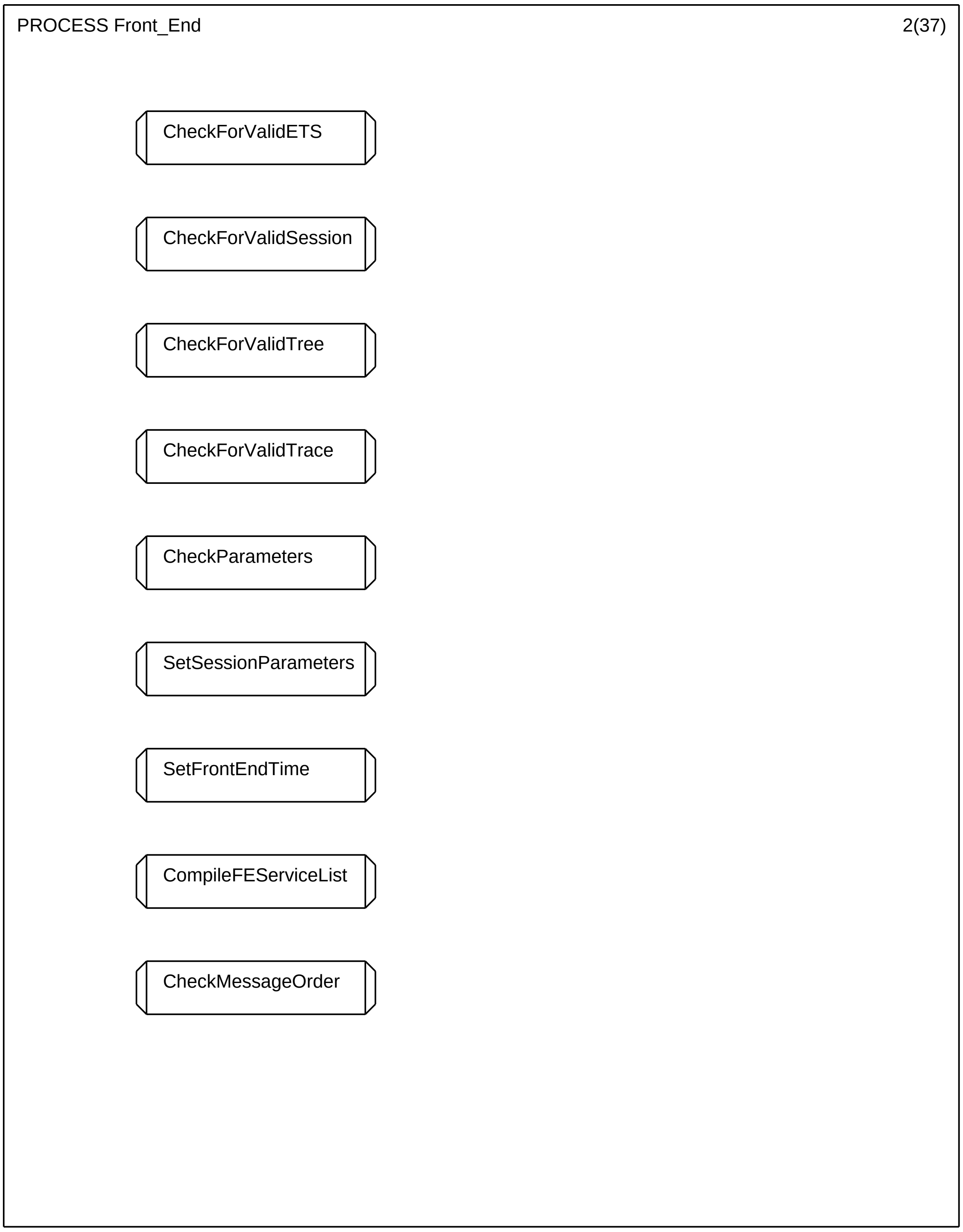

PROCEDURE GetSessionParameters
1(1)
;FPAR
IN Sess SessionId,
IN/OUT Ets EtsId,
ConfElemList ConfElemIdList,
RConfig RealConfig,
TConfig TraceConfig;
Undefined Procedure:
Use input Session Identifier
to determine other parameters
for TSP1_INIT signal

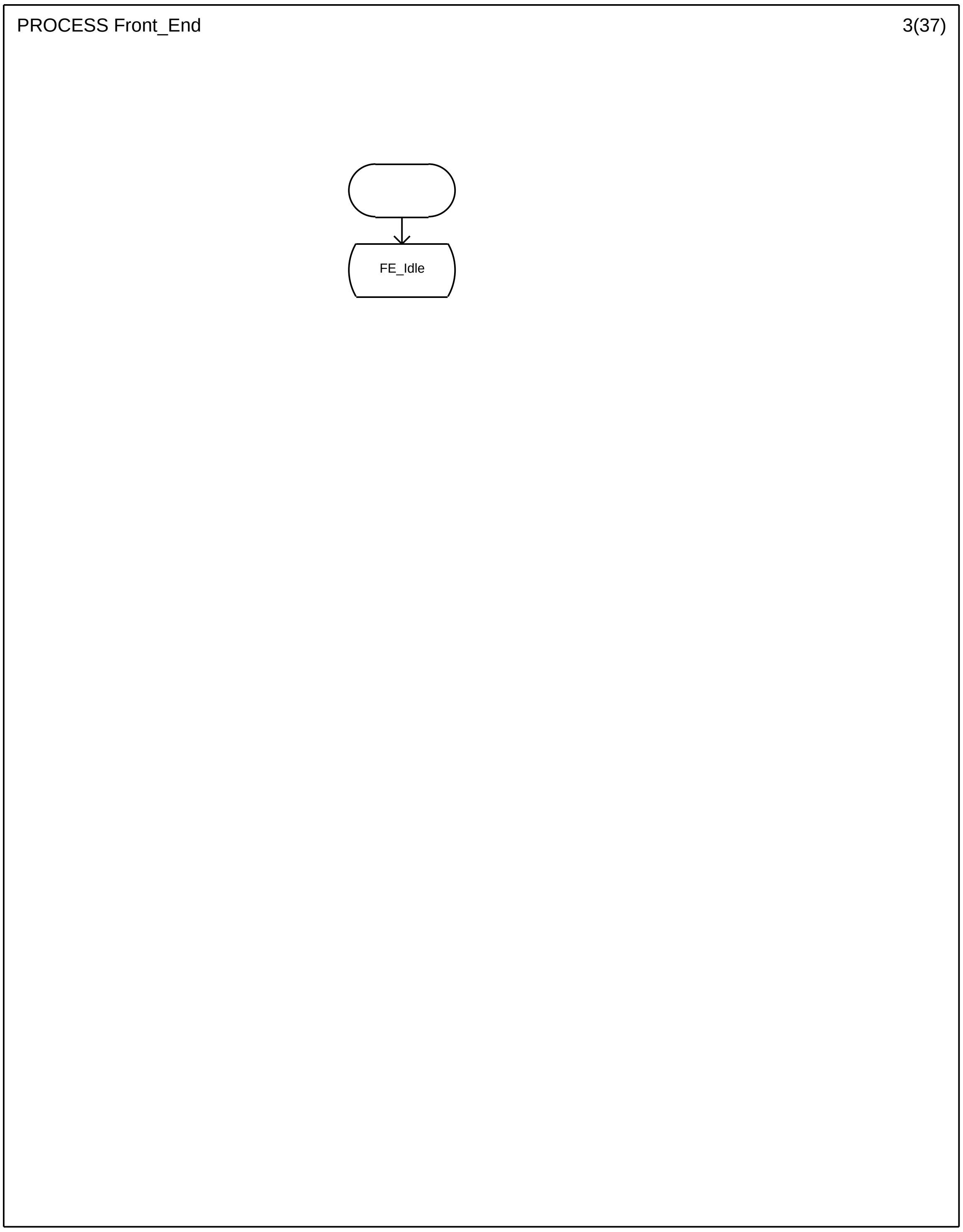

PROCEDURE GetTSP1LinkId
1(1)
;FPAR
IN FE FeId,
IN/OUT Link CharString;
Sample procedure:
Uses the identifier of the Front End
to determine which TSP1+ link should
be used to access it
FE := FE
Link := 'TSP1_1'

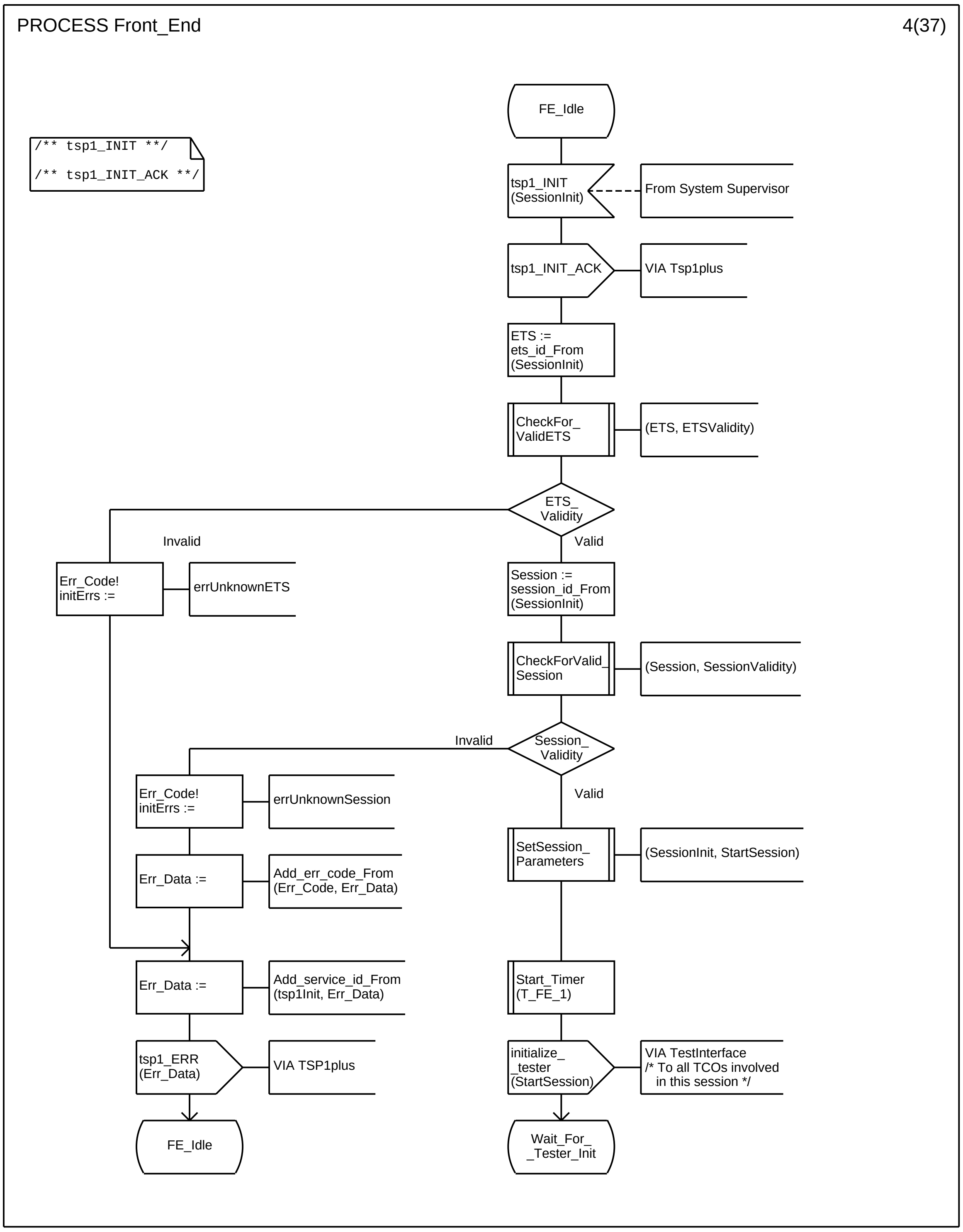

PROCEDURE GetFEforTCO
1(1)
;FPAR
IN TCOident Tcoid,
IN/OUT FE FeId;
Sample procedure:
Uses the TCO Identifier
to determine which
Front End it resides in
TCOIdent :=
TCOIdent
FE :=
'FrontEnd_1'

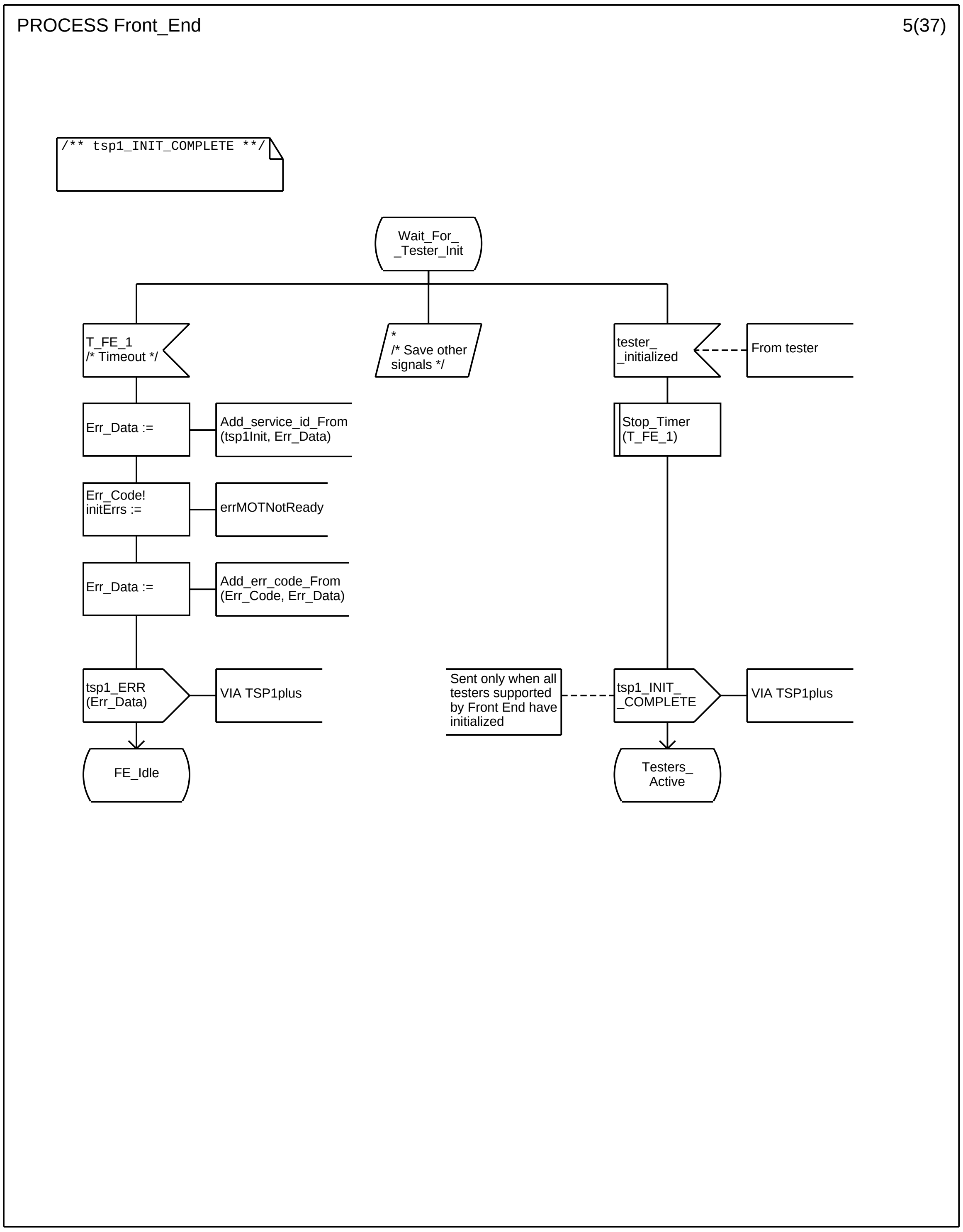

PROCEDURE GetFEforConfigElement
1(1)
;FPAR
IN CFEident ConfElemId,
IN/OUT FE FeId;
Sample procedure: Uses the Configurate Element Identifier to determine which Front End it resides in
CFEIdent := CFEIdent
FE := 'FrontEnd_1'

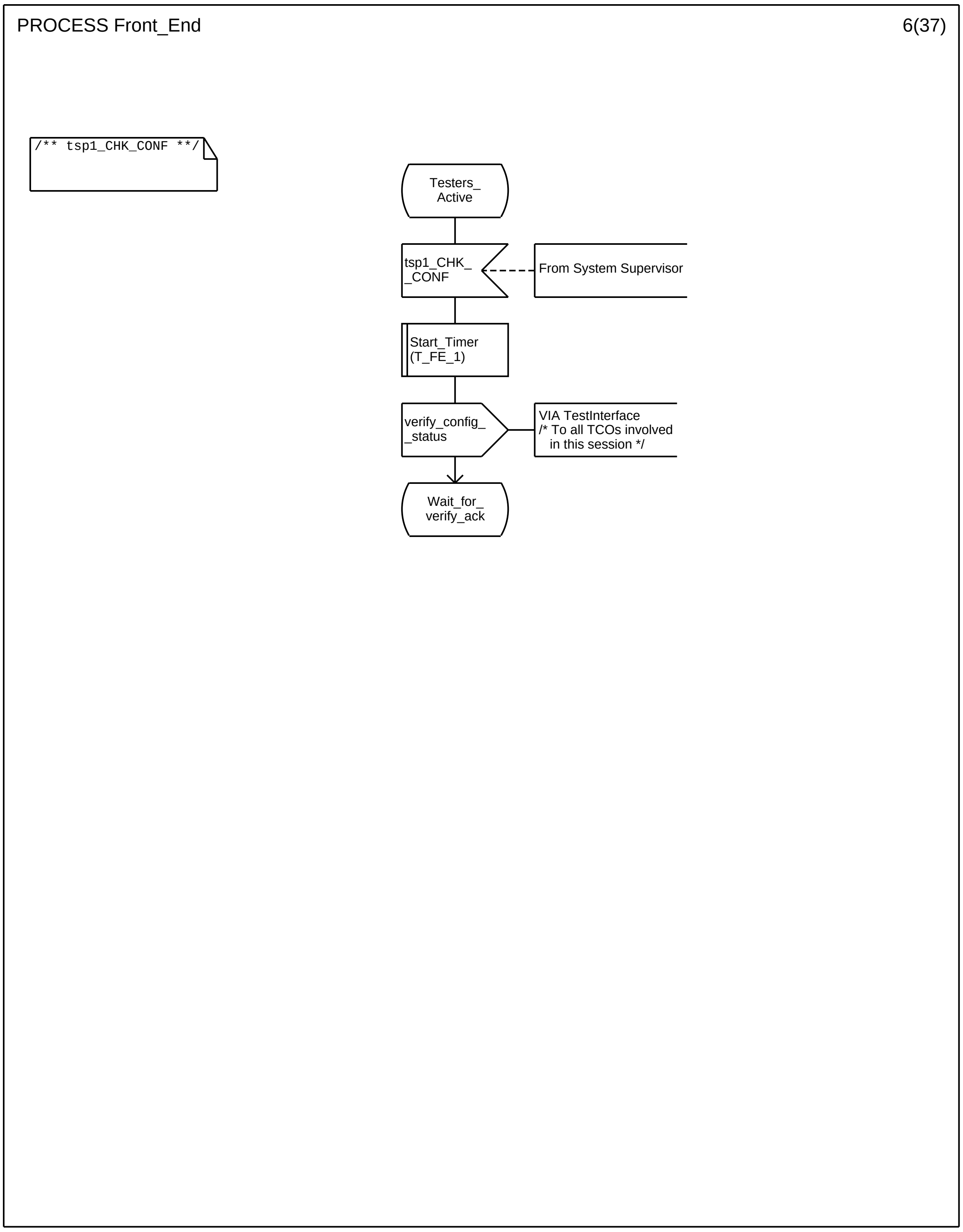

BLOCK Front_End
1(1)
NORMATIVE
TSP1plus
TestInterface
TSP1
Front_End
TPI_1
(FE_to_SS)
(SS_to_FE)
(TCO_to_FE)
(FE_to_TCO)
display_info
FEManagementIF
FMI
display_info

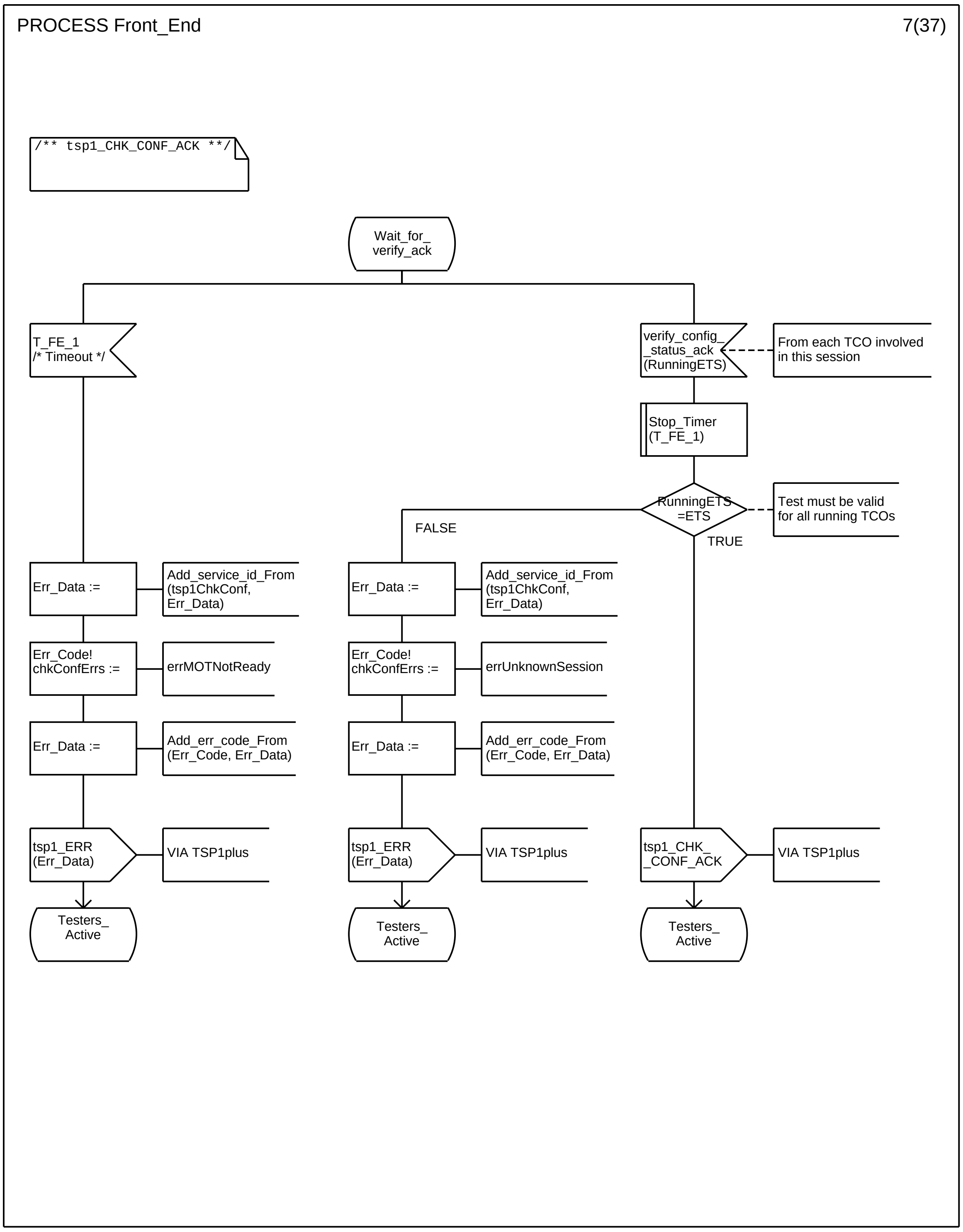

PROCESS Front_End
2(37)
CheckForValidETS
CheckForValidSession
CheckForValidTree
CheckForValidTrace
CheckParameters
SetSessionParameters
SetFrontEndTime
CompileFEServiceList
CheckMessageOrder

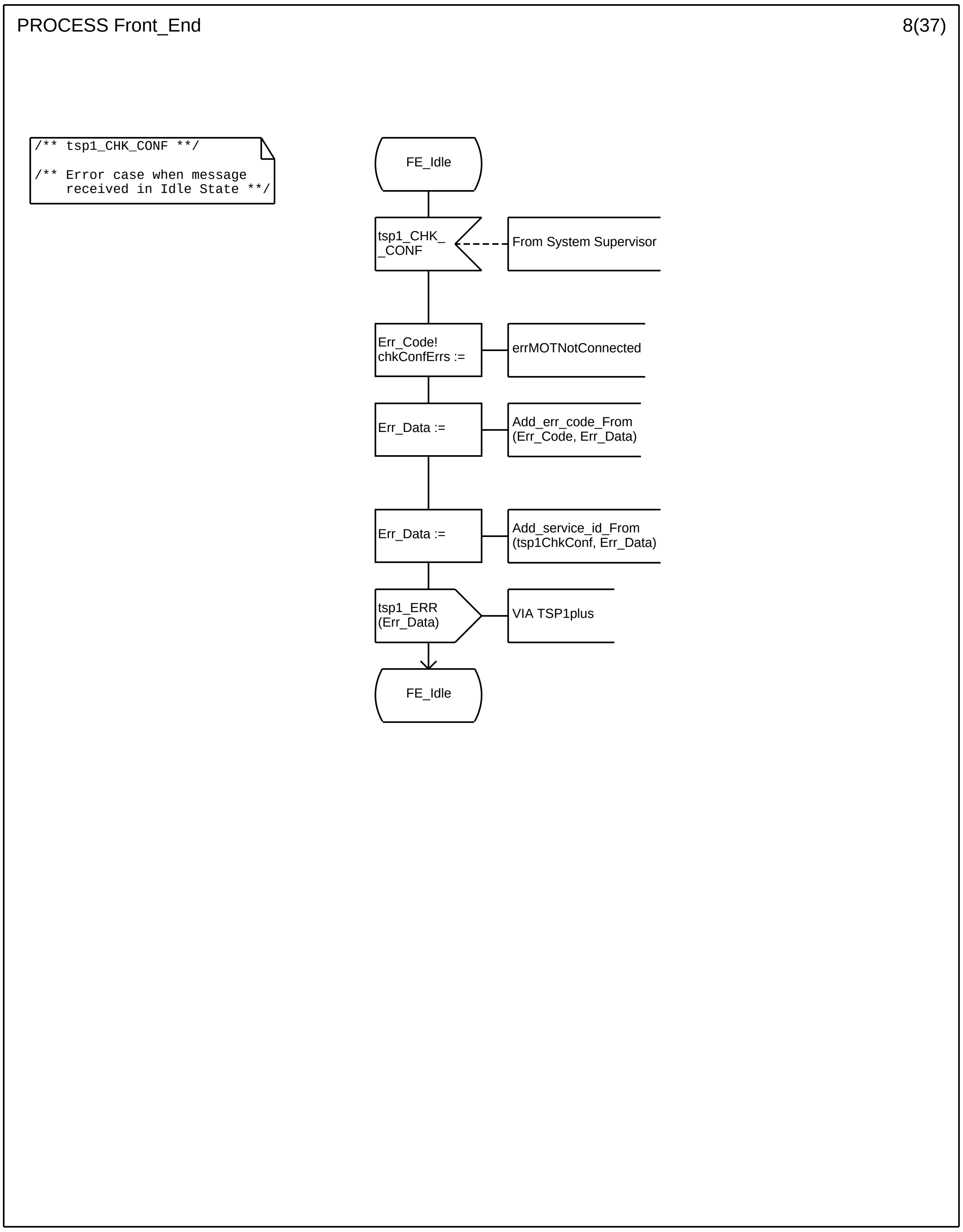

PROCESS Front_End
3(37)
FE_Idle

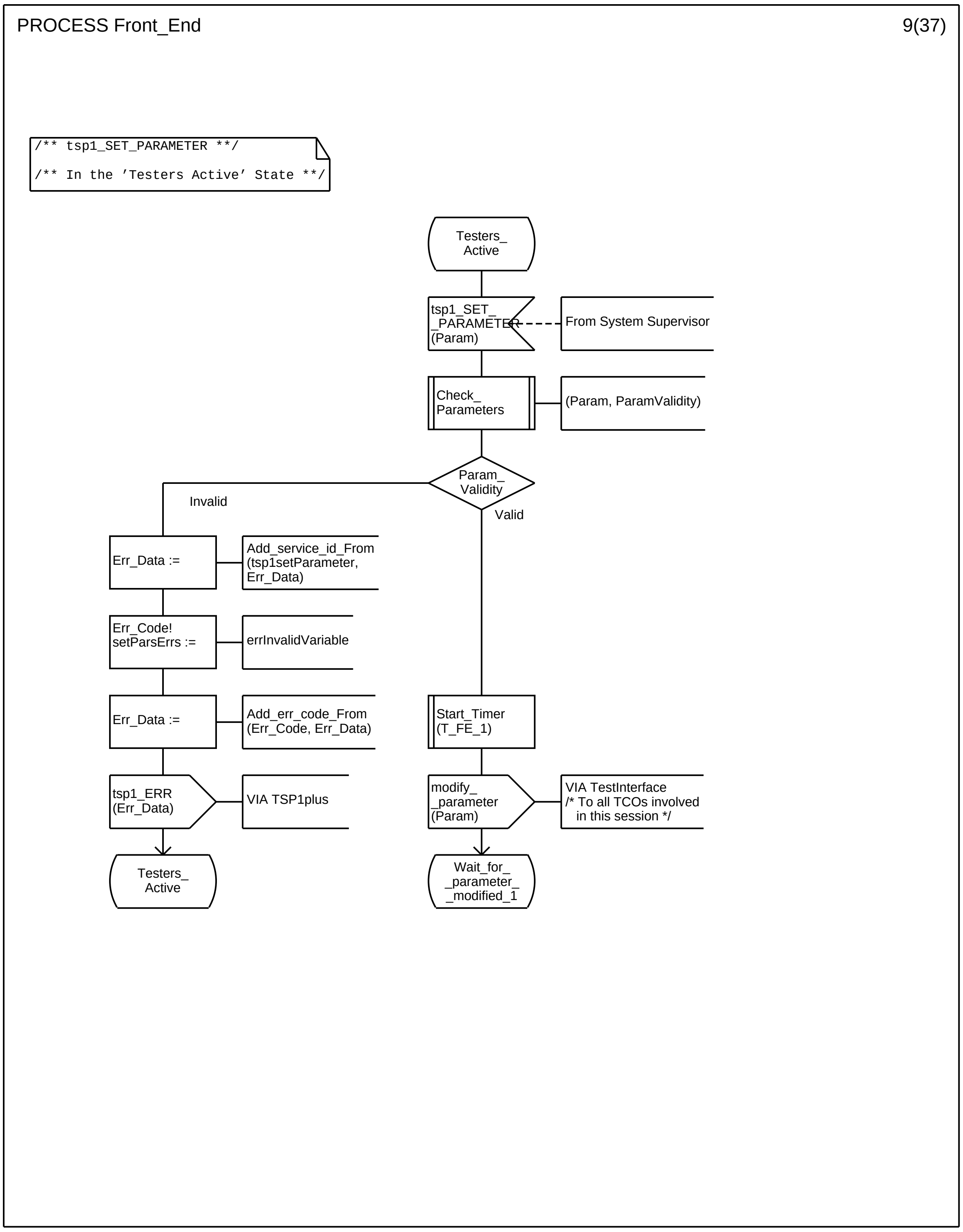

PROCESS Front_End
4(37)
/** tsp1_INIT **/
/** tsp1_INIT_ACK **/
FE_Idle
tsp1_INIT (SessionInit)
From System Supervisor
tsp1_INIT_ACK
VIA Tsp1plus
ETS := ets_id_From (SessionInit)
CheckFor_ ValidETS
(ETS, ETSValidity)
ETS_ Validity
Invalid
Valid
Err_Code! initErrs :=
errUnknownETS
Session := session_id_From (SessionInit)
CheckForValid_ Session
(Session, SessionValidity)
Invalid
Session_ Validity
Valid
Err_Code! initErrs :=
errUnknownSession
SetSession_ Parameters
(SessionInit, StartSession)
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
Err_Data :=
Add_service_id_From (tsp1Init, Err_Data)
Start_Timer (T_FE_1)
tsp1_ERR (Err_Data)
VIA TSP1plus
initialize_ _tester (StartSession)
VIA TestInterface /* To all TCOs involved in this session */
FE_Idle
Wait_For_ _Tester_Init

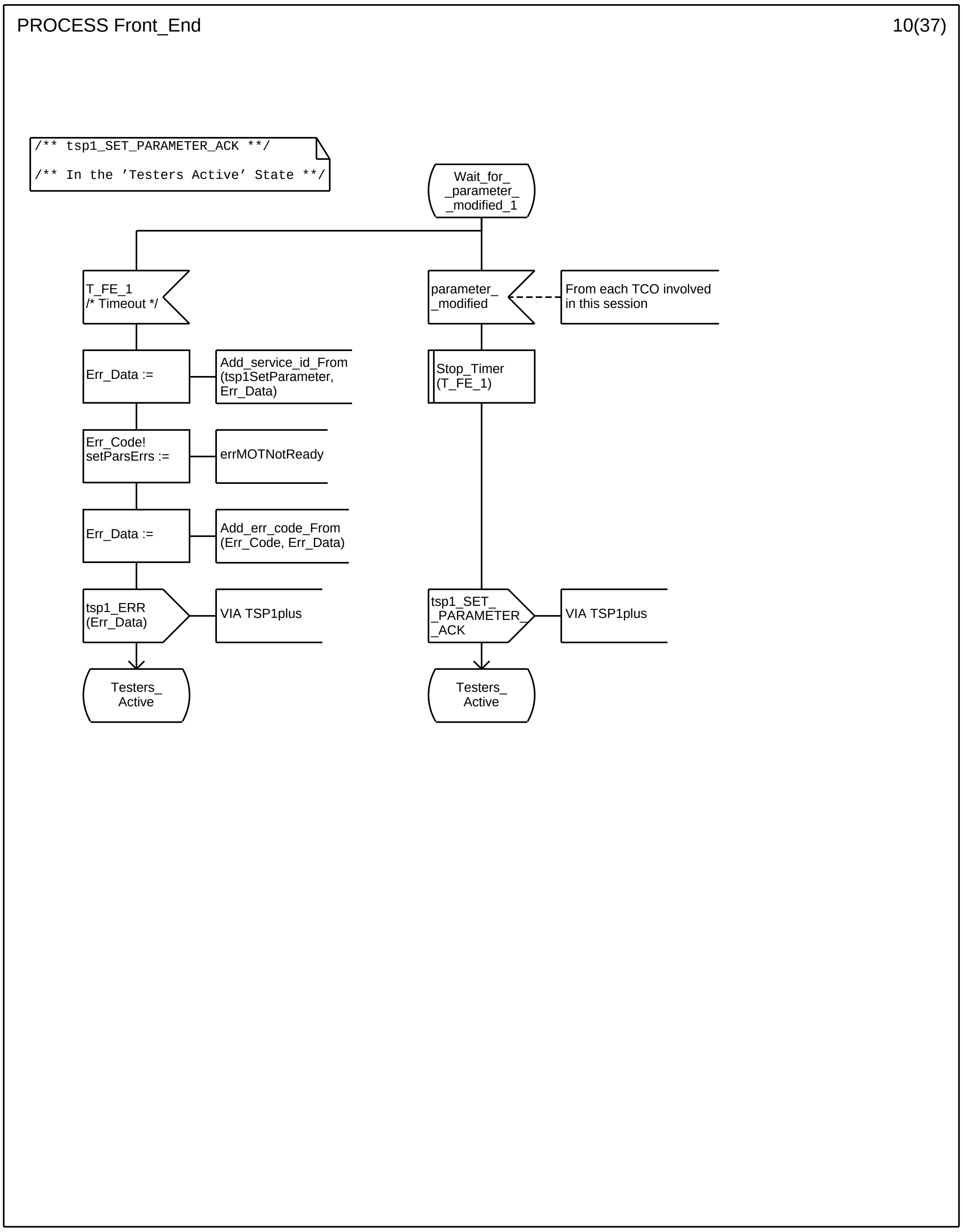

PROCESS Front_End
5(37)
/** tsp1_INIT_COMPLETE **/
Wait_For_
_Tester_Init
T_FE_1
/* Timeout */
*
/* Save other
signals */
tester_
_initialized
From tester
Err_Data :=
Add_service_id_From
(tsp1Init, Err_Data)
Stop_Timer
(T_FE_1)
Err_Code!
initErrs :=
errMOTNotReady
Err_Data :=
Add_err_code_From
(Err_Code, Err_Data)
tsp1_ERR
(Err_Data)
VIA TSP1plus
Sent only when all
testers supported
by Front End have
initialized
tsp1_INIT_
_COMPLETE
VIA TSP1plus
FE_Idle
Testers_
Active

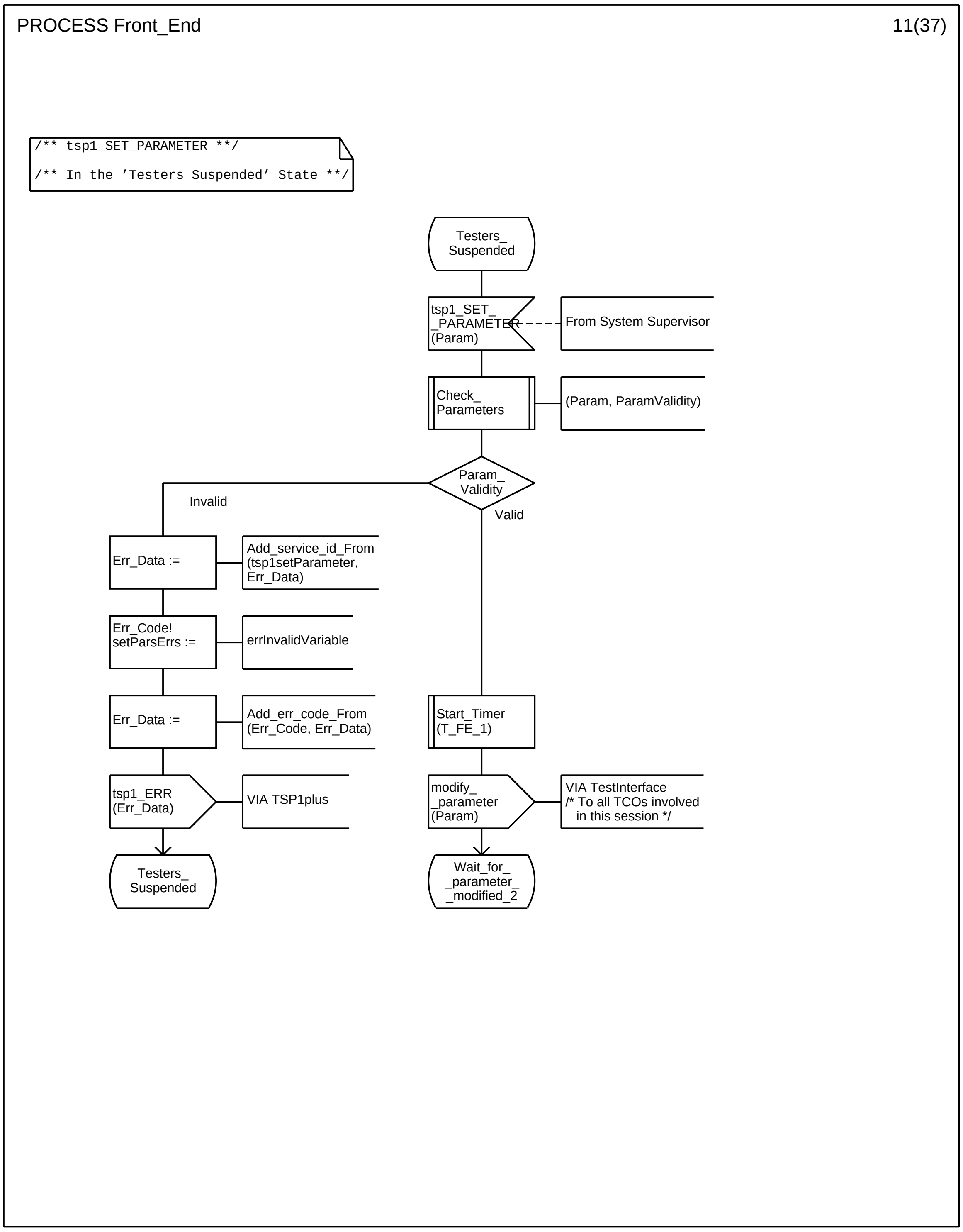

PROCESS Front_End
6(37)
/** tsp1_CHK_CONF **/
Testers_ Active
tsp1_CHK_ _CONF
From System Supervisor
Start_Timer (T_FE_1)
verify_config_ _status
VIA TestInterface /* To all TCOs involved in this session */
Wait_for_ verify_ack


PROCESS Front_End
7(37)
/** tsp1_CHK_CONF_ACK **/
Wait_for_ verify_ack
T_FE_1 /* Timeout */
verify_config_ _status_ack (RunningETS)
From each TCO involved in this session
Stop_Timer (T_FE_1)
RunningETS =ETS
Test must be valid for all running TCOs
FALSE
TRUE
Err_Data :=
Add_service_id_From (tsp1ChkConf, Err_Data)
Err_Code! chkConfErrs :=
errMOTNotReady
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
Testers_ Active
Err_Data :=
Add_service_id_From (tsp1ChkConf, Err_Data)
Err_Code! chkConfErrs :=
errUnknownSession
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
Testers_ Active
tsp1_CHK_ _CONF_ACK
VIA TSP1plus
Testers_ Active

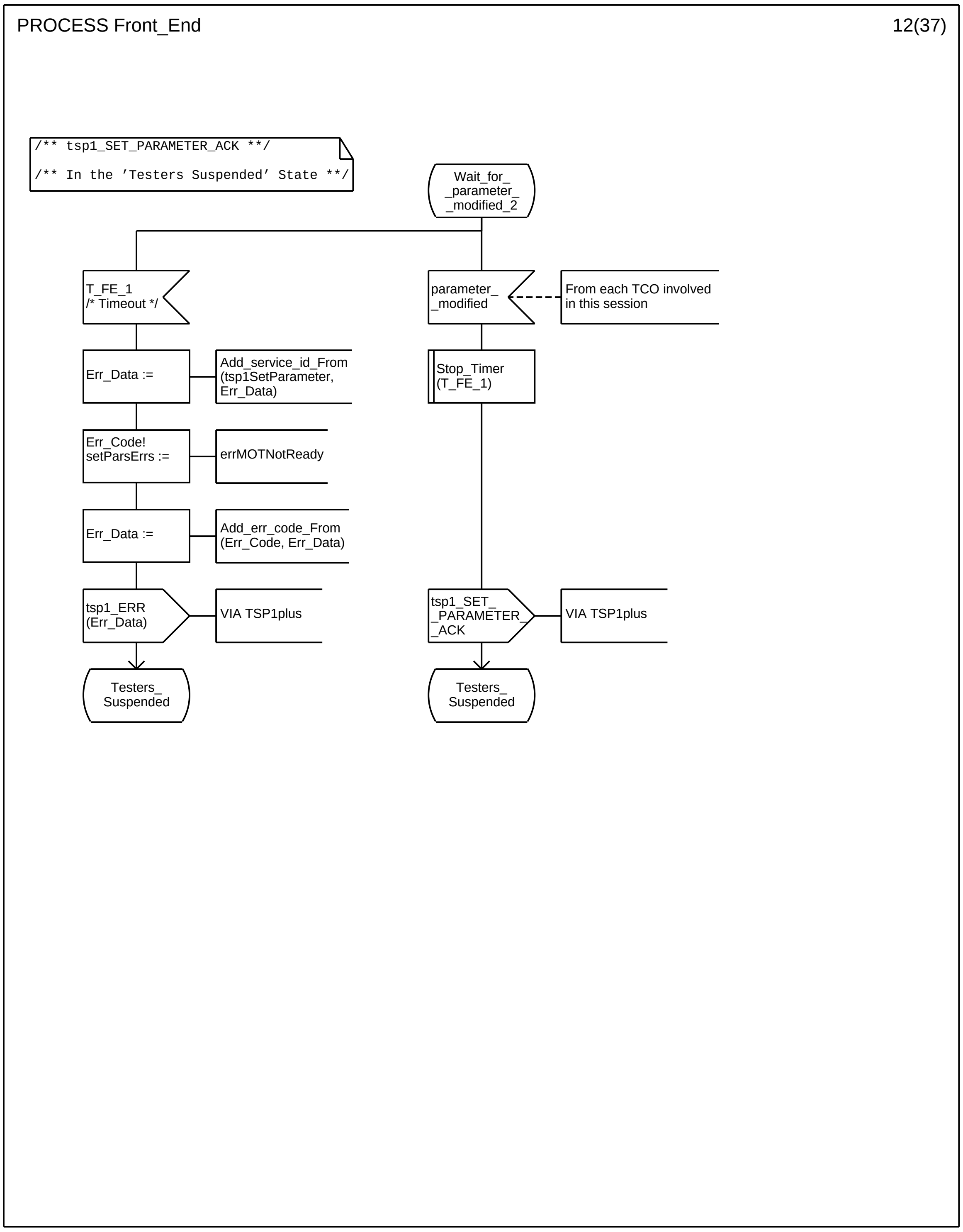

PROCESS Front_End
8(37)
/** tsp1_CHK_CONF **/
/** Error case when message received in Idle State **/
FE_Idle
tsp1_CHK_ _CONF
From System Supervisor
Err_Code! chkConfErrs :=
errMOTNotConnected
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
Err_Data :=
Add_service_id_From (tsp1ChkConf, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
FE_Idle

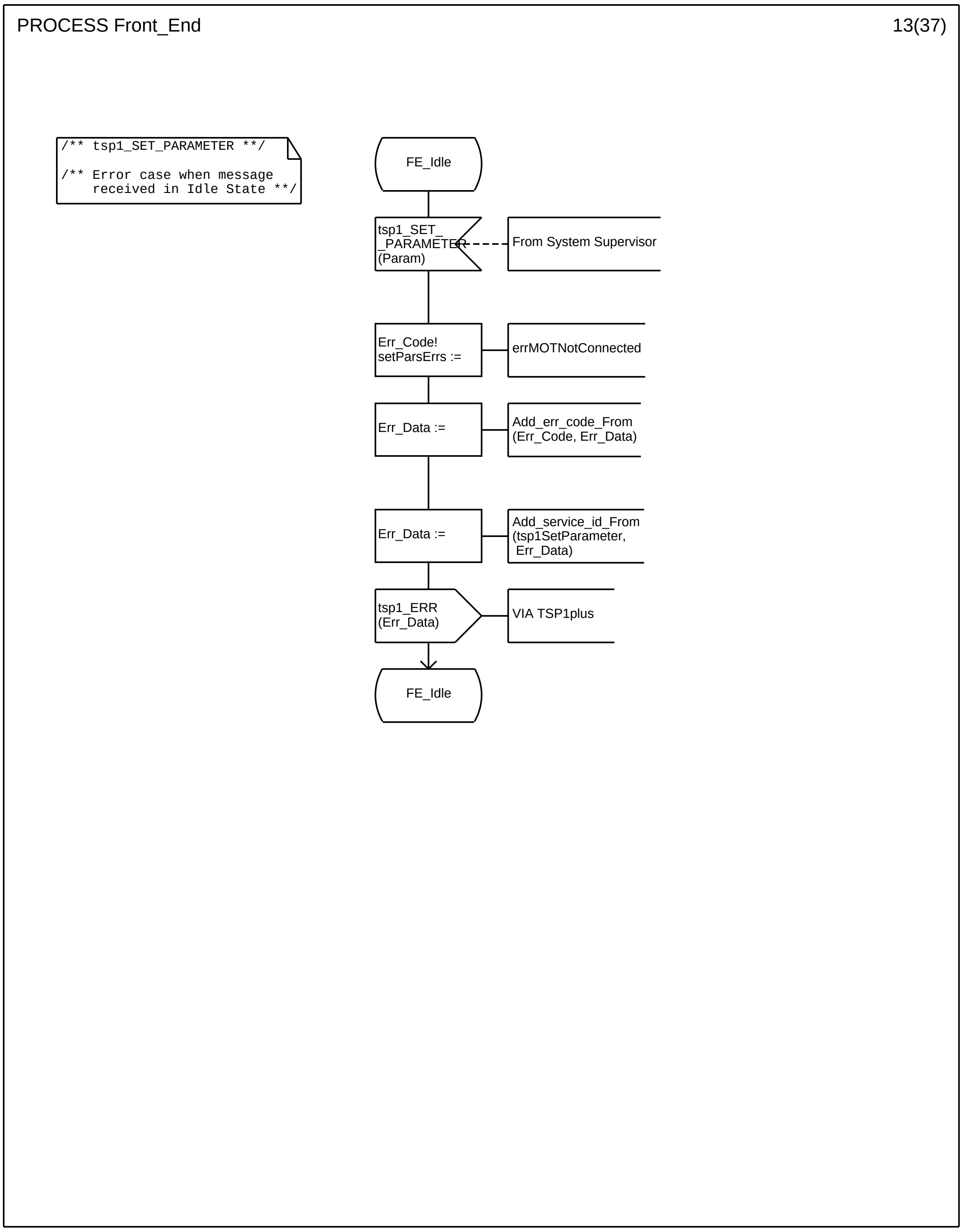

PROCESS Front_End
9(37)
/** tsp1_SET_PARAMETER **/
/** In the 'Testers Active' State **/
Testers_ Active
tsp1_SET_ _PARAMETER (Param)
From System Supervisor
Check_ Parameters
(Param, ParamValidity)
Param_ Validity
Invalid
Valid
Err_Data :=
Add_service_id_From (tsp1setParameter, Err_Data)
Err_Code! setParsErrs :=
errInvalidVariable
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
Testers_ Active
Start_Timer (T_FE_1)
modify_ _parameter (Param)
VIA TestInterface /* To all TCOs involved in this session */
Wait_for_ _parameter_ _modified_1

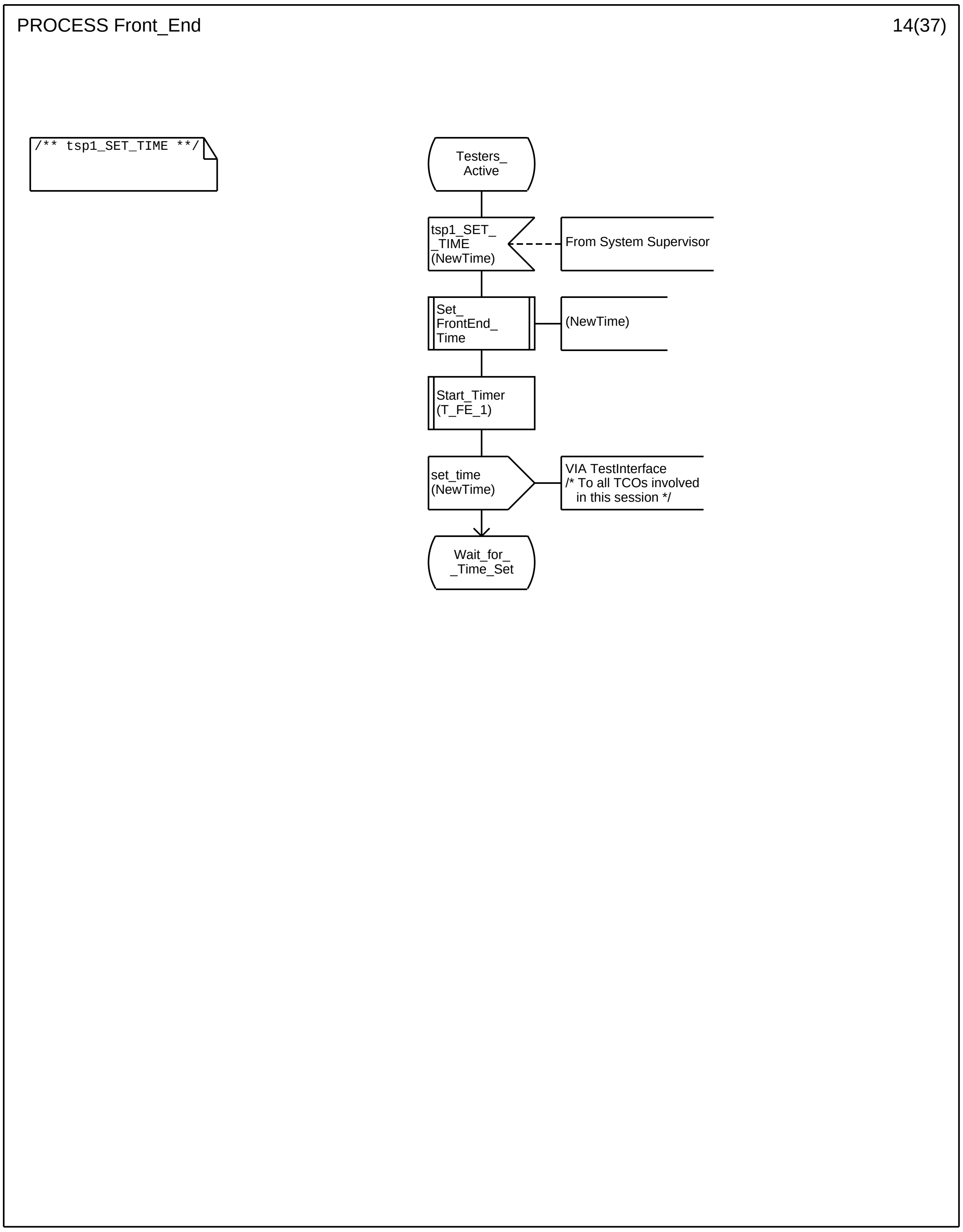

PROCESS Front_End
10(37)
/** tsp1_SET_PARAMETER_ACK **/
/** In the 'Testers Active' State **/
Wait_for_ _parameter_ _modified_1
T_FE_1 /* Timeout */
Err_Data :=
Add_service_id_From (tsp1SetParameter, Err_Data)
Err_Code! setParsErrs :=
errMOTNotReady
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
Testers_ Active
parameter_ _modified
From each TCO involved in this session
Stop_Timer (T_FE_1)
tsp1_SET_ _PARAMETER_ _ACK
VIA TSP1plus
Testers_ Active

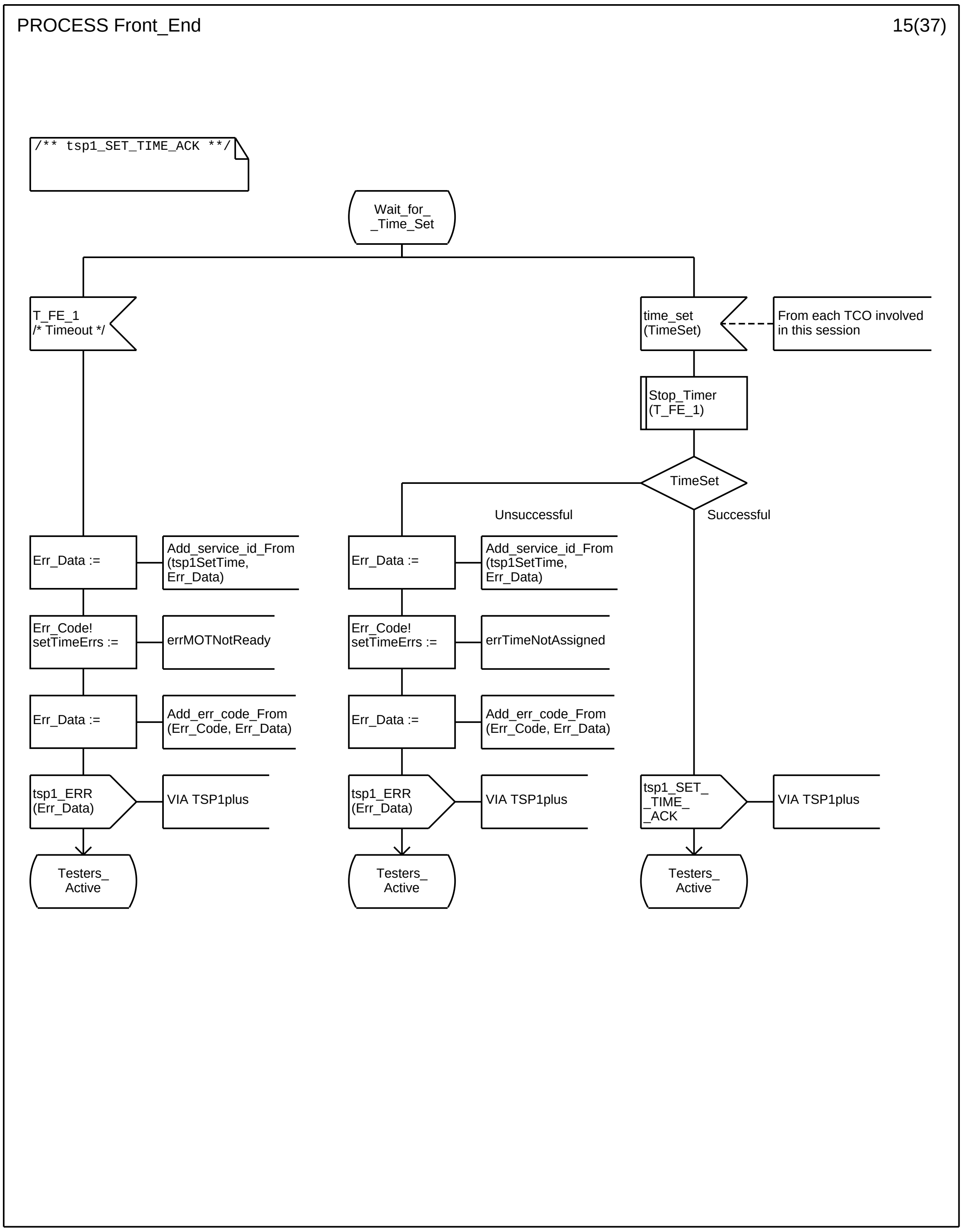


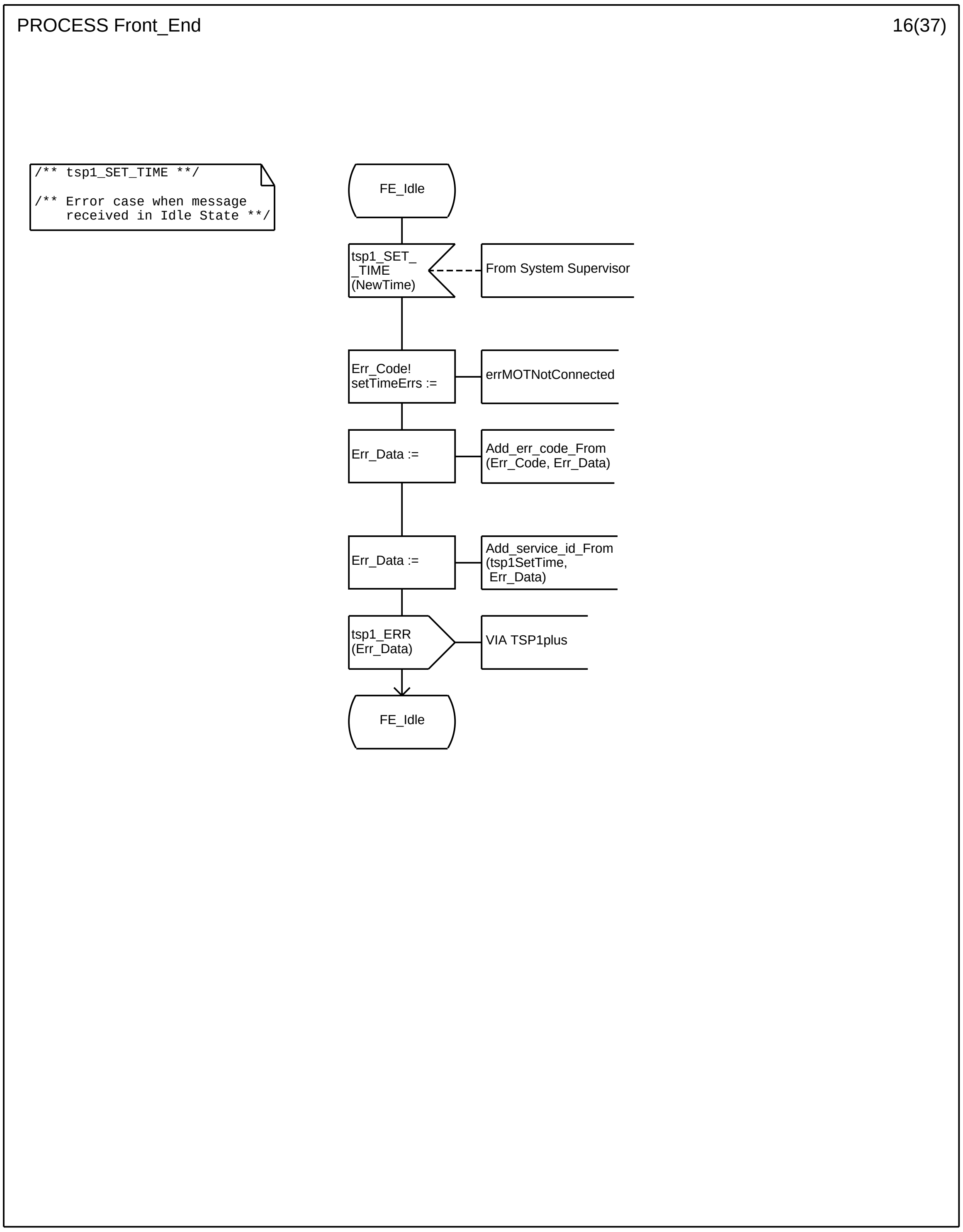
PROCESS Front_End
11(37)
/** tsp1_SET_PARAMETER **/
/** In the 'Testers Suspended' State **/
Testers_ Suspended
tsp1_SET_ _PARAMETER (Param)
From System Supervisor
Check_ Parameters
(Param, ParamValidity)
Param_ Validity
Invalid
Valid
Err_Data :=
Add_service_id_From (tsp1setParameter, Err_Data)
Err_Code! setParsErrs :=
errInvalidVariable
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
Testers_ Suspended
Start_Timer (T_FE_1)
modify_ _parameter (Param)
VIA TestInterface /* To all TCOs involved in this session */
Wait_for_ _parameter_ _modified_2

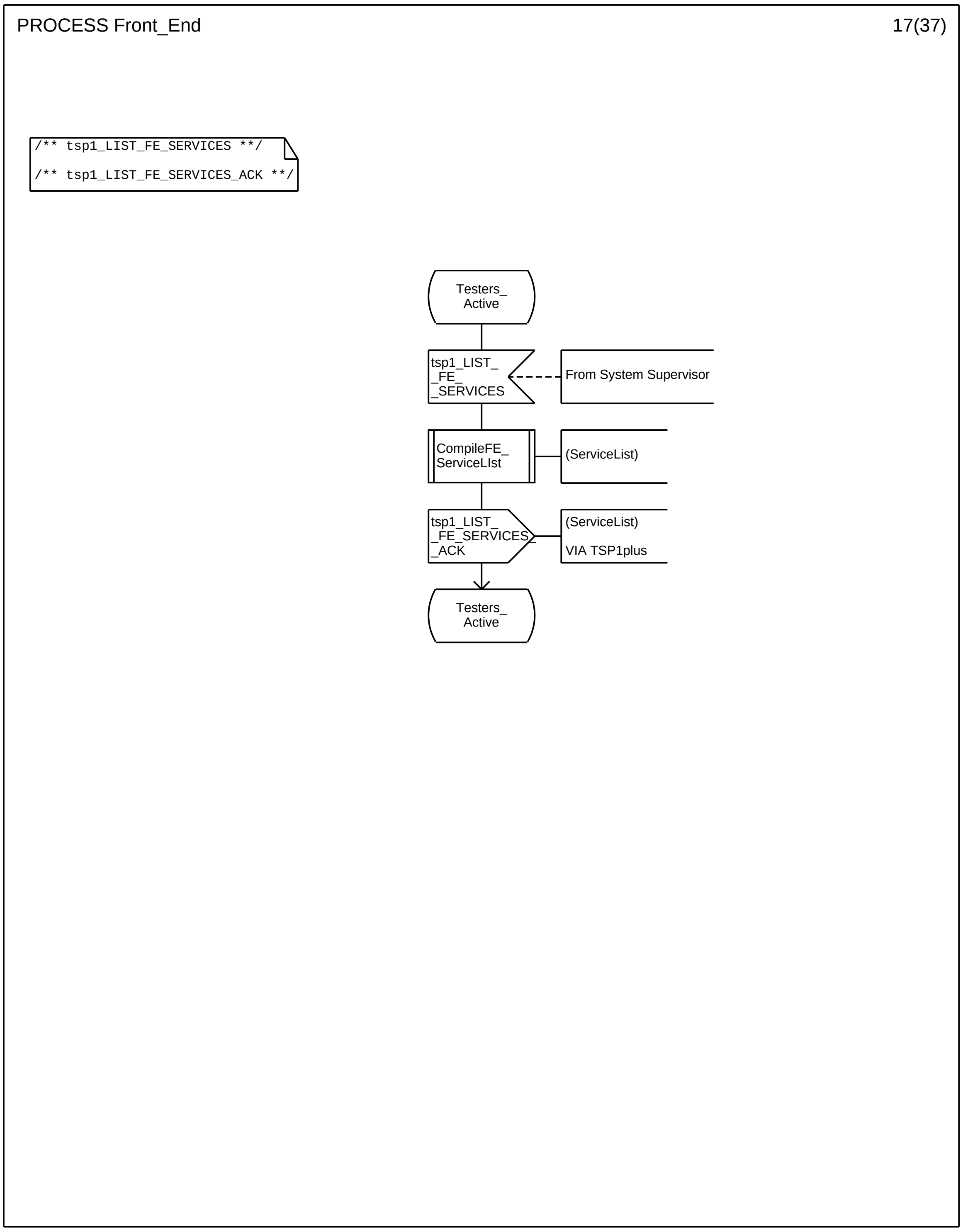

PROCESS Front_End
12(37)
/** tsp1_SET_PARAMETER_ACK **/
/** In the 'Testers Suspended' State **/
Wait_for_ _parameter_ _modified_2
T_FE_1 /* Timeout */
Err_Data :=
Add_service_id_From (tsp1SetParameter, Err_Data)
Err_Code! setParsErrs :=
errMOTNotReady
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
Testers_ Suspended
parameter_ _modified
From each TCO involved in this session
Stop_Timer (T_FE_1)
tsp1_SET_ _PARAMETER_ _ACK
VIA TSP1plus
Testers_ Suspended

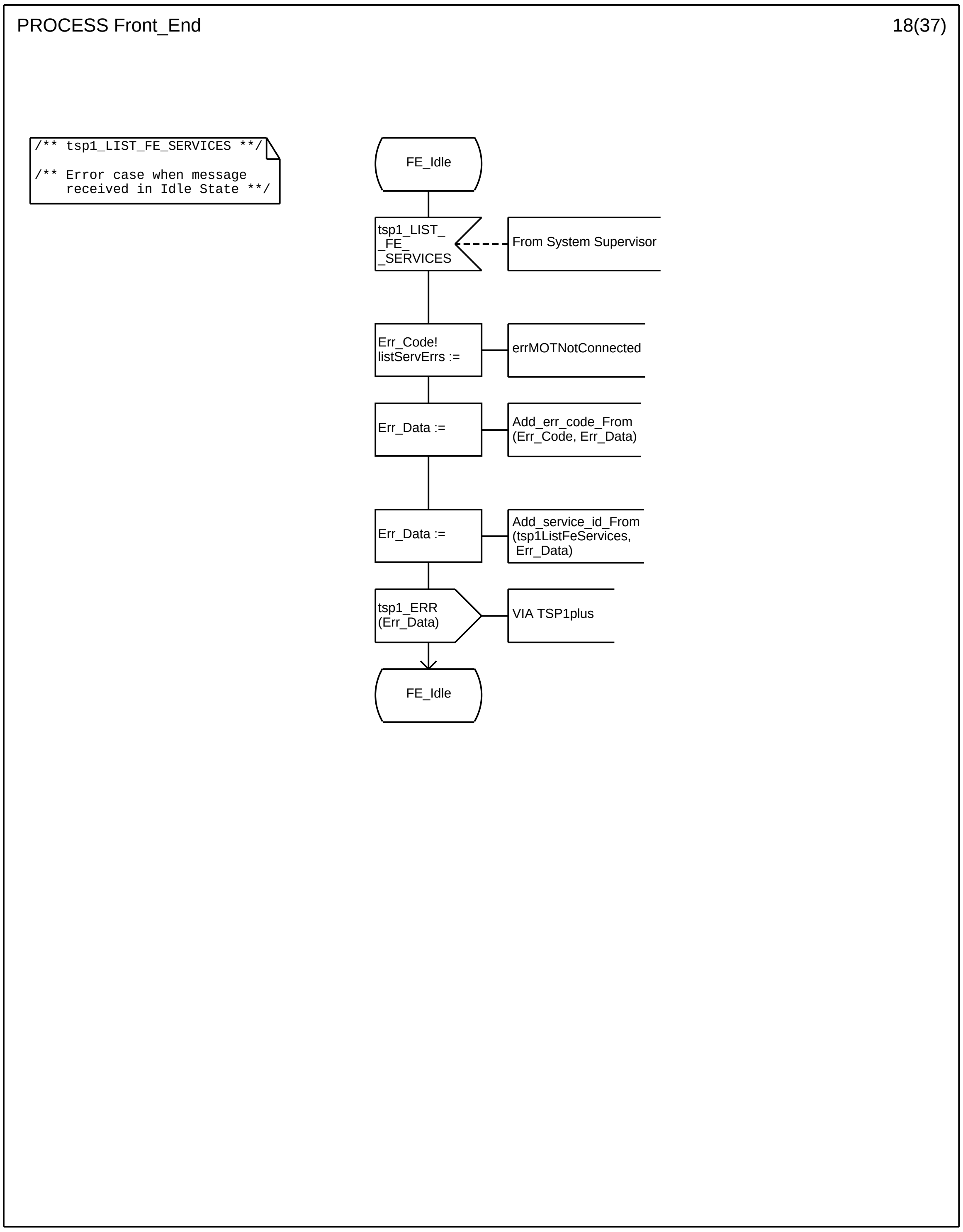

PROCESS Front_End
13(37)
/** tsp1_SET_PARAMETER **/
/** Error case when message received in Idle State **/
FE_Idle
tsp1_SET_ _PARAMETER (Param)
From System Supervisor
Err_Code! setParsErrs :=
errMOTNotConnected
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
Err_Data :=
Add_service_id_From (tsp1SetParameter, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
FE_Idle

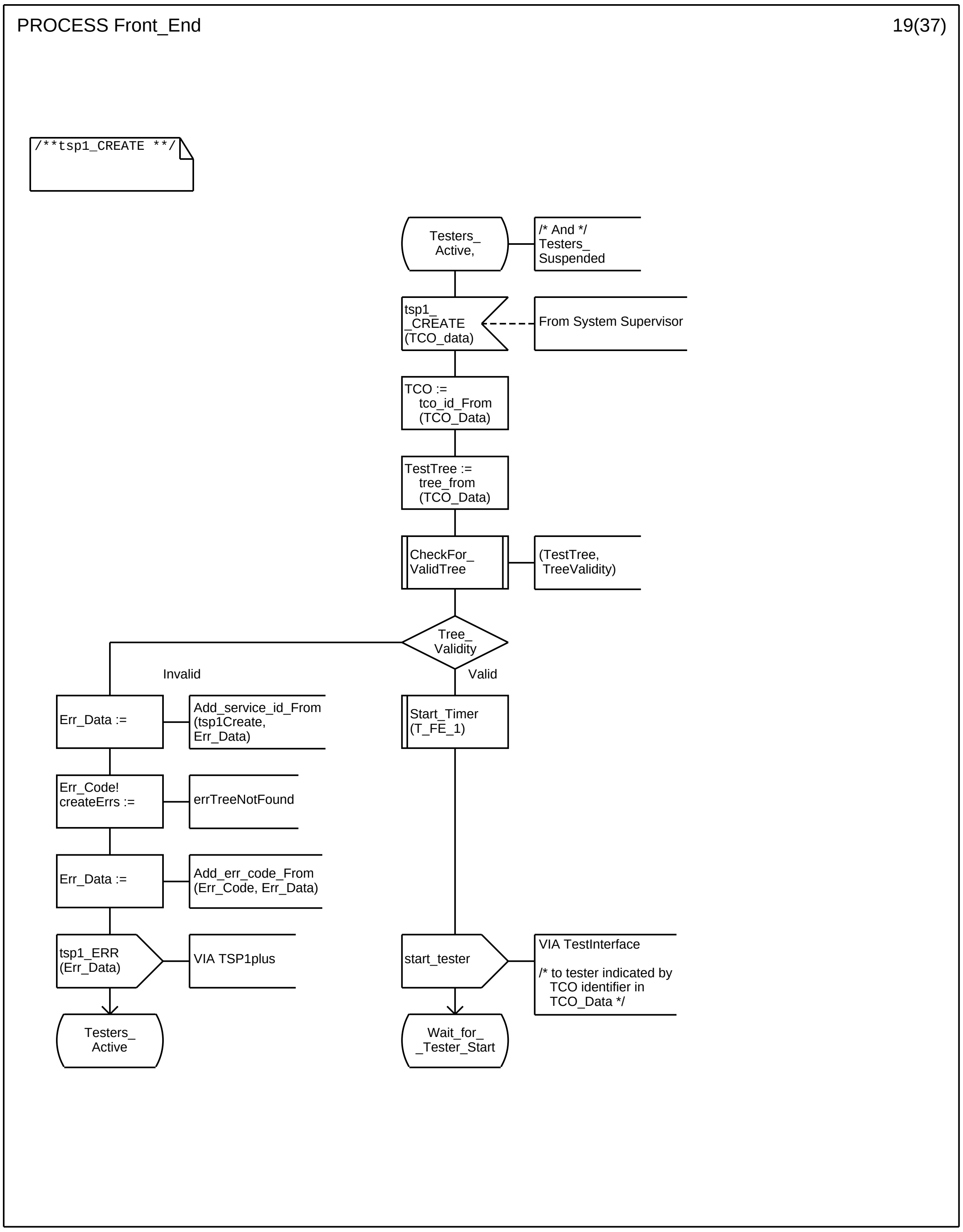

PROCESS Front_End
14(37)
/** tsp1_SET_TIME **/
Testers_
Active
tsp1_SET_
_TIME
(NewTime)
From System Supervisor
Set_
FrontEnd_
Time
(NewTime)
Start_Timer
(T_FE_1)
set_time
(NewTime)
VIA TestInterface
/* To all TCOs involved
in this session */
Wait_for_
_Time_Set

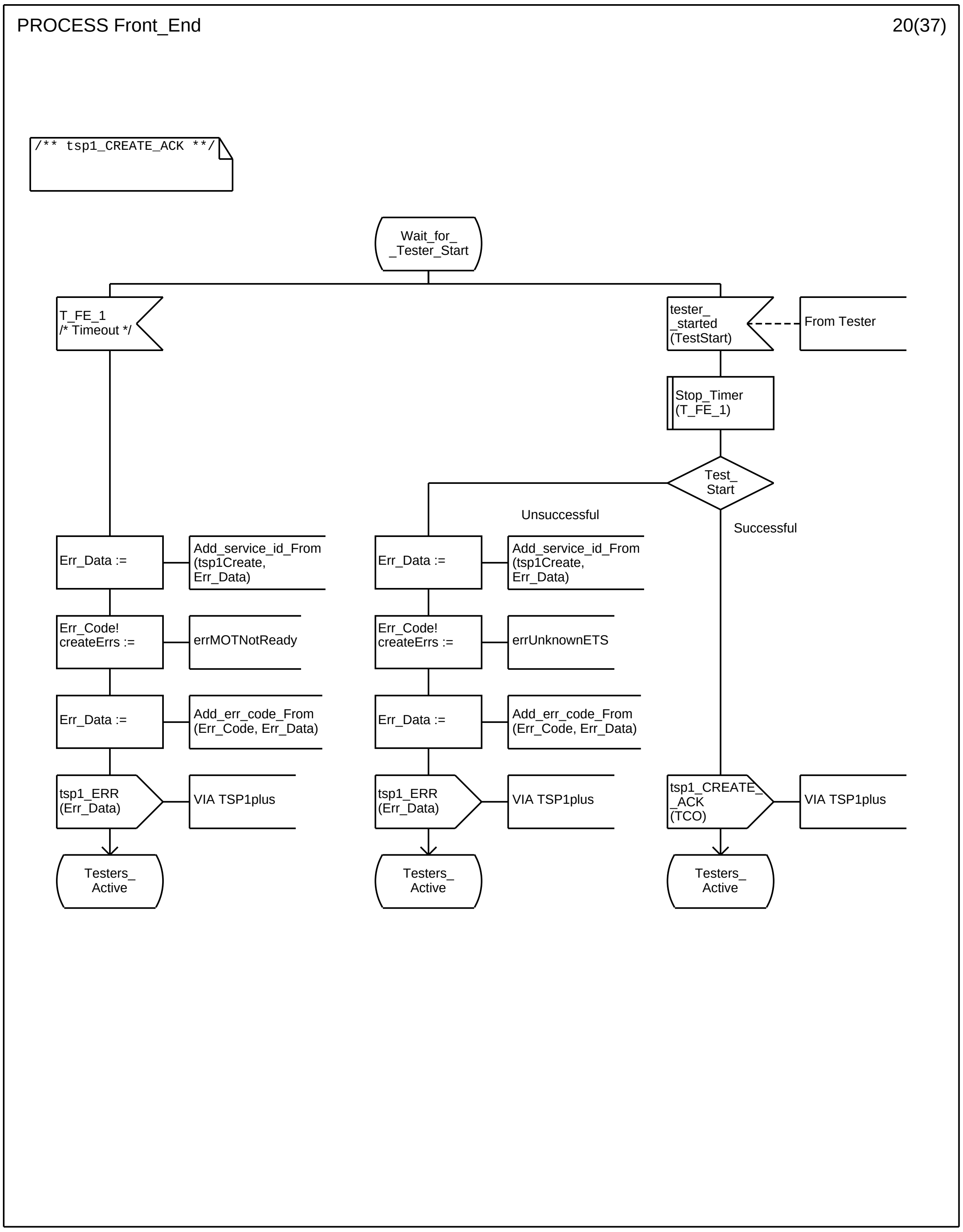

PROCESS Front_End
15(37)
/** tsp1_SET_TIME_ACK **/
Wait_for_ _Time_Set
T_FE_1 /* Timeout */
time_set (TimeSet)
From each TCO involved in this session
Stop_Timer (T_FE_1)
TimeSet
Unsuccessful
Successful
Err_Data :=
Add_service_id_From (tsp1SetTime, Err_Data)
Err_Code! setTimeErrs :=
errMOTNotReady
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
Testers_ Active
Err_Data :=
Add_service_id_From (tsp1SetTime, Err_Data)
Err_Code! setTimeErrs :=
errTimeNotAssigned
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
Testers_ Active
tsp1_SET_ _TIME_ _ACK
VIA TSP1plus
Testers_ Active

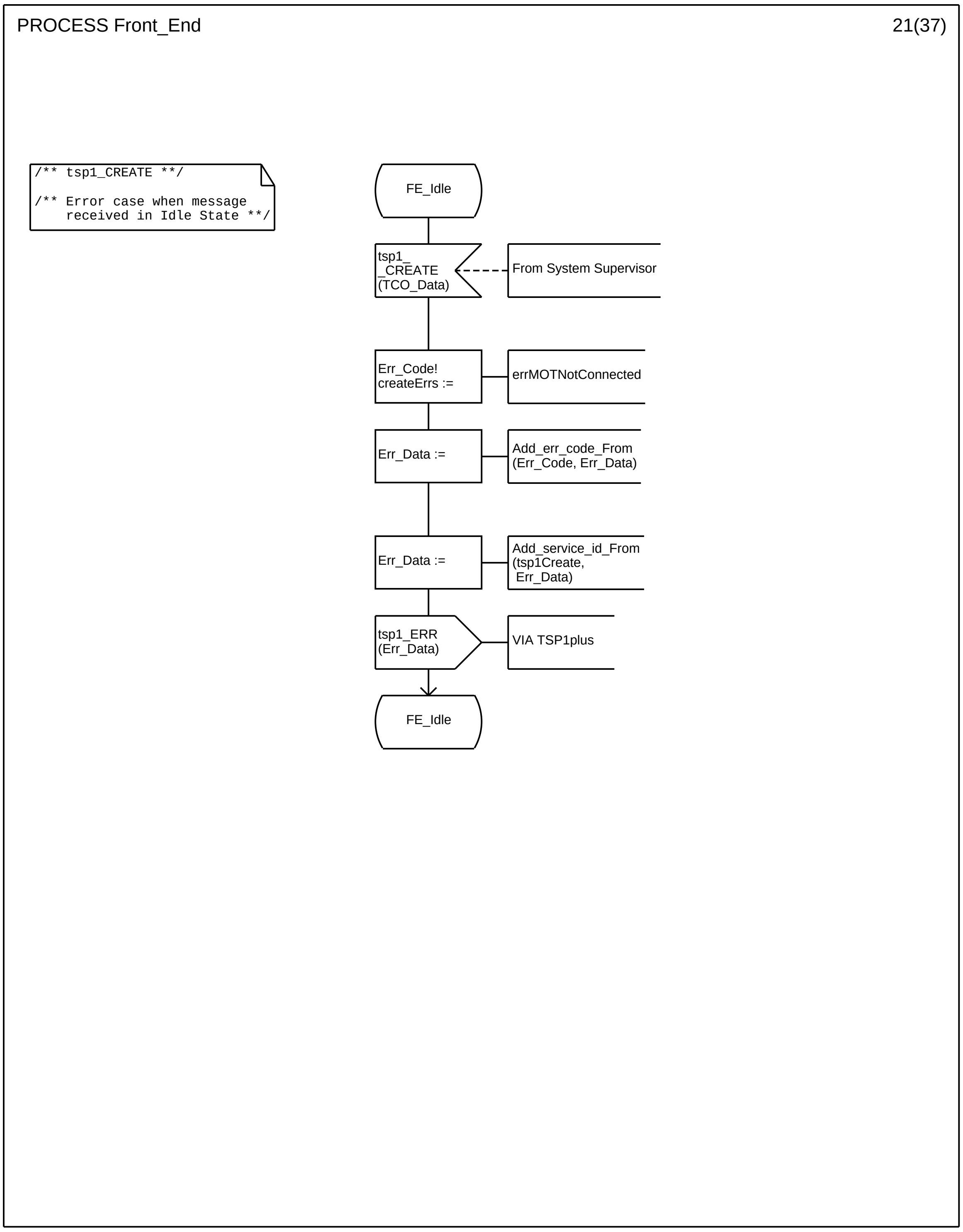

PROCESS Front_End
16(37)
/** tsp1_SET_TIME **/
/** Error case when message received in Idle State **/
FE_Idle
tsp1_SET_ _TIME (NewTime)
From System Supervisor
Err_Code! setTimeErrs :=
errMOTNotConnected
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
Err_Data :=
Add_service_id_From (tsp1SetTime, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
FE_Idle

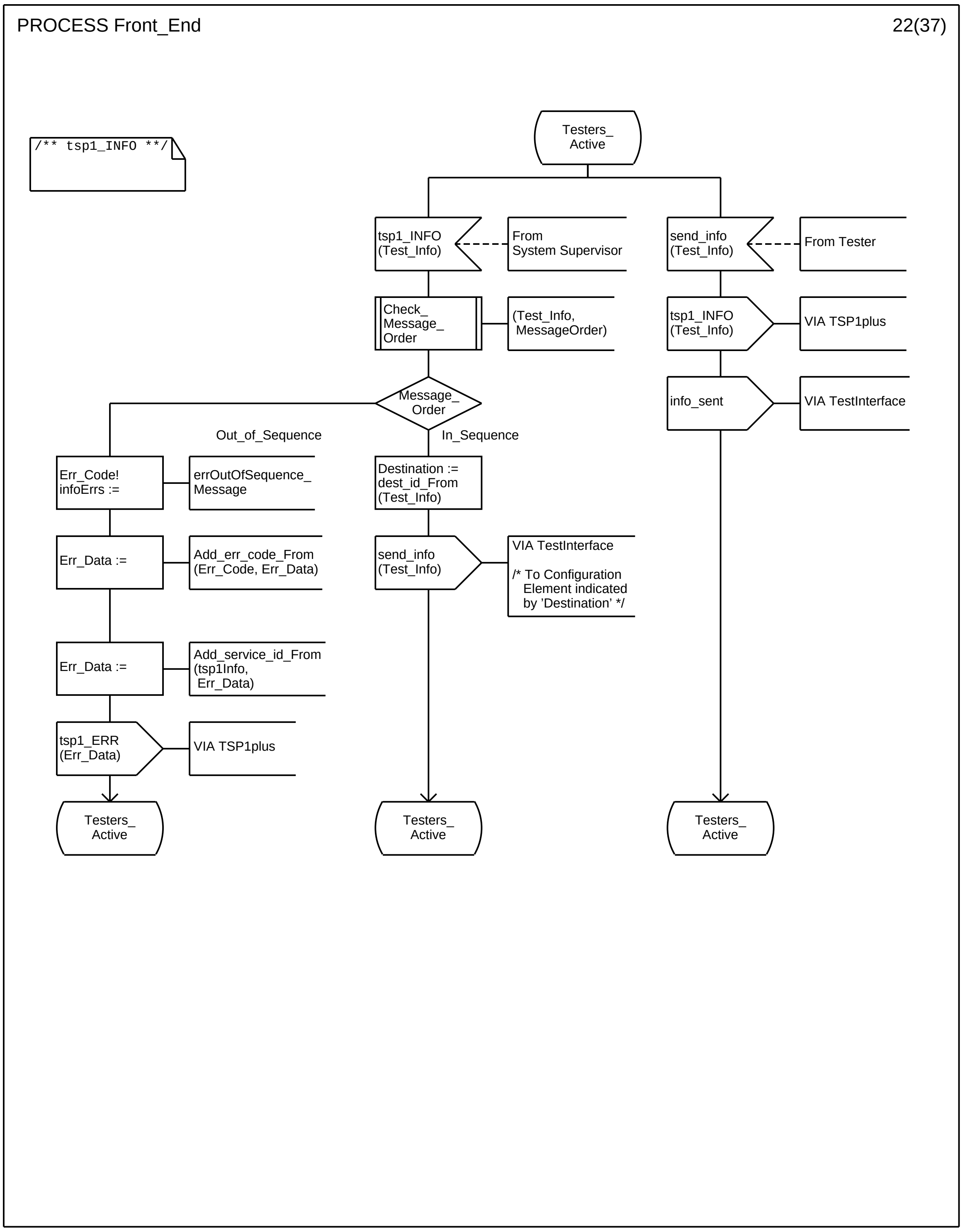

PROCESS Front_End
17(37)
/** tsp1_LIST_FE_SERVICES **/
/** tsp1_LIST_FE_SERVICES_ACK **/
Testers_ Active
tsp1_LIST_ _FE_ _SERVICES
From System Supervisor
CompileFE_ ServiceLIst
(ServiceList)
tsp1_LIST_ _FE_SERVICES_ _ACK
(ServiceList)
VIA TSP1plus
Testers_ Active

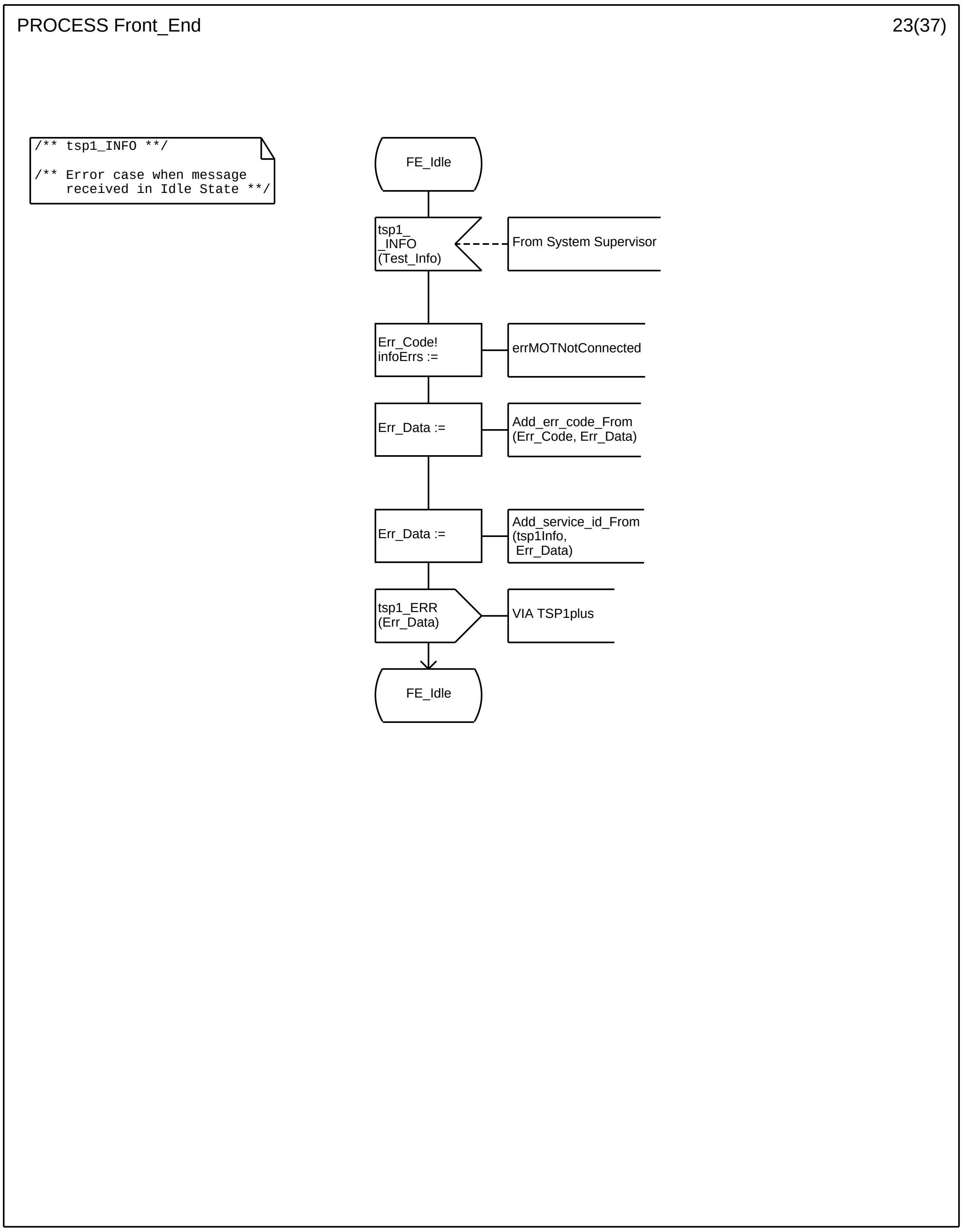

PROCESS Front_End
18(37)
/** tsp1_LIST_FE_SERVICES **/
/** Error case when message received in Idle State **/
FE_Idle
tsp1_LIST_ _FE_ _SERVICES
From System Supervisor
Err_Code! listServErrs :=
errMOTNotConnected
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
Err_Data :=
Add_service_id_From (tsp1ListFeServices, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
FE_Idle

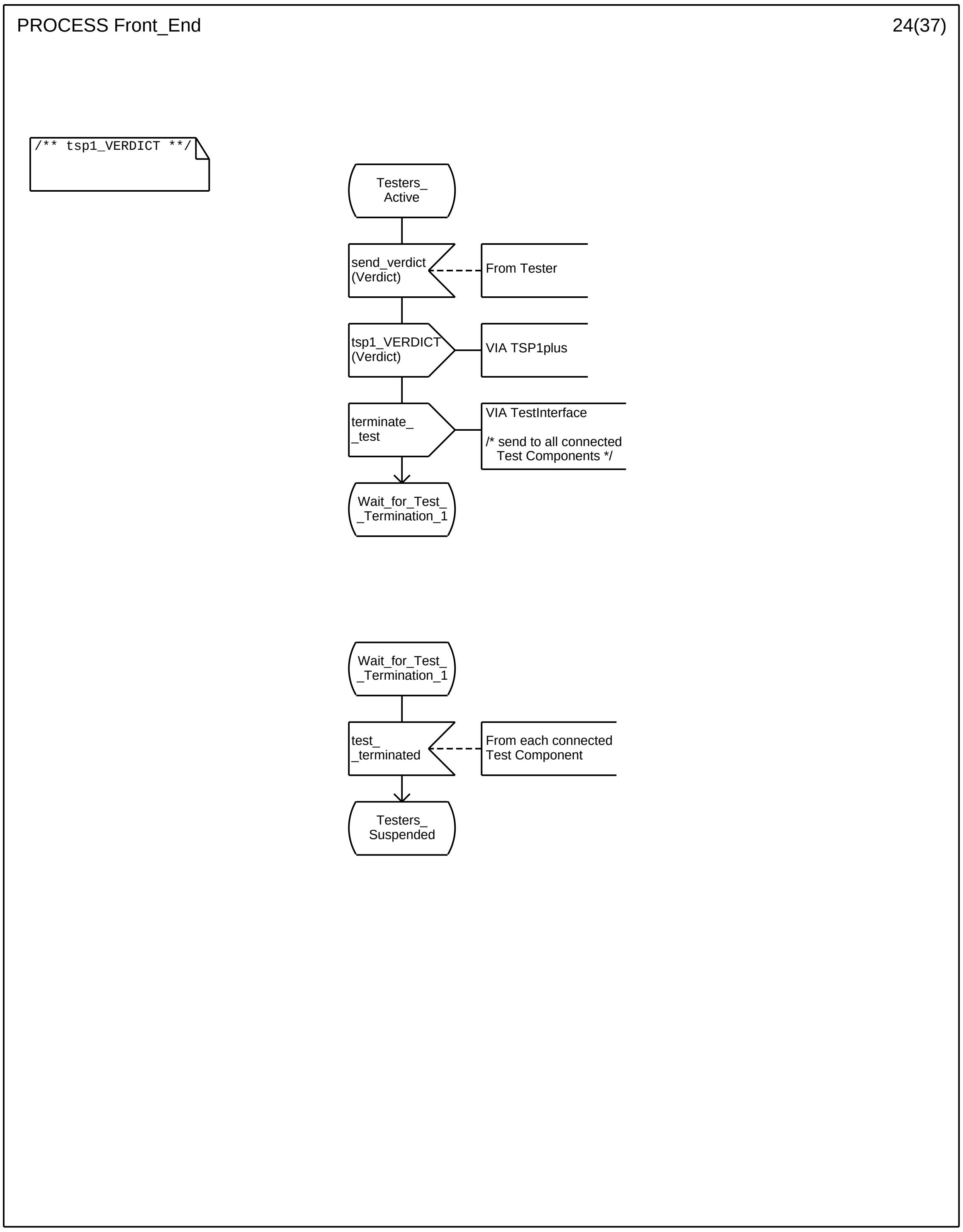

PROCESS Front_End
19(37)
/**tsp1_CREATE **/
Testers_ Active,
/* And */ Testers_ Suspended
tsp1_ _CREATE (TCO_data)
From System Supervisor
TCO := tco_id_From (TCO_Data)
TestTree := tree_from (TCO_Data)
CheckFor_ ValidTree
(TestTree, TreeValidity)
Tree_ Validity
Invalid
Valid
Err_Data :=
Add_service_id_From (tsp1Create, Err_Data)
Err_Code! createErrs :=
errTreeNotFound
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
Testers_ Active
Start_Timer (T_FE_1)
start_tester
VIA TestInterface
/* to tester indicated by TCO identifier in TCO_Data */
Wait_for_ _Tester_Start

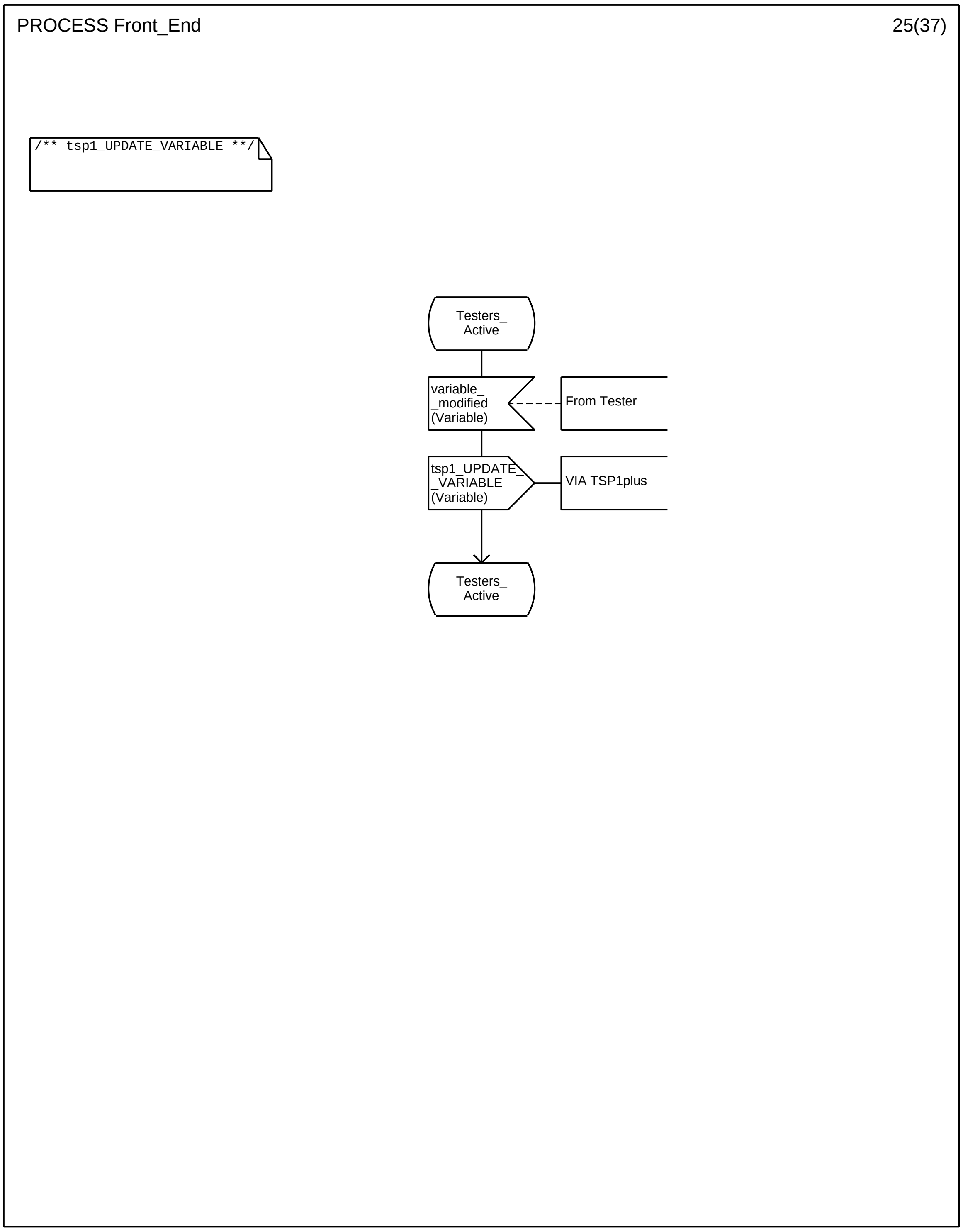

PROCESS Front_End
20(37)
/** tsp1_CREATE_ACK **/
Wait_for_ _Tester_Start
T_FE_1 /* Timeout */
tester_ _started (TestStart)
From Tester
Stop_Timer (T_FE_1)
Test_ Start
Unsuccessful
Successful
Err_Data :=
Add_service_id_From (tsp1Create, Err_Data)
Err_Code! createErrs :=
errMOTNotReady
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
Testers_ Active
Err_Data :=
Add_service_id_From (tsp1Create, Err_Data)
Err_Code! createErrs :=
errUnknownETS
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
Testers_ Active
tsp1_CREATE_ _ACK (TCO)
VIA TSP1plus
Testers_ Active

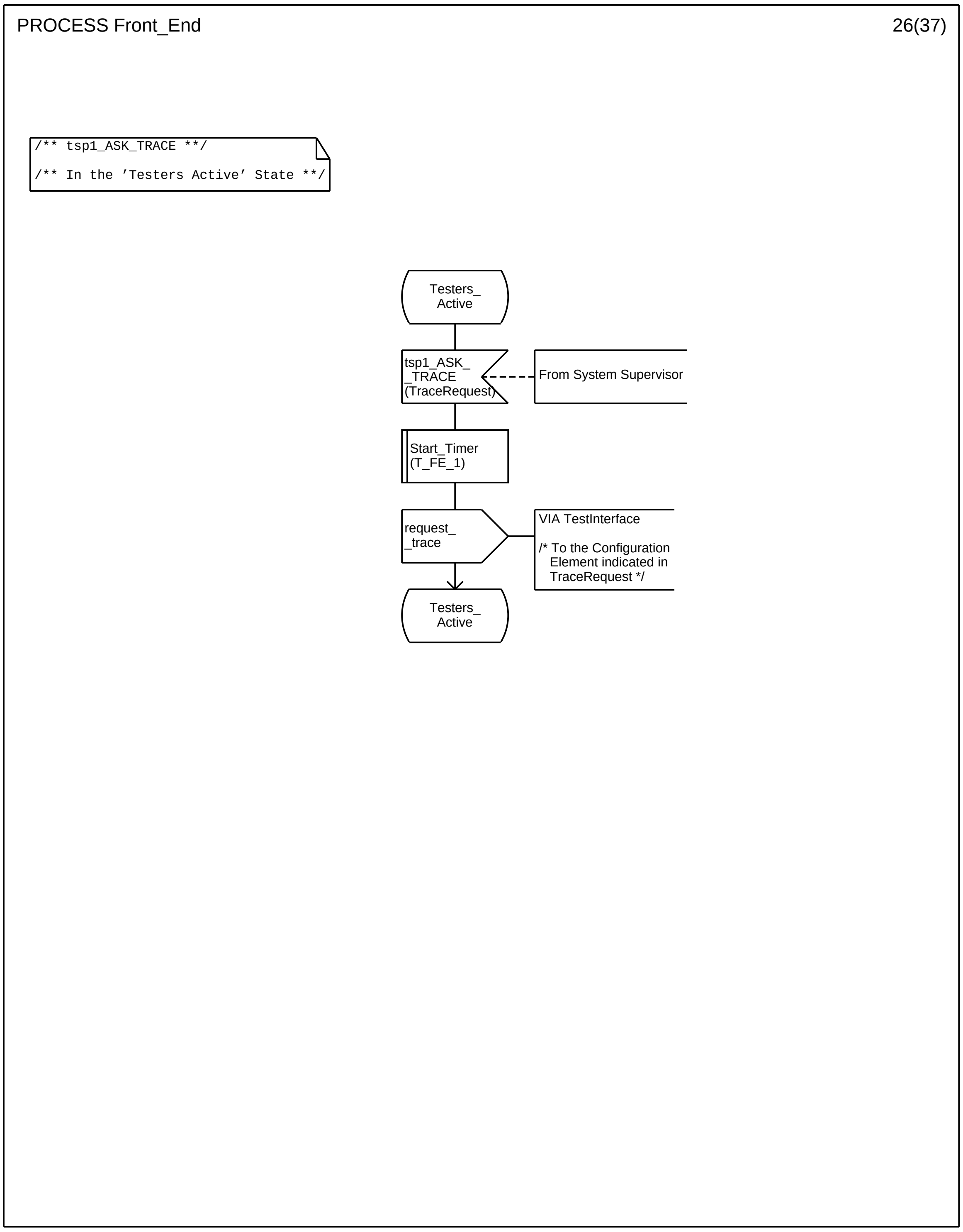

PROCESS Front_End
21(37)
/** tsp1_CREATE **/
/** Error case when message received in Idle State **/
FE_Idle
tsp1_ _CREATE (TCO_Data)
From System Supervisor
Err_Code! createErrs :=
errMOTNotConnected
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
Err_Data :=
Add_service_id_From (tsp1Create, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
FE_Idle

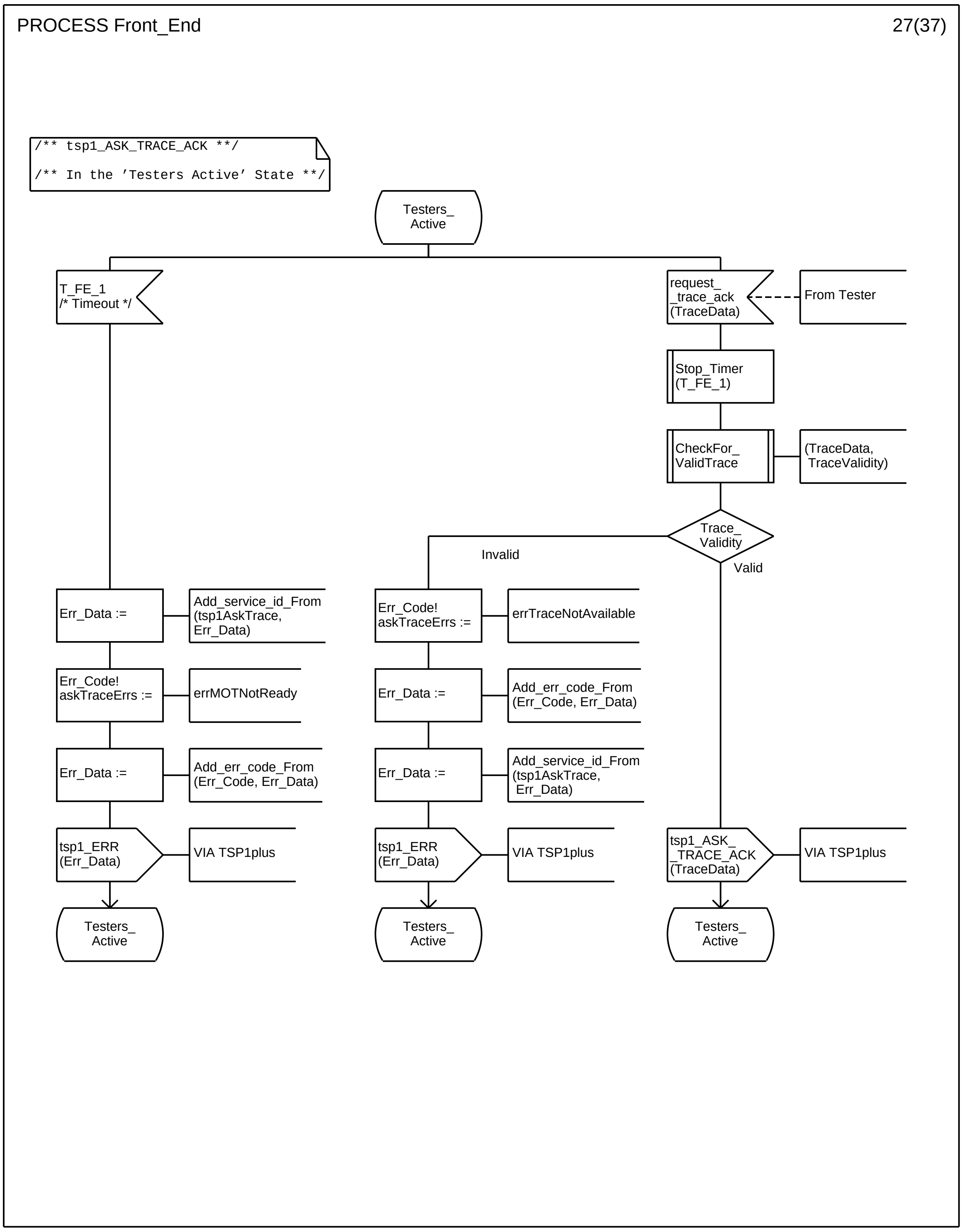

PROCESS Front_End
22(37)
/** tsp1_INFO **/
Testers_ Active
tsp1_INFO (Test_Info)
From System Supervisor
send_info (Test_Info)
From Tester
Check_ Message_ Order
(Test_Info, MessageOrder)
tsp1_INFO (Test_Info)
VIA TSP1plus
Message_ Order
info_sent
VIA TestInterface
Out_of_Sequence
In_Sequence
Err_Code! infoErrs :=
errOutOfSequence_ Message
Destination := dest_id_From (Test_Info)
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
send_info (Test_Info)
VIA TestInterface
/* To Configuration Element indicated by 'Destination' */
Err_Data :=
Add_service_id_From (tsp1Info, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
Testers_ Active
Testers_ Active
Testers_ Active

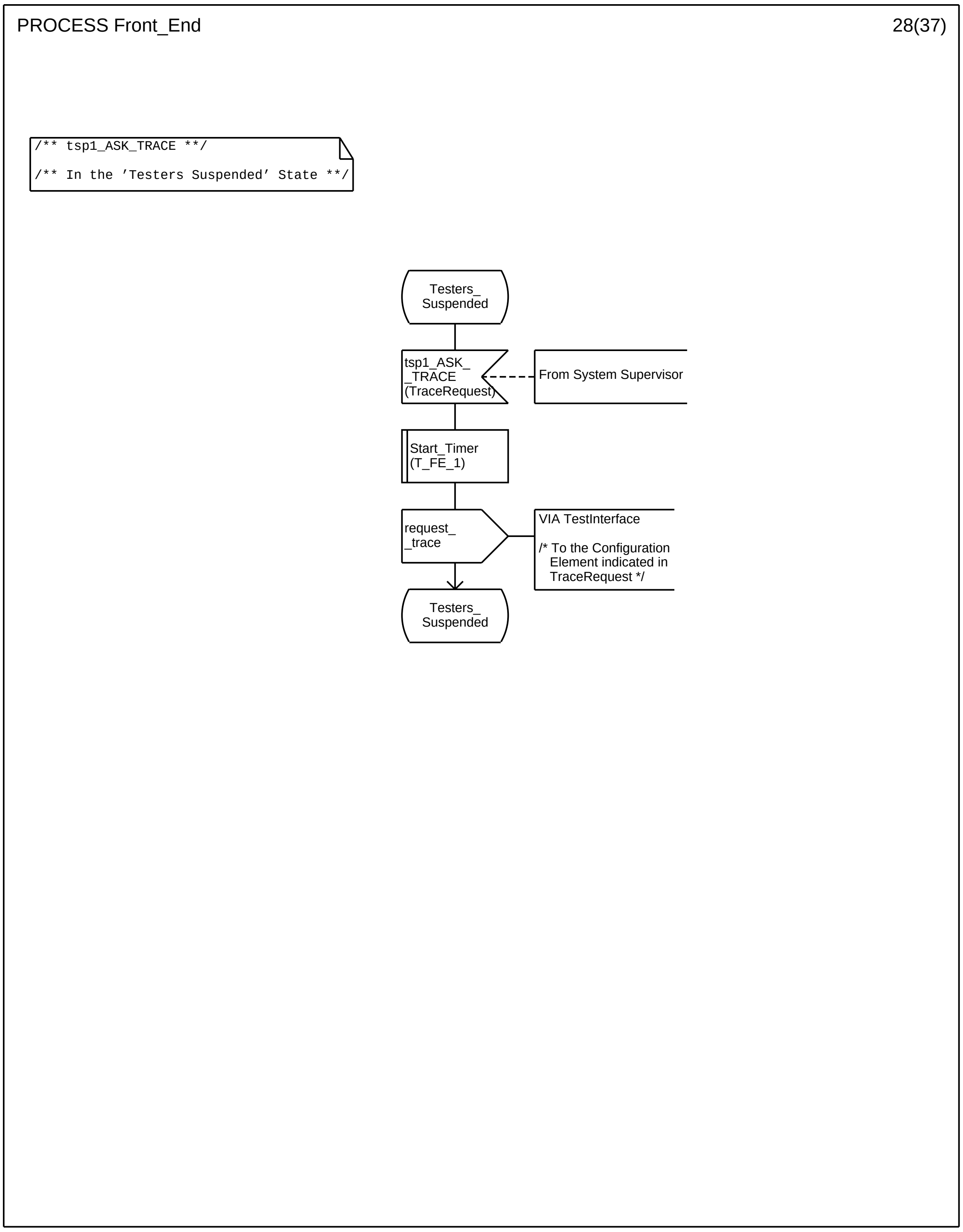

PROCESS Front_End
23(37)
/** tsp1_INFO **/
/** Error case when message received in Idle State **/
FE_Idle
tsp1_ _INFO (Test_Info)
From System Supervisor
Err_Code! infoErrs :=
errMOTNotConnected
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
Err_Data :=
Add_service_id_From (tsp1Info, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
FE_Idle

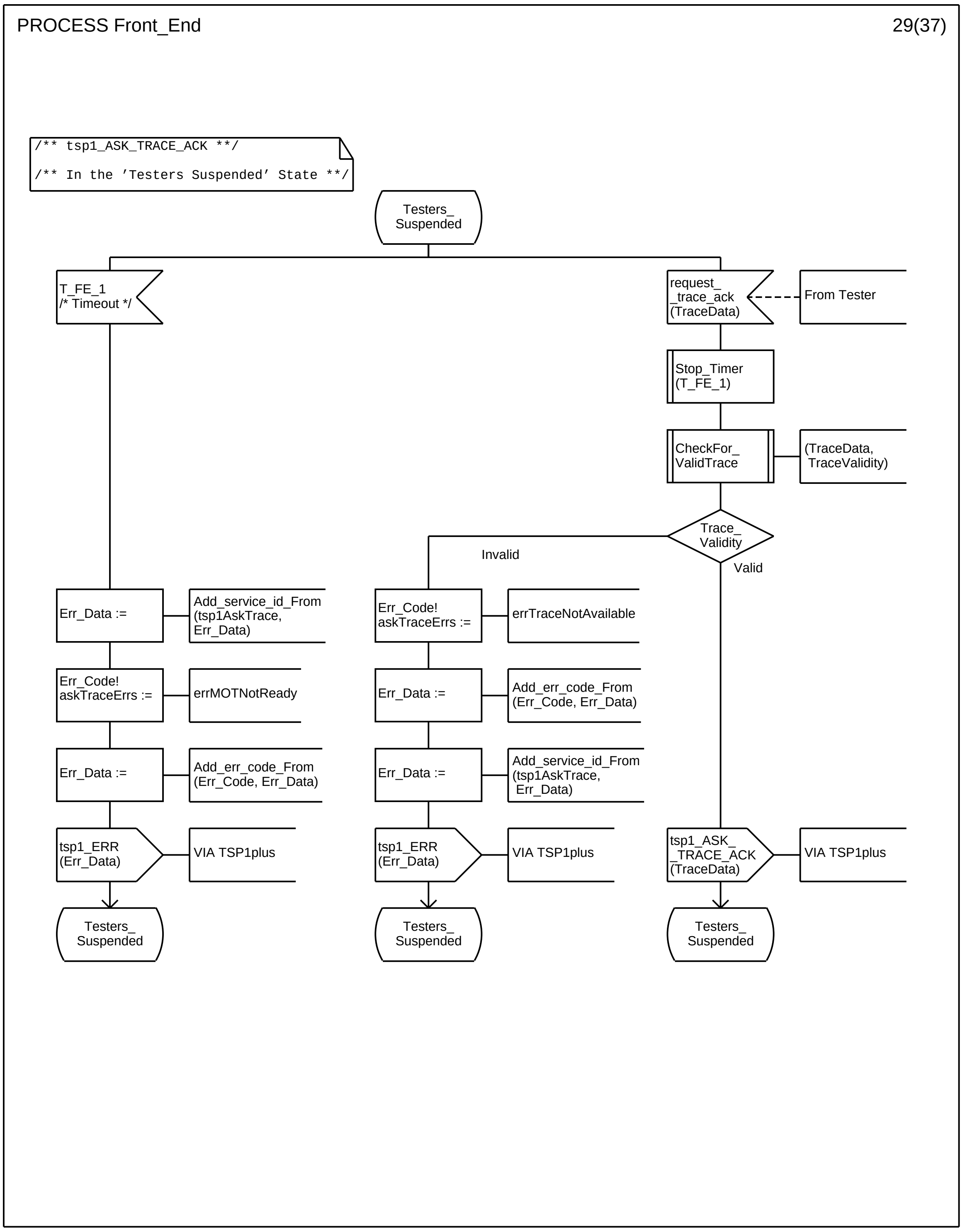

PROCESS Front_End
24(37)
/** tsp1_VERDICT **/
Testers_ Active
send_verdict (Verdict)
From Tester
tsp1_VERDICT (Verdict)
VIA TSP1plus
terminate_ _test
VIA TestInterface
/* send to all connected Test Components */
Wait_for_Test_ _Termination_1
Wait_for_Test_ _Termination_1
test_ _terminated
From each connected Test Component
Testers_ Suspended

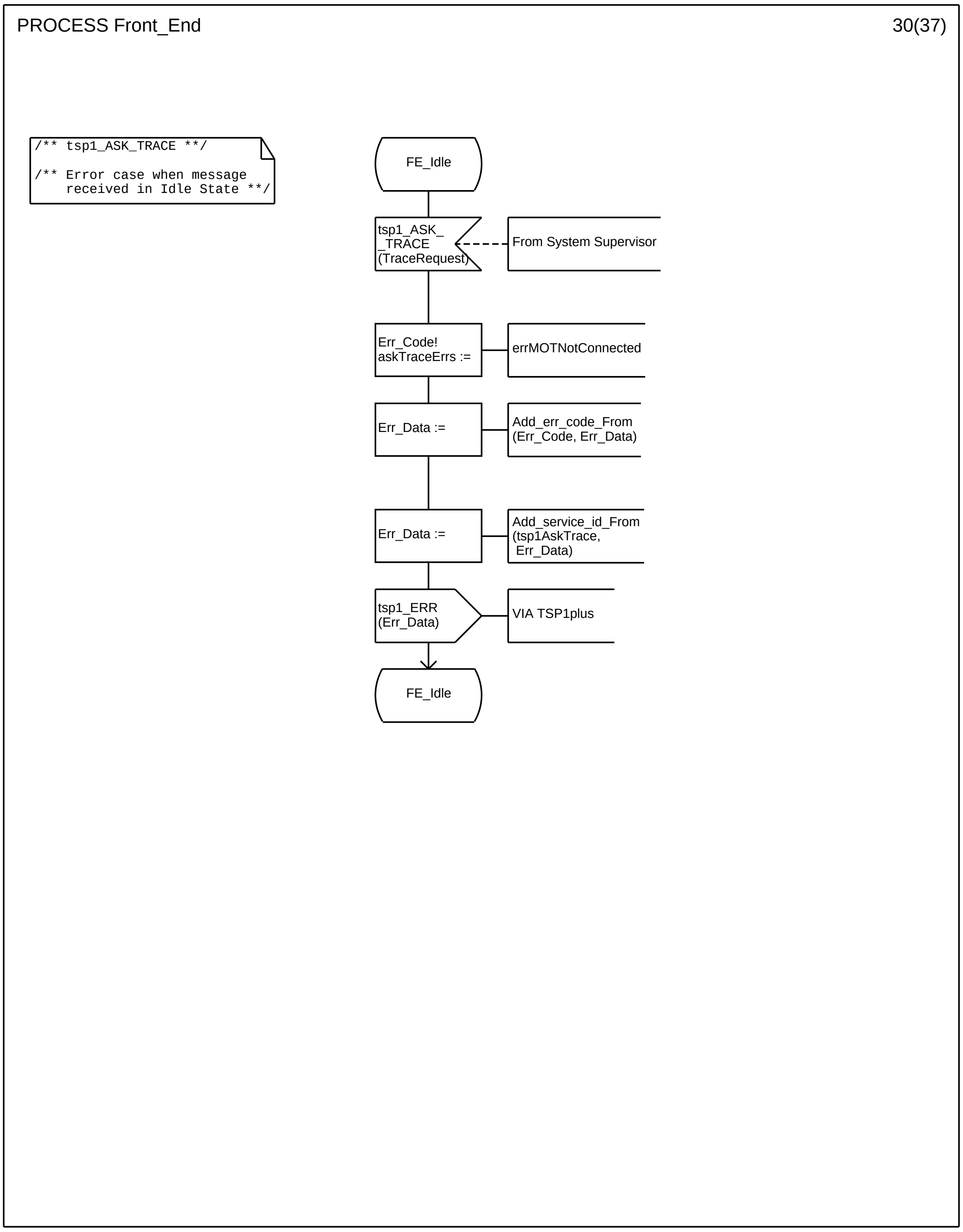

PROCESS Front_End
25(37)
/** tsp1_UPDATE_VARIABLE **/
Testers_
Active
variable_
_modified
(Variable)
From Tester
tsp1_UPDATE_
_VARIABLE
(Variable)
VIA TSP1plus
Testers_
Active

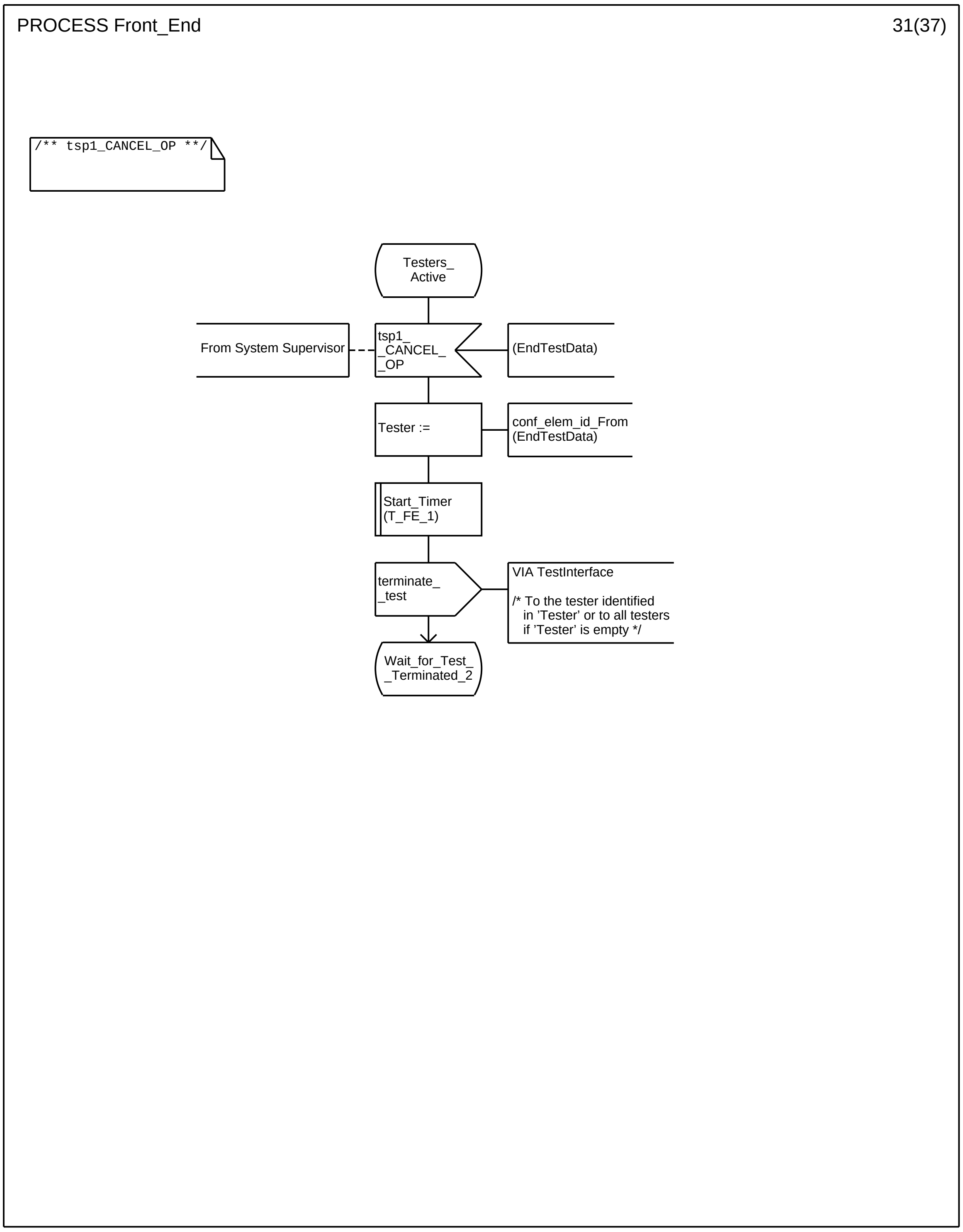

PROCESS Front_End
26(37)
/** tsp1_ASK_TRACE **/
/** In the 'Testers Active' State **/
Testers_ Active
tsp1_ASK_ _TRACE (TraceRequest)
From System Supervisor
Start_Timer (T_FE_1)
request_ _trace
VIA TestInterface
/* To the Configuration Element indicated in TraceRequest */
Testers_ Active

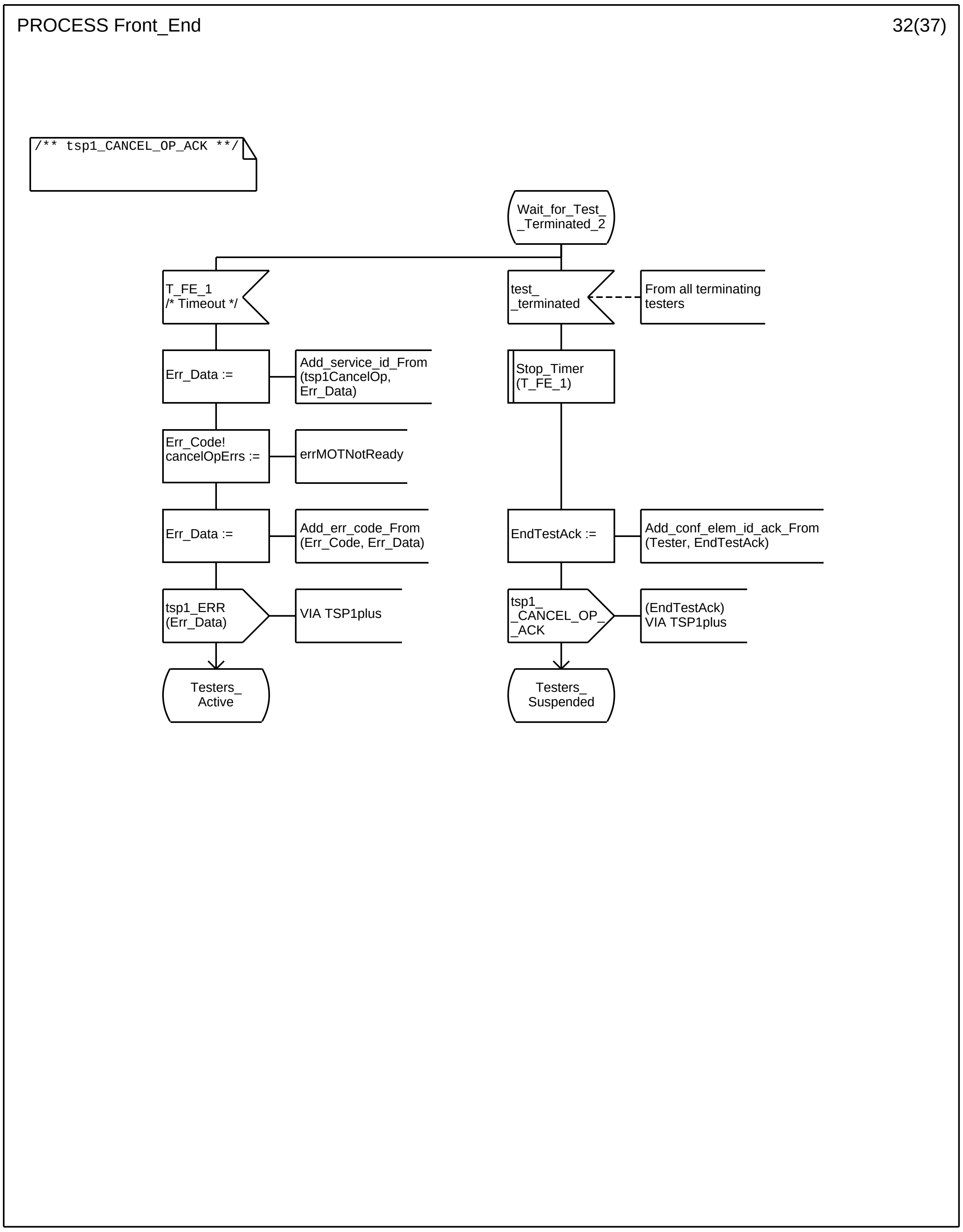

PROCESS Front_End
27(37)
/** tsp1_ASK_TRACE_ACK **/
/** In the 'Testers Active' State **/
Testers_ Active
T_FE_1 /* Timeout */
request_ _trace_ack (TraceData)
From Tester
Stop_Timer (T_FE_1)
CheckFor_ ValidTrace
(TraceData, TraceValidity)
Trace_ Validity
Invalid
Valid
Err_Data :=
Add_service_id_From (tsp1AskTrace, Err_Data)
Err_Code! askTraceErrs :=
errMOTNotReady
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
Testers_ Active
Err_Code! askTraceErrs :=
errTraceNotAvailable
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
Err_Data :=
Add_service_id_From (tsp1AskTrace, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
Testers_ Active
tsp1_ASK_ _TRACE_ACK (TraceData)
VIA TSP1plus
Testers_ Active

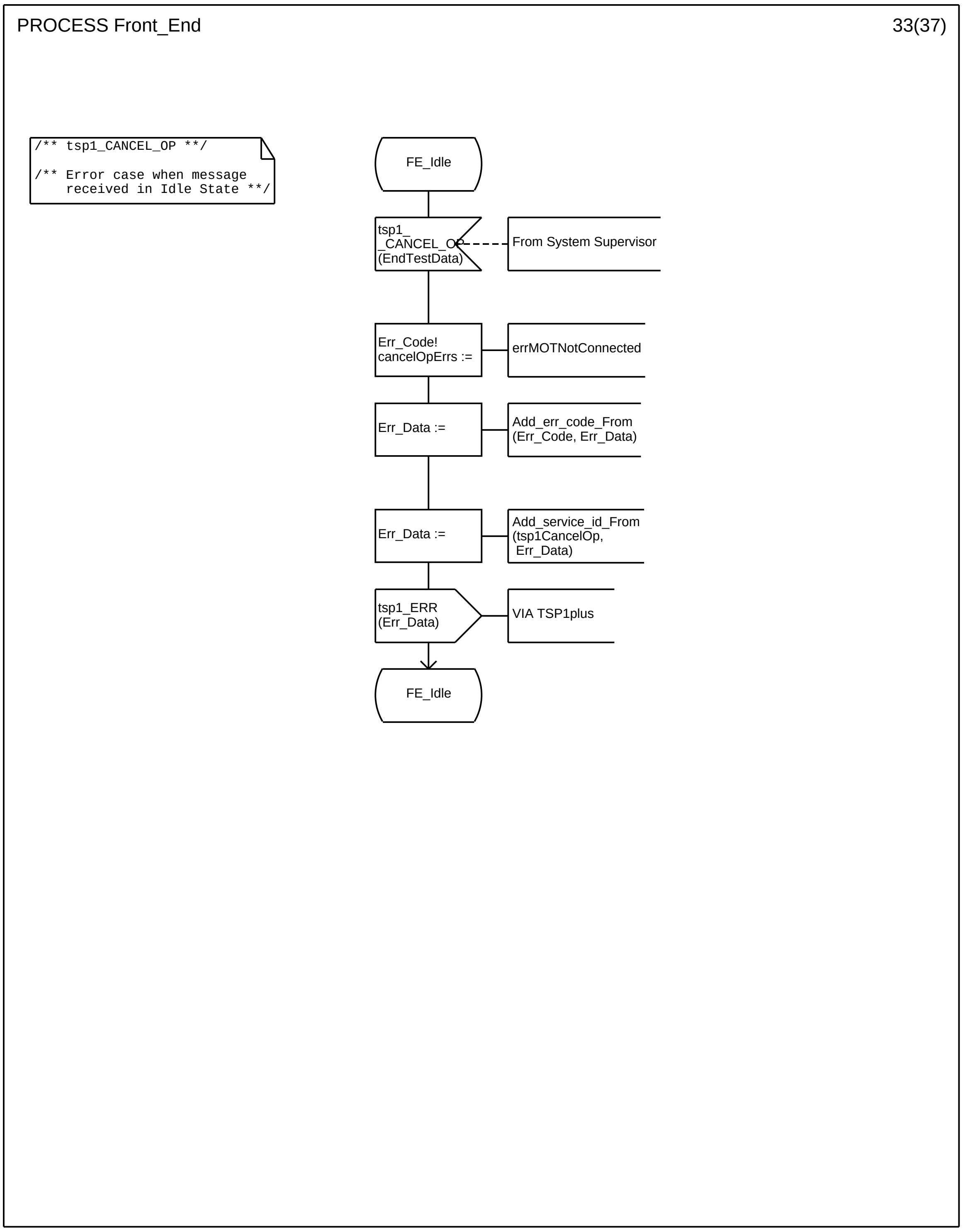

PROCESS Front_End
28(37)
/** tsp1_ASK_TRACE **/
/** In the 'Testers Suspended' State **/
Testers_ Suspended
tsp1_ASK_ _TRACE (TraceRequest)
From System Supervisor
Start_Timer (T_FE_1)
request_ _trace
VIA TestInterface
/* To the Configuration Element indicated in TraceRequest */
Testers_ Suspended

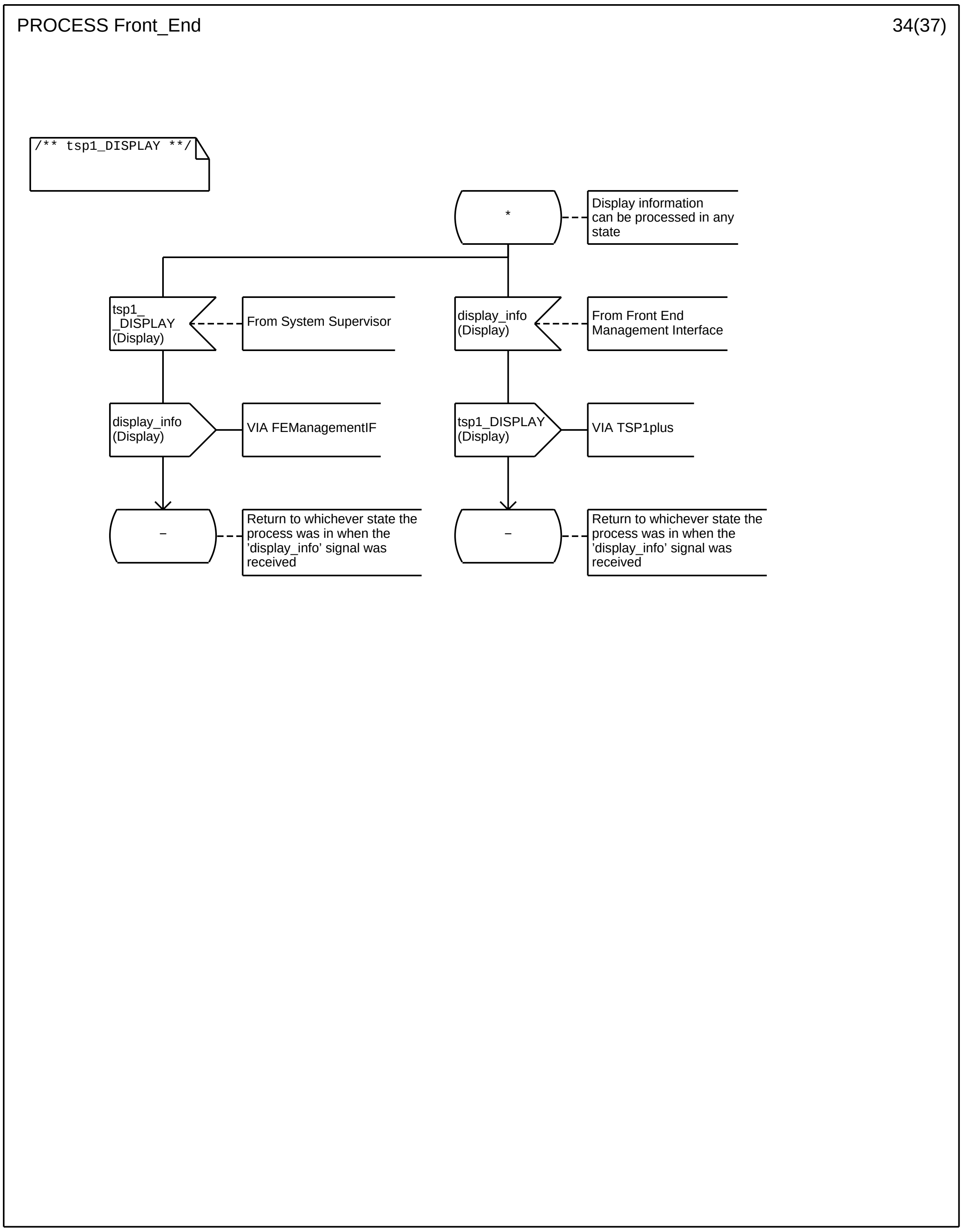

PROCESS Front_End
29(37)
/** tsp1_ASK_TRACE_ACK **/
/** In the 'Testers Suspended' State **/
Testers_ Suspended
T_FE_1 /* Timeout */
request_ _trace_ack (TraceData)
From Tester
Stop_Timer (T_FE_1)
CheckFor_ ValidTrace
(TraceData, TraceValidity)
Trace_ Validity
Invalid
Valid
Err_Data :=
Add_service_id_From (tsp1AskTrace, Err_Data)
Err_Code! askTraceErrs :=
errMOTNotReady
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
Testers_ Suspended
Err_Code! askTraceErrs :=
errTraceNotAvailable
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
Err_Data :=
Add_service_id_From (tsp1AskTrace, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
Testers_ Suspended
tsp1_ASK_ _TRACE_ACK (TraceData)
VIA TSP1plus
Testers_ Suspended

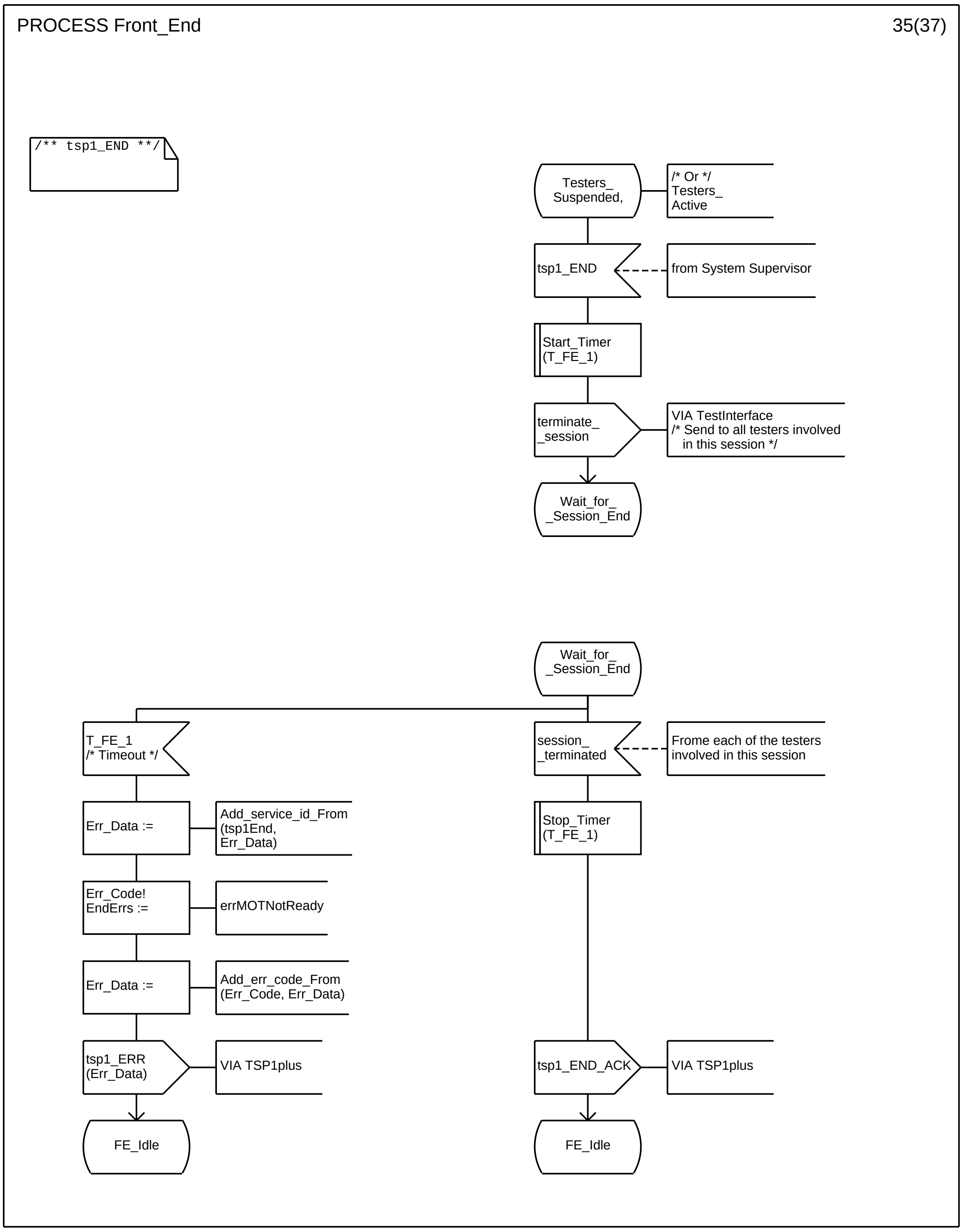

PROCESS Front_End
30(37)
/** tsp1_ASK_TRACE **/
/** Error case when message received in Idle State **/
FE_Idle
tsp1_ASK_ _TRACE (TraceRequest)
From System Supervisor
Err_Code! askTraceErrs :=
errMOTNotConnected
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
Err_Data :=
Add_service_id_From (tsp1AskTrace, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
FE_Idle

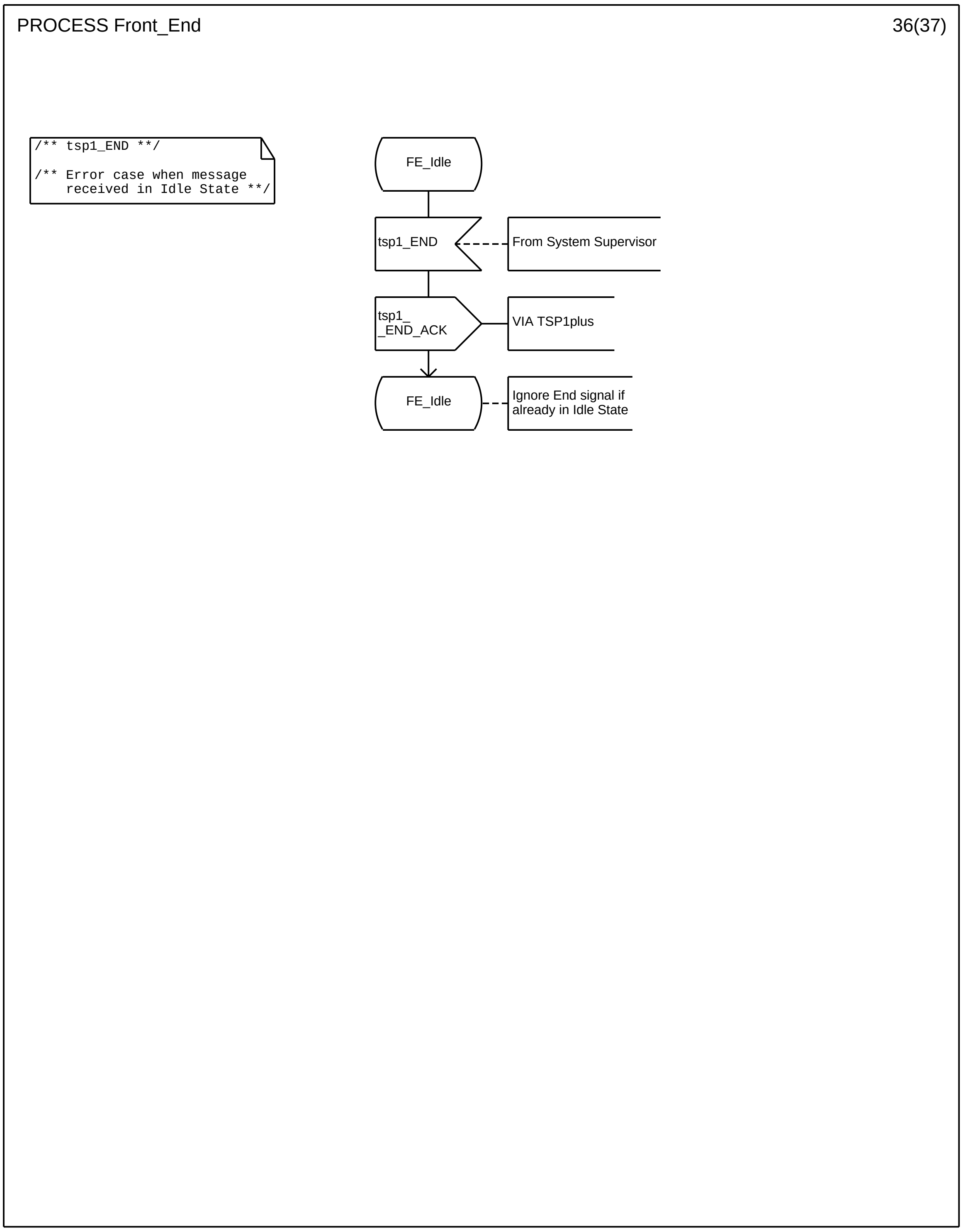

PROCESS Front_End
31(37)
/** tsp1_CANCEL_OP **/
Testers_ Active
From System Supervisor
tsp1_ _CANCEL_ _OP
(EndTestData)
Tester :=
conf_elem_id_From (EndTestData)
Start_Timer (T_FE_1)
terminate_ _test
VIA TestInterface
/* To the tester identified in 'Tester' or to all testers if 'Tester' is empty */
Wait_for_Test_ _Terminated_2

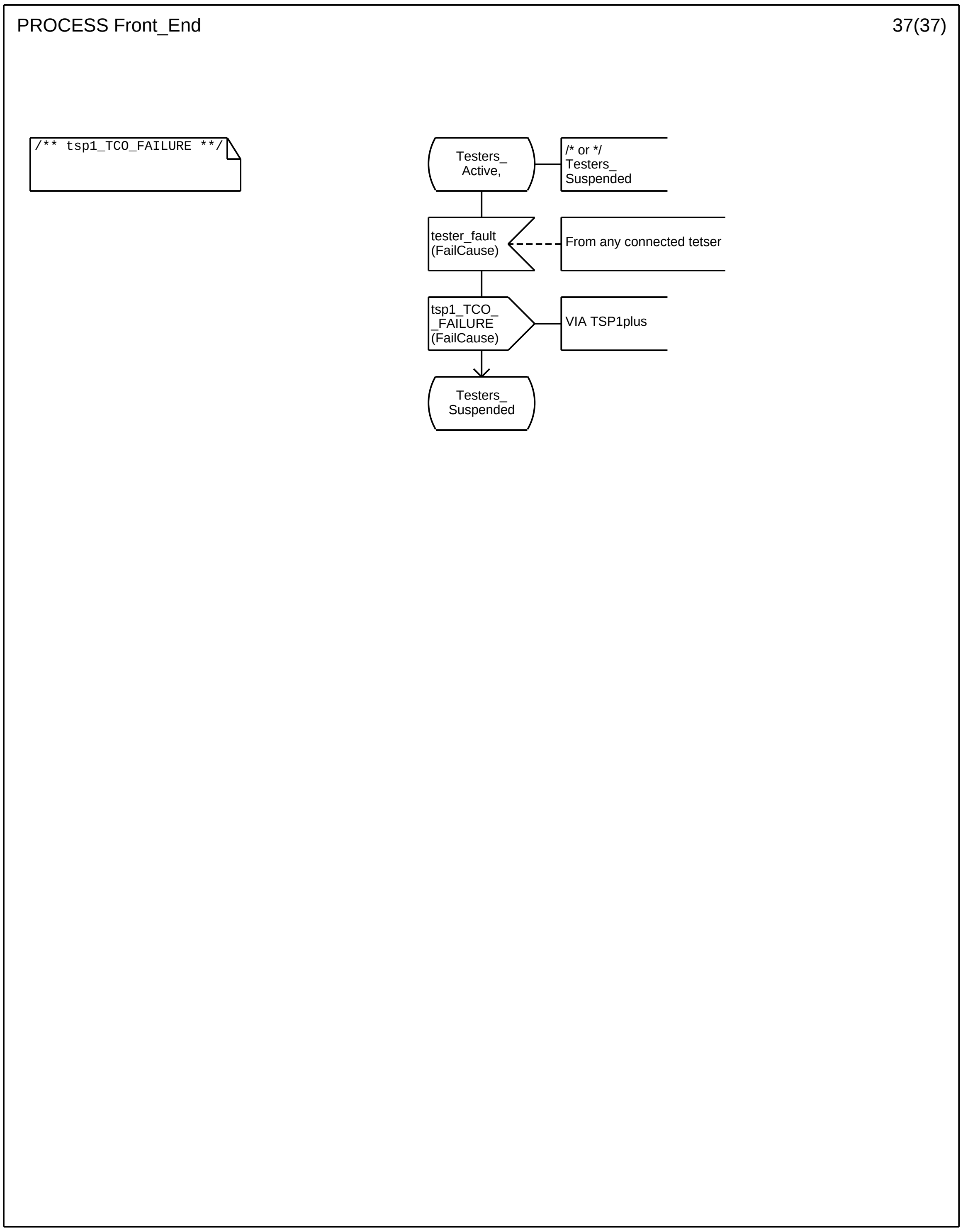

PROCESS Front_End
32(37)
/** tsp1_CANCEL_OP_ACK **/
Wait_for_Test_ _Terminated_2
T_FE_1 /* Timeout */
test_ _terminated
From all terminating testers
Err_Data :=
Add_service_id_From (tsp1CancelOp, Err_Data)
Stop_Timer (T_FE_1)
Err_Code! cancelOpErrs :=
errMOTNotReady
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
EndTestAck :=
Add_conf_elem_id_ack_From (Tester, EndTestAck)
tsp1_ERR (Err_Data)
VIA TSP1plus
tsp1_ _CANCEL_OP_ _ACK
(EndTestAck) VIA TSP1plus
Testers_ Active
Testers_ Suspended

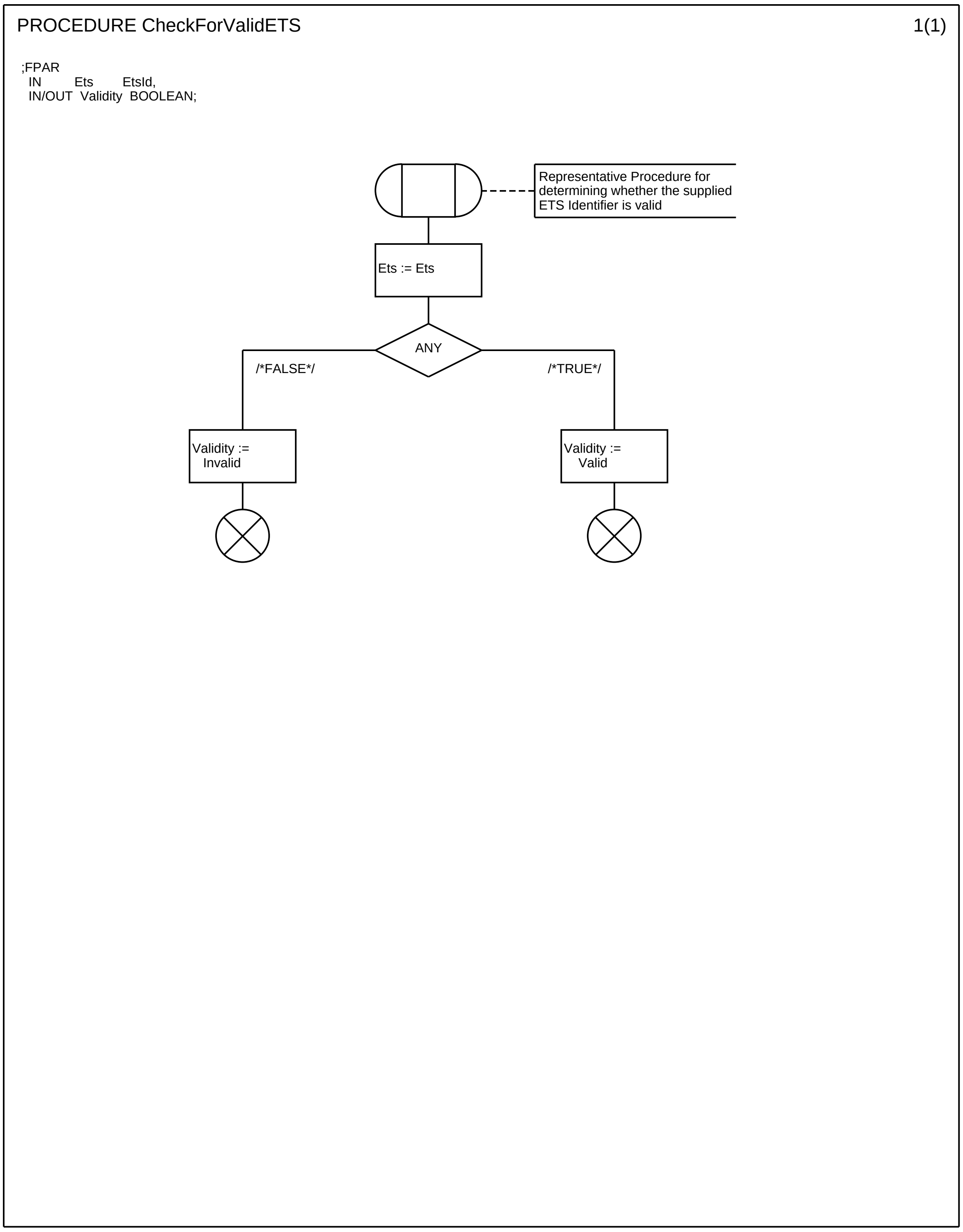

PROCESS Front_End
33(37)
/** tsp1_CANCEL_OP **/
/** Error case when message received in Idle State **/
FE_Idle
tsp1_ _CANCEL_OP (EndTestData)
From System Supervisor
Err_Code! cancelOpErrs :=
errMOTNotConnected
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
Err_Data :=
Add_service_id_From (tsp1CancelOp, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
FE_Idle

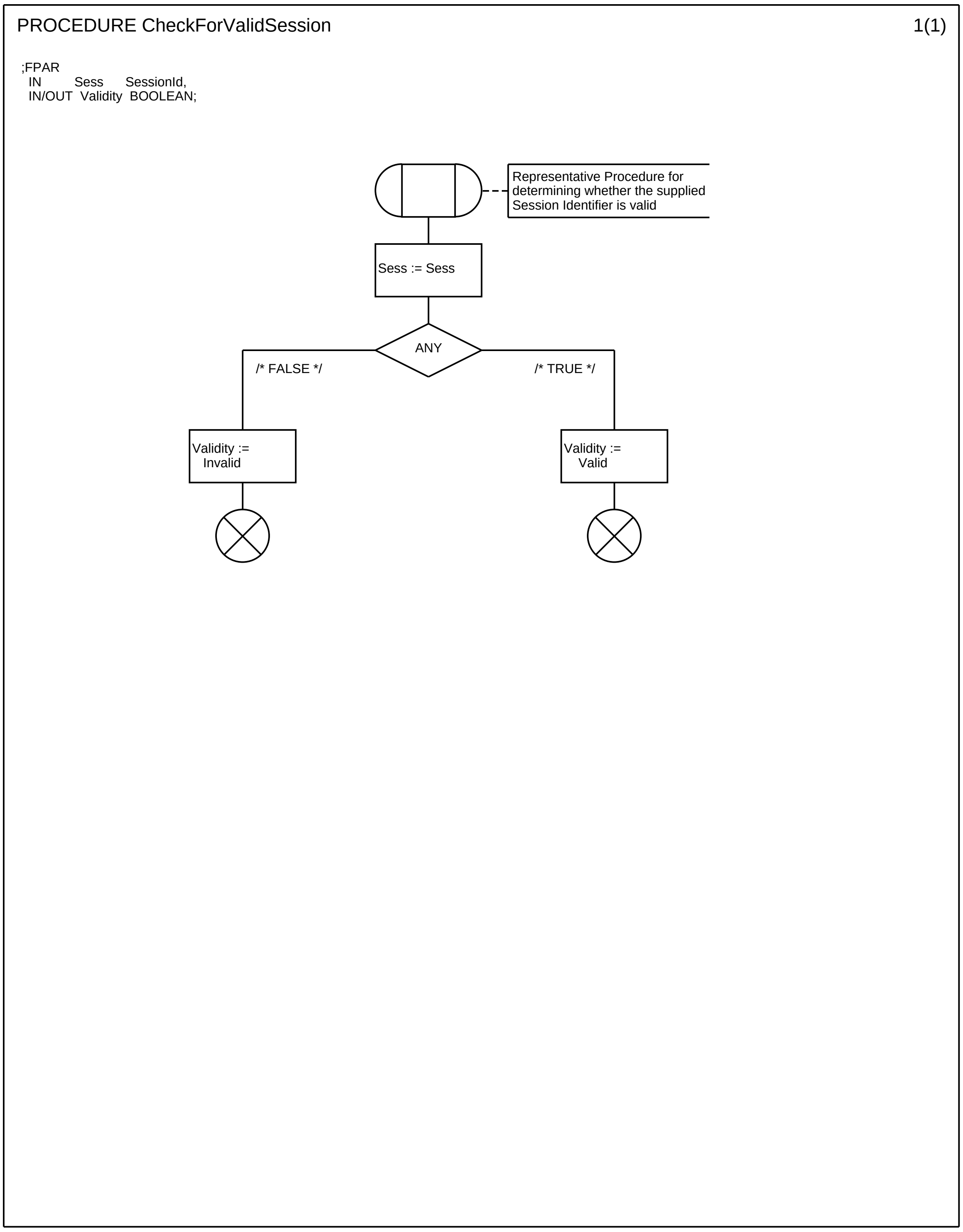

PROCESS Front_End
34(37)
/** tsp1_DISPLAY **/
*
Display information can be processed in any state
tsp1_ _DISPLAY (Display)
From System Supervisor
display_info (Display)
From Front End Management Interface
display_info (Display)
VIA FEManagementIF
tsp1_DISPLAY (Display)
VIA TSP1plus
–
Return to whichever state the process was in when the 'display_info' signal was received
–
Return to whichever state the process was in when the 'display_info' signal was received

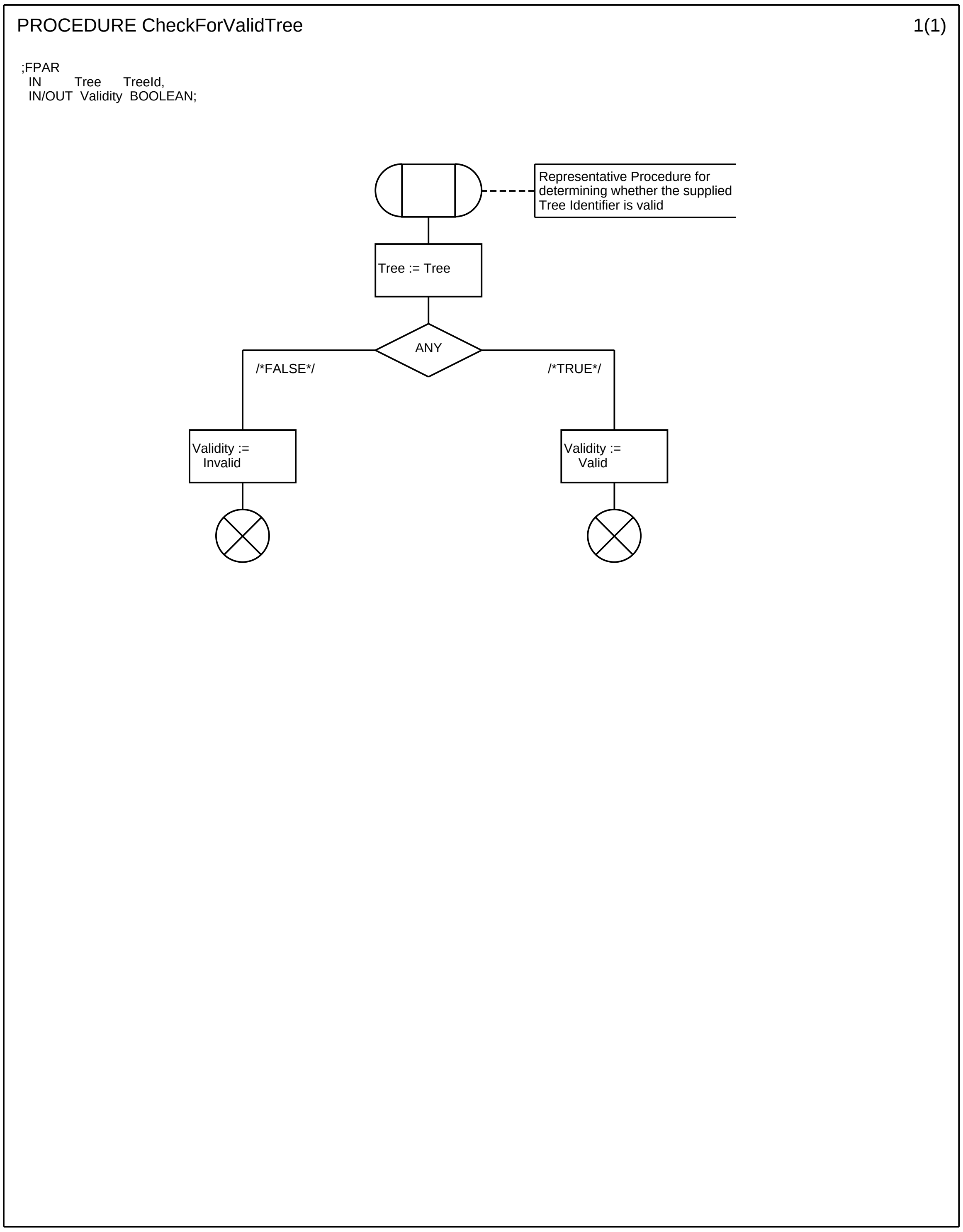

PROCESS Front_End
35(37)
/** tsp1_END **/
Testers_ Suspended,
/* Or */ Testers_ Active
tsp1_END
from System Supervisor
Start_Timer (T_FE_1)
terminate_ _session
VIA TestInterface /* Send to all testers involved in this session */
Wait_for_ _Session_End
Wait_for_ _Session_End
T_FE_1 /* Timeout */
Err_Data :=
Add_service_id_From (tsp1End, Err_Data)
Err_Code! EndErrs :=
errMOTNotReady
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
FE_Idle
session_ _terminated
Frome each of the testers involved in this session
Stop_Timer (T_FE_1)
tsp1_END_ACK
VIA TSP1plus
FE_Idle

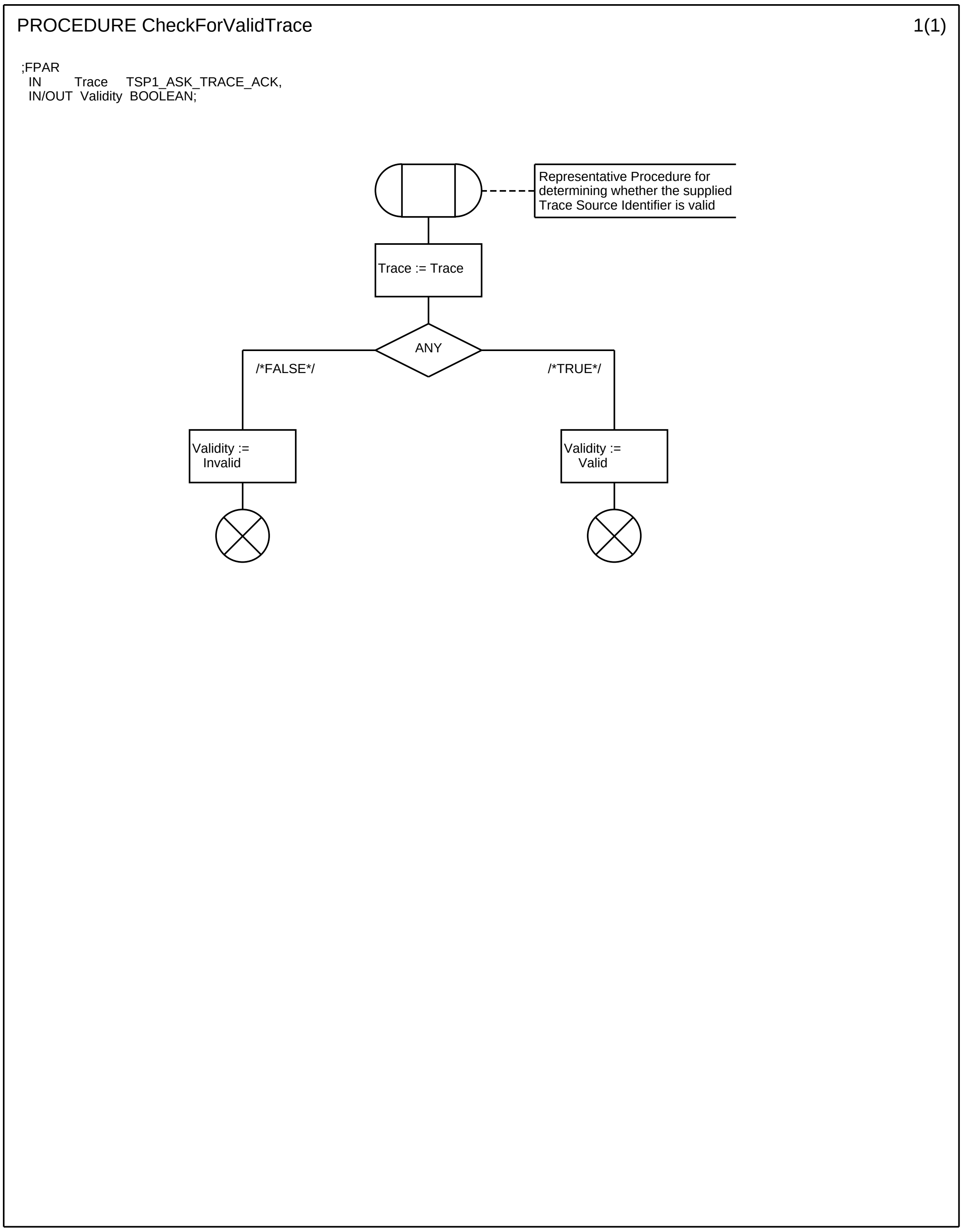

PROCESS Front_End
36(37)
/** tsp1_END **/
/** Error case when message
received in Idle State **/
FE_Idle
tsp1_END
From System Supervisor
tsp1_
_END_ACK
VIA TSP1plus
FE_Idle
Ignore End signal if
already in Idle State

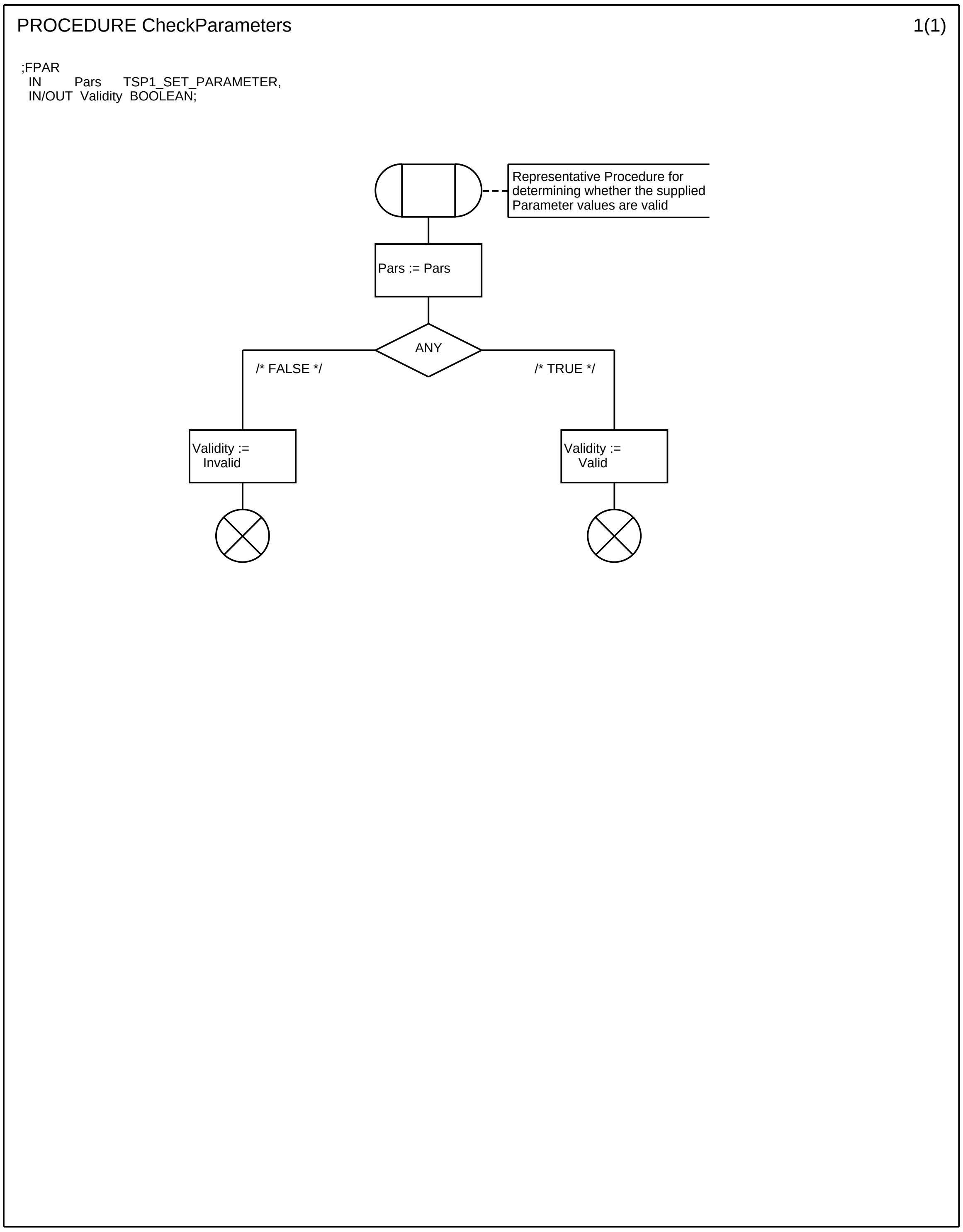

PROCESS Front_End
37(37)
/** tsp1_TCO_FAILURE **/
Testers_ Active,
/* or */ Testers_ Suspended
tester_fault (FailCause)
From any connected tetser
tsp1_TCO_ _FAILURE (FailCause)
VIA TSP1plus
Testers_ Suspended

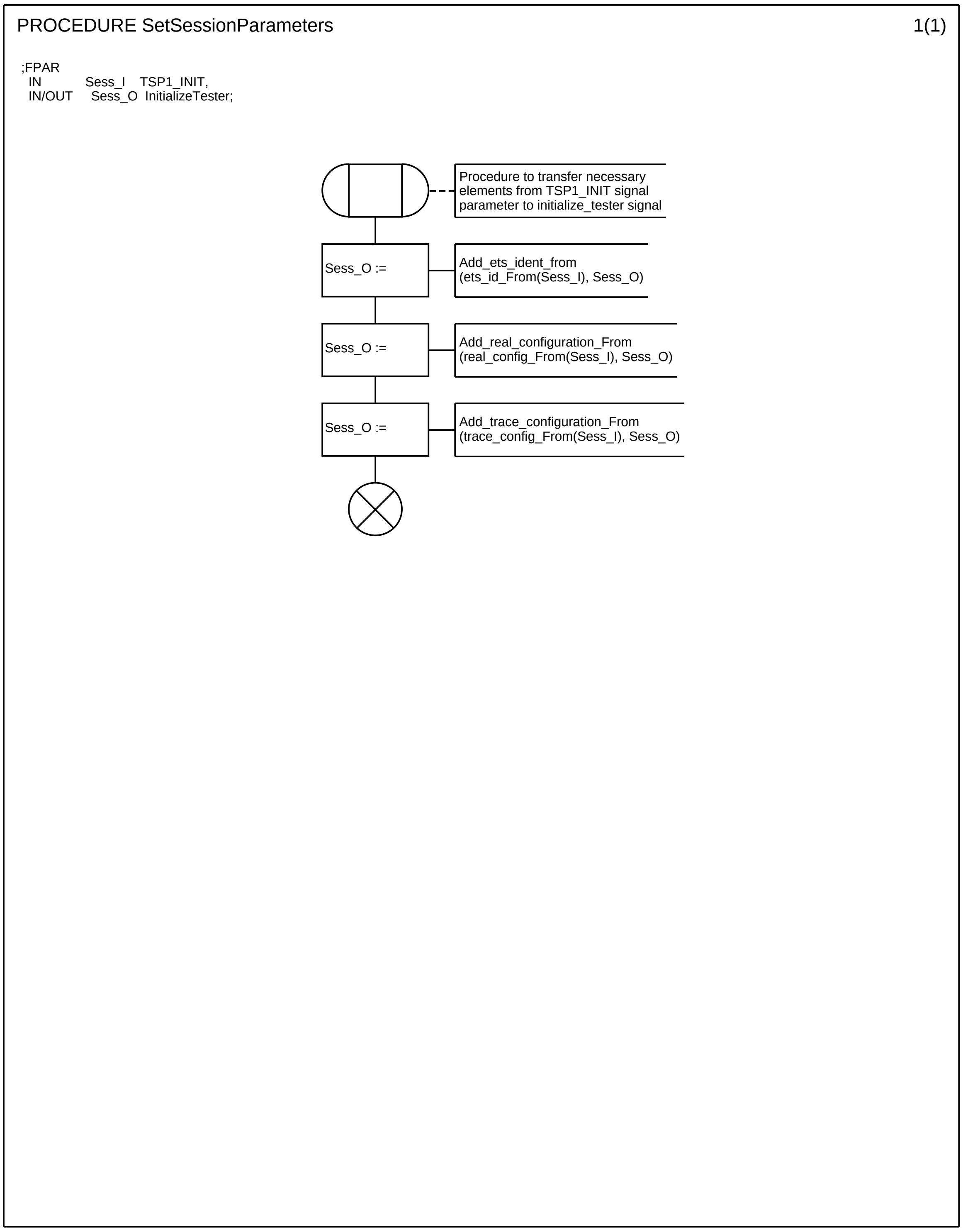

PROCEDURE CheckForValidETS
1(1)
;FPAR
IN Ets EtsId,
IN/OUT Validity BOOLEAN;
Representative Procedure for determining whether the supplied ETS Identifier is valid
Ets := Ets
ANY
/*FALSE*/
/*TRUE*/
Validity := Invalid
Validity := Valid

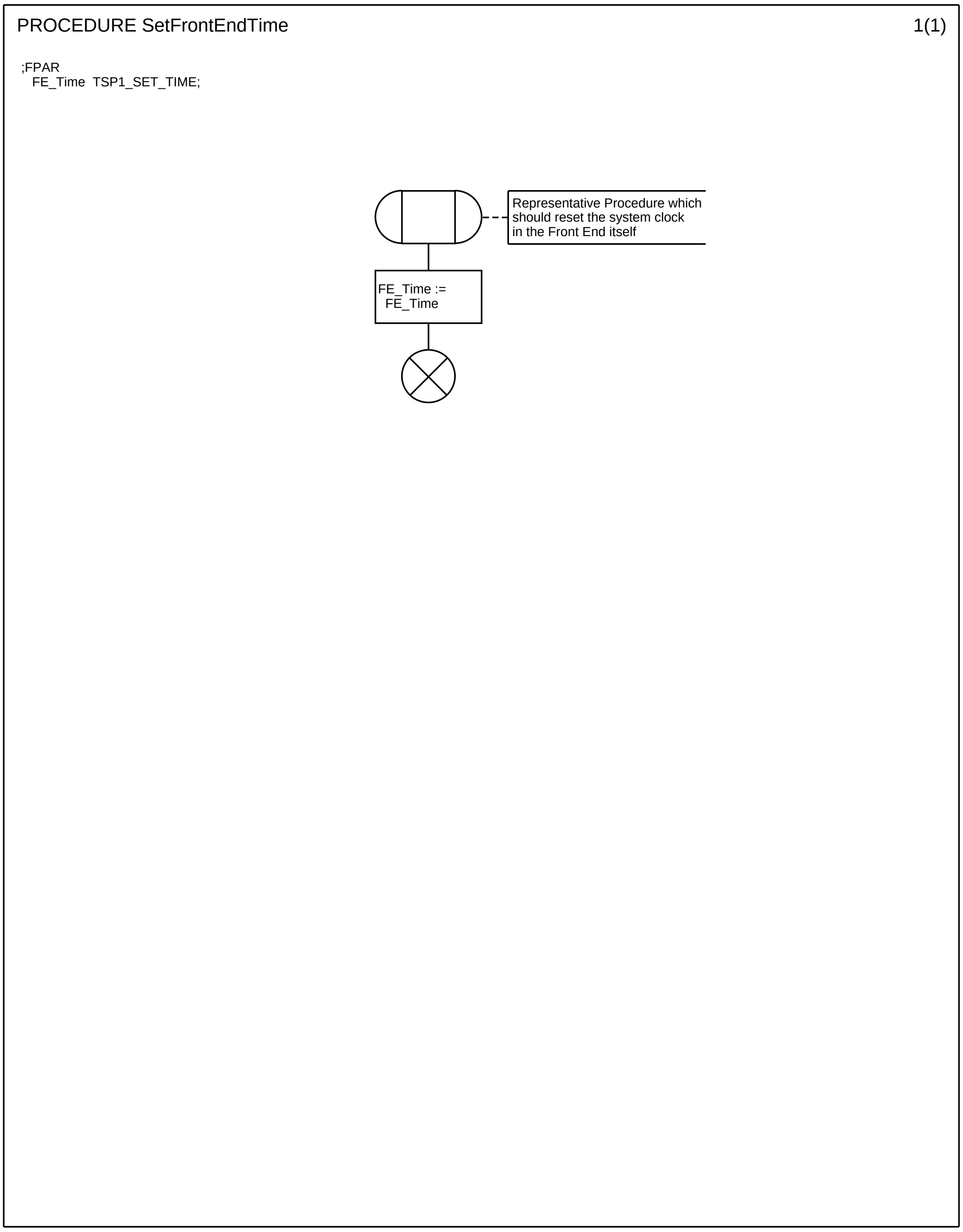

PROCEDURE CheckForValidSession
1(1)
;FPAR
IN Sess SessionId,
IN/OUT Validity BOOLEAN;
Representative Procedure for determining whether the supplied Session Identifier is valid
Sess := Sess
ANY
/* FALSE */
/* TRUE */
Validity := Invalid
Validity := Valid

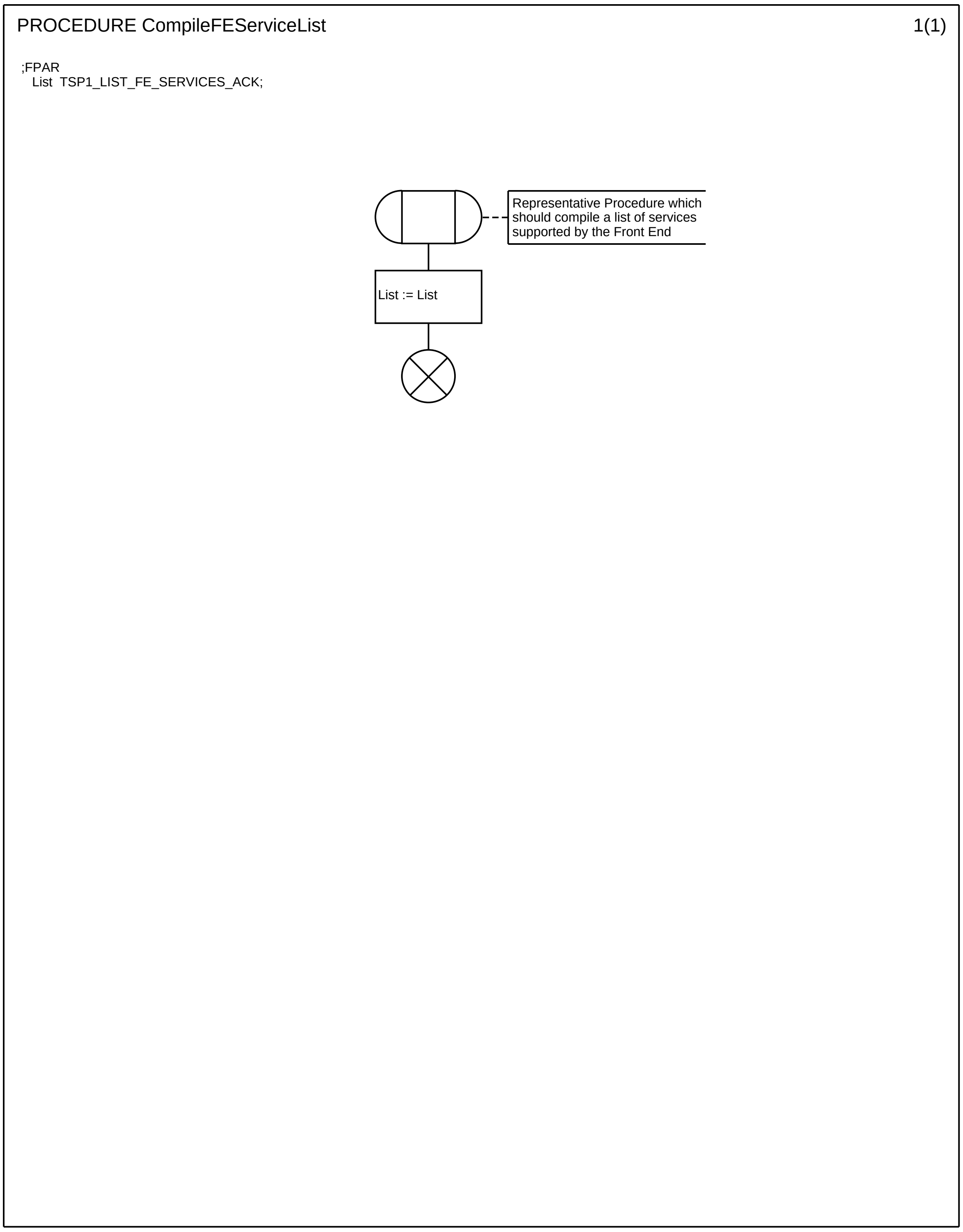

PROCEDURE CheckForValidTree
1(1)
;FPAR
IN Tree TreeId,
IN/OUT Validity BOOLEAN;
Representative Procedure for determining whether the supplied Tree Identifier is valid
Tree := Tree
ANY
/*FALSE*/
/*TRUE*/
Validity := Invalid
Validity := Valid

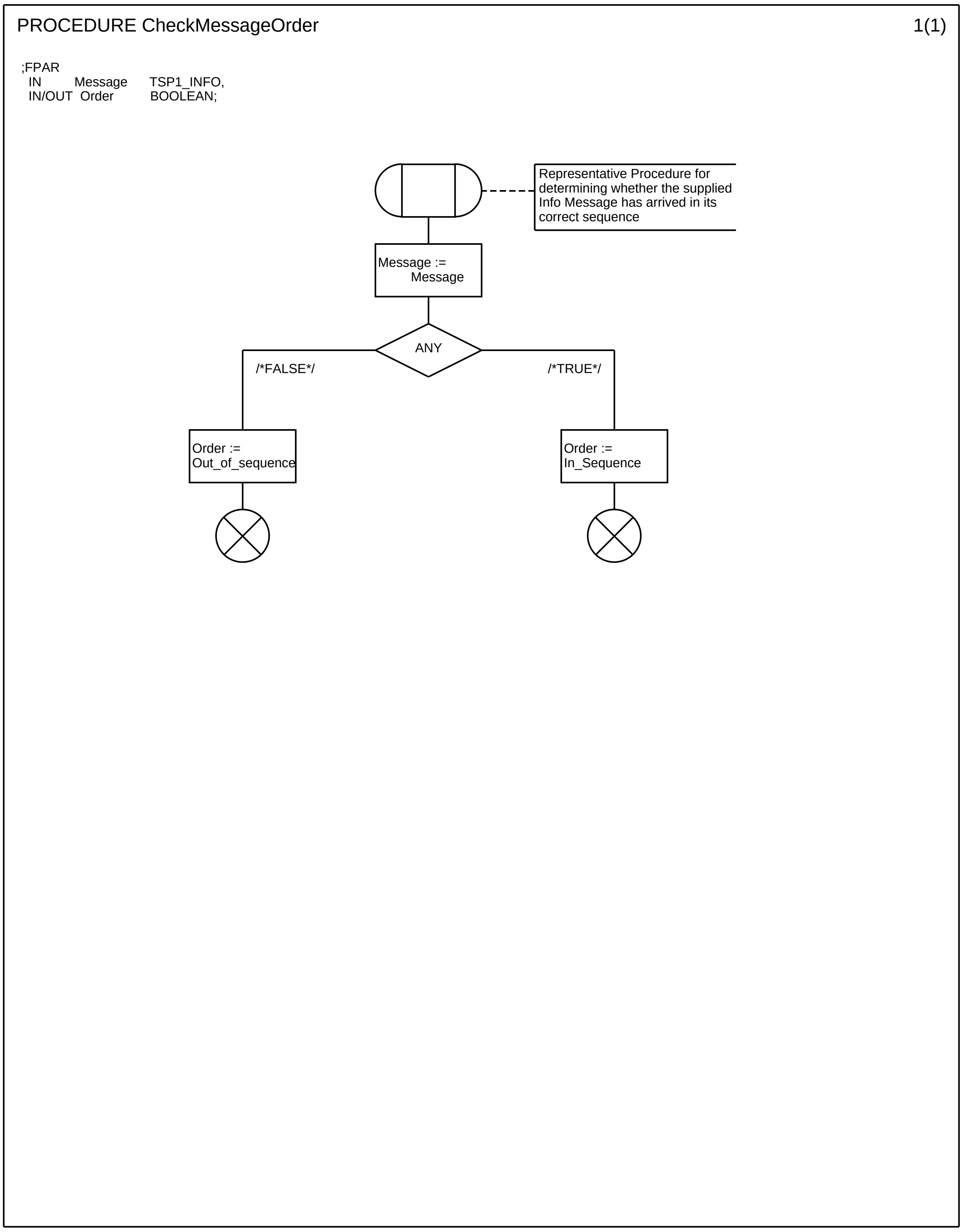

PROCEDURE CheckForValidTrace
1(1)
;FPAR
IN Trace TSP1_ASK_TRACE_ACK,
IN/OUT Validity BOOLEAN;
Representative Procedure for determining whether the supplied Trace Source Identifier is valid
Trace := Trace
ANY
/*FALSE*/
/*TRUE*/
Validity := Invalid
Validity := Valid

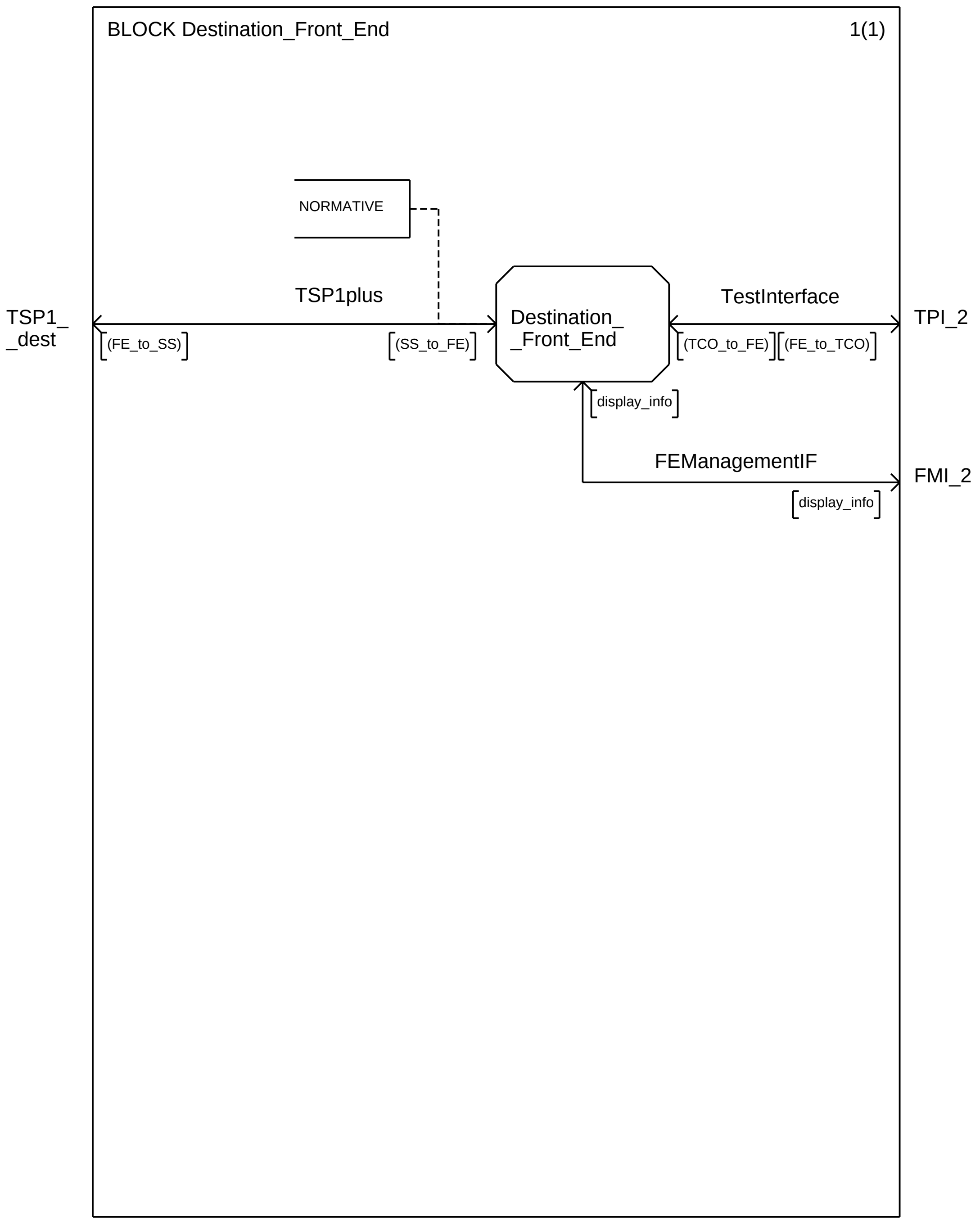

PROCEDURE CheckParameters
1(1)
;FPAR
IN Pars TSP1_SET_PARAMETER,
IN/OUT Validity BOOLEAN;
Representative Procedure for determining whether the supplied Parameter values are valid
Pars := Pars
ANY
/* FALSE */
/* TRUE */
Validity := Invalid
Validity := Valid

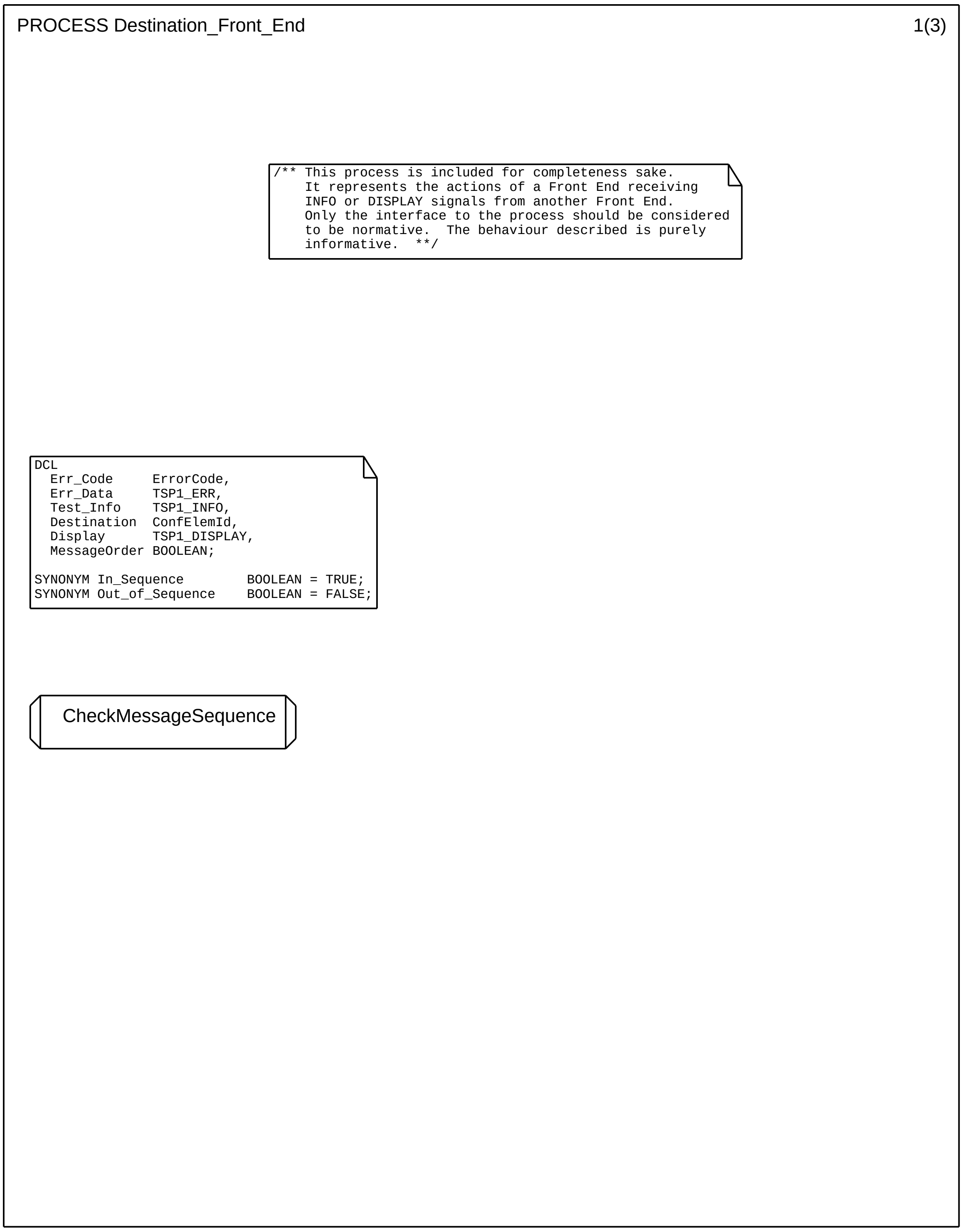

PROCEDURE SetSessionParameters
1(1)
;FPAR
IN Sess_I TSP1_INIT,
IN/OUT Sess_O InitializeTester;
Procedure to transfer necessary elements from TSP1_INIT signal parameter to initialize_tester signal
Sess_O :=
Add_ets_ident_from (ets_id_From(Sess_I), Sess_O)
Sess_O :=
Add_real_configuration_From (real_config_From(Sess_I), Sess_O)
Sess_O :=
Add_trace_configuration_From (trace_config_From(Sess_I), Sess_O)


PROCEDURE SetFrontEndTime
1(1)
;FPAR
FE_Time TSP1_SET_TIME;
Representative Procedure which should reset the system clock in the Front End itself
FE_Time := FE_Time


PROCEDURE CompileFEServiceList
1(1)
;FPAR
List TSP1_LIST_FE_SERVICES_ACK;
Representative Procedure which should compile a list of services supported by the Front End
List := List

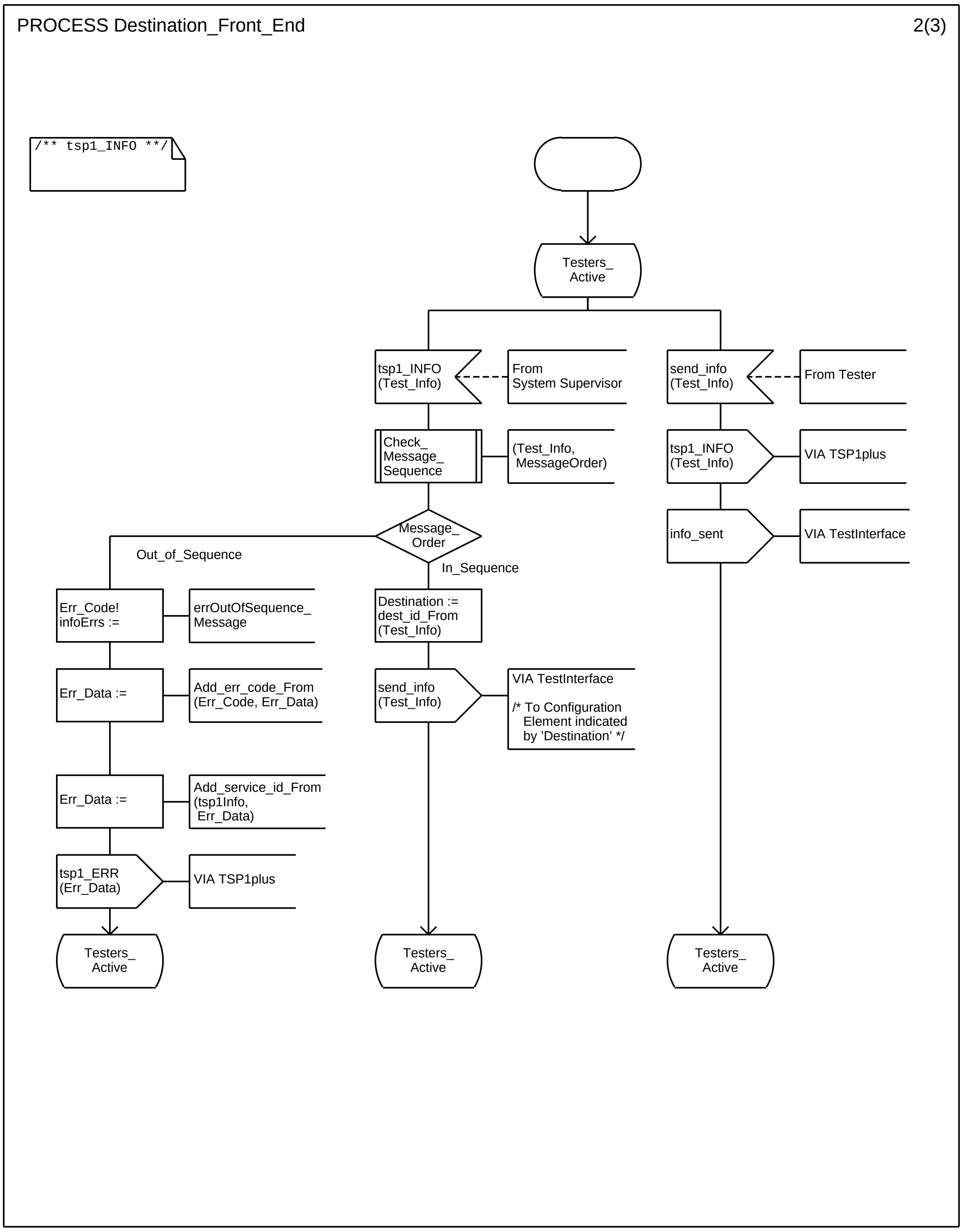

PROCEDURE CheckMessageOrder
1(1)
;FPAR
IN Message TSP1_INFO,
IN/OUT Order BOOLEAN;
Representative Procedure for determining whether the supplied Info Message has arrived in its correct sequence
Message := Message
ANY
/*FALSE*/
/*TRUE*/
Order := Out_of_sequence
Order := In_Sequence

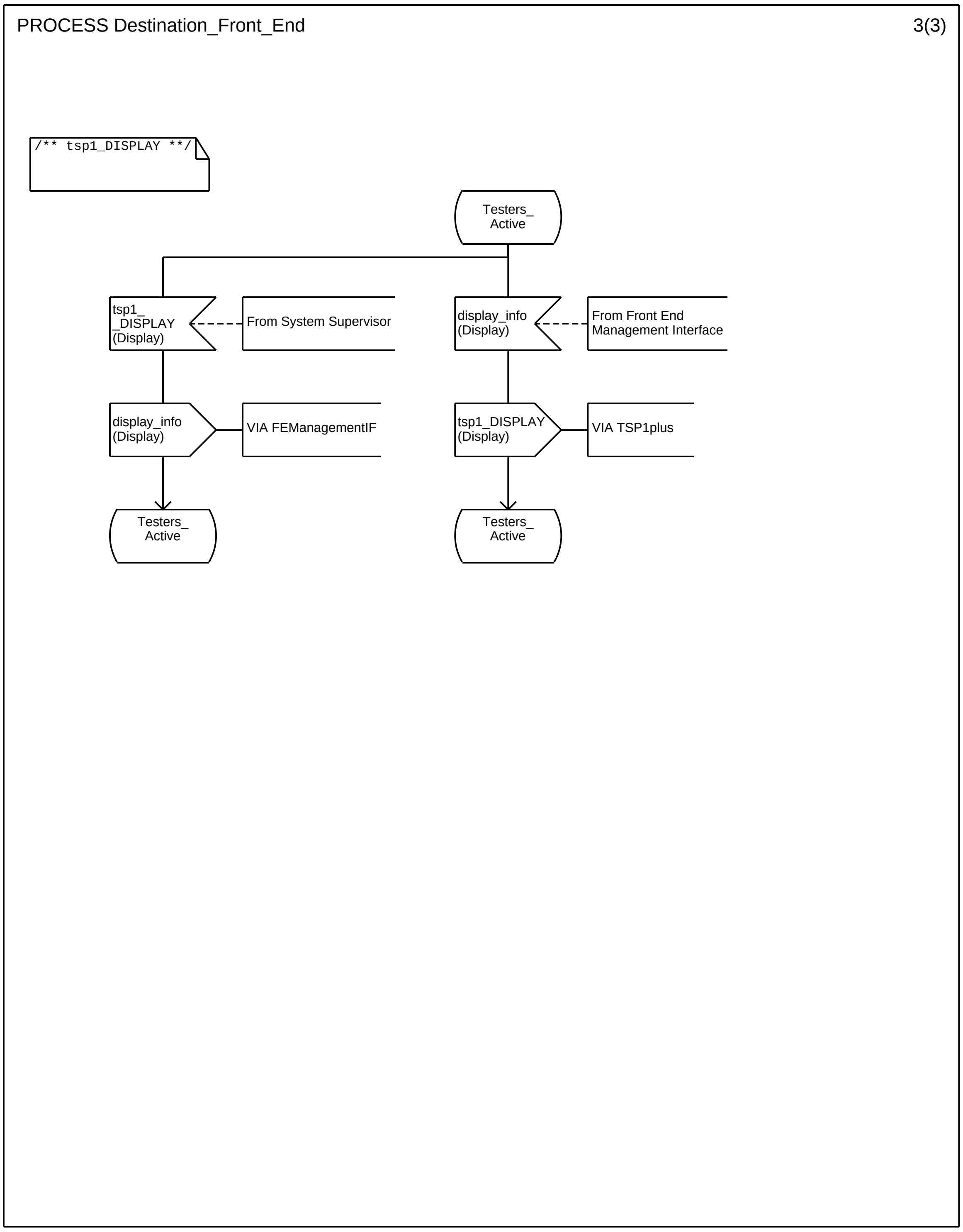

BLOCK Destination_Front_End
1(1)
NORMATIVE
TSP1plus
TSP1_
_dest
(FE_to_SS)
(SS_to_FE)
Destination_
_Front_End
TestInterface
TPI_2
(TCO_to_FE)
(FE_to_TCO)
display_info
FEManagementIF
FMI_2
display_info


PROCESS Destination_Front_End 1(3)

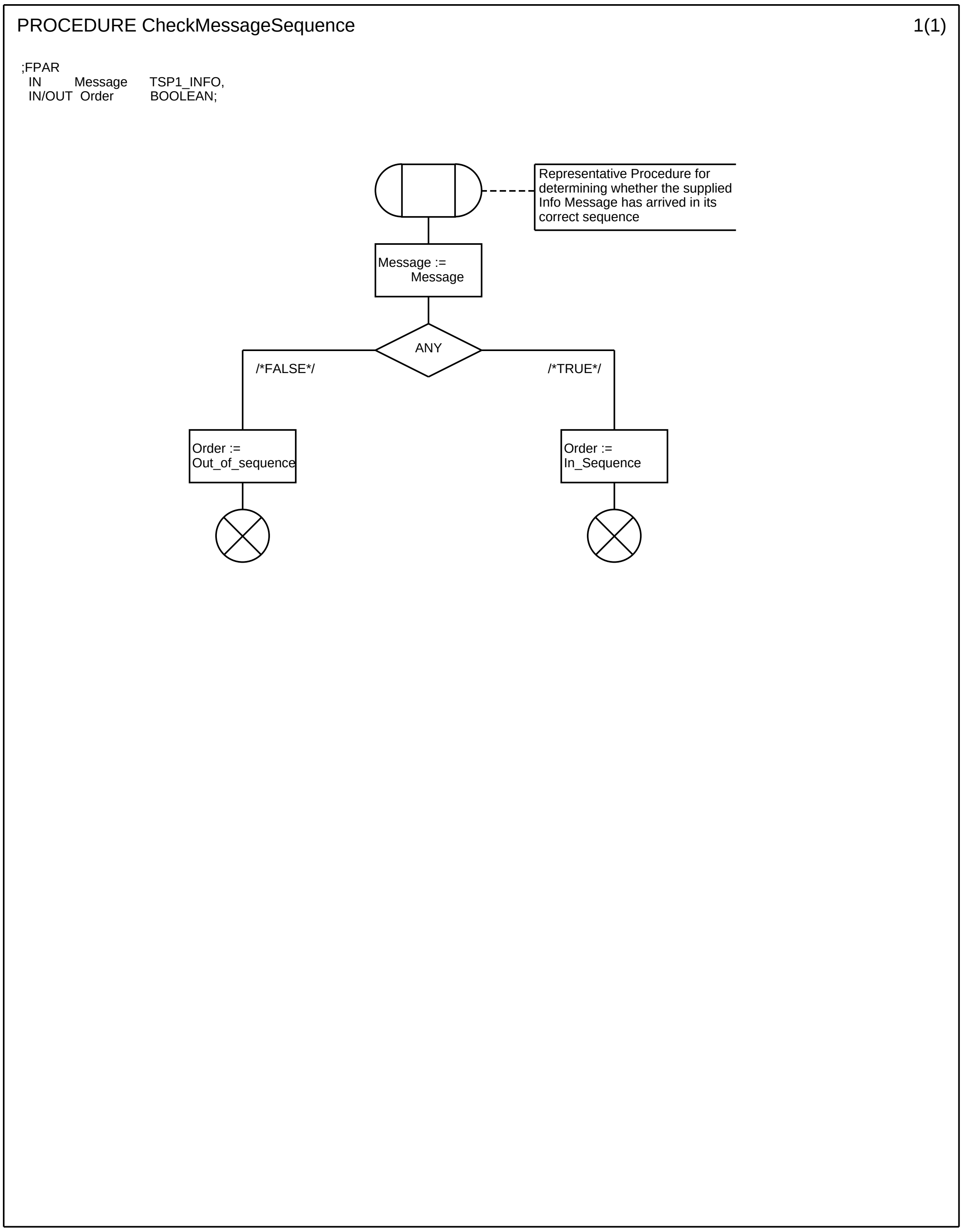
```
/** This process is included for completeness sake.
    It represents the actions of a Front End receiving
    INFO or DISPLAY signals from another Front End.
    Only the interface to the process should be considered
    to be normative.  The behaviour described is purely
    informative.  **/
```

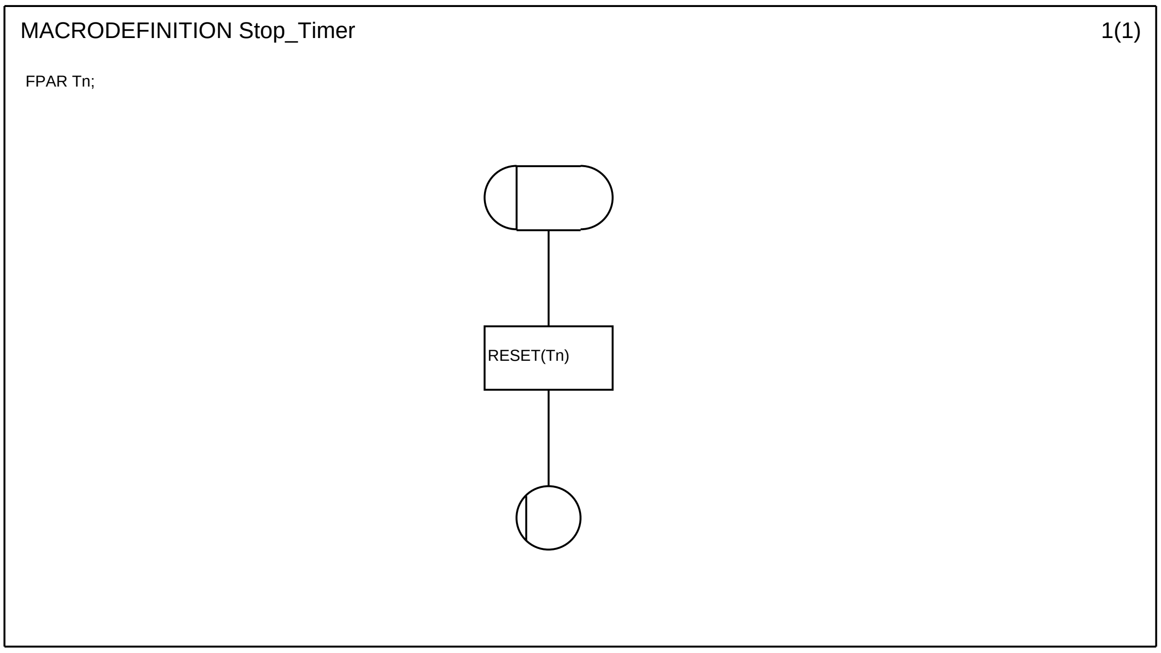
```
DCL
  Err_Code     ErrorCode,
  Err_Data     TSP1_ERR,
  Test_Info    TSP1_INFO,
  Destination  ConfElemId,
  Display      TSP1_DISPLAY,
  MessageOrder BOOLEAN;

SYNONYM In_Sequence        BOOLEAN = TRUE;
SYNONYM Out_of_Sequence    BOOLEAN = FALSE;
```

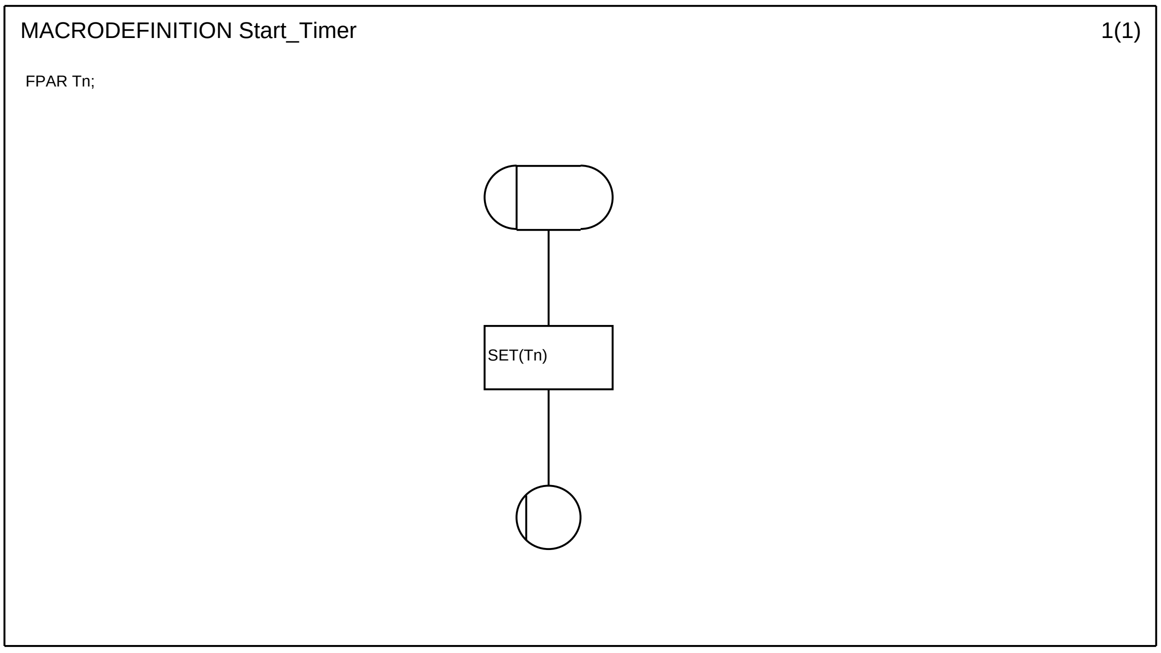

PROCESS Destination_Front_End
2(3)
/** tsp1_INFO **/
Testers_ Active
tsp1_INFO (Test_Info)
From System Supervisor
send_info (Test_Info)
From Tester
Check_ Message_ Sequence
(Test_Info, MessageOrder)
tsp1_INFO (Test_Info)
VIA TSP1plus
Message_ Order
info_sent
VIA TestInterface
Out_of_Sequence
In_Sequence
Err_Code! infoErrs :=
errOutOfSequence_ Message
Destination := dest_id_From (Test_Info)
Err_Data :=
Add_err_code_From (Err_Code, Err_Data)
send_info (Test_Info)
VIA TestInterface
/* To Configuration Element indicated by 'Destination' */
Err_Data :=
Add_service_id_From (tsp1Info, Err_Data)
tsp1_ERR (Err_Data)
VIA TSP1plus
Testers_ Active
Testers_ Active
Testers_ Active



PROCESS Destination_Front_End
3(3)
/** tsp1_DISPLAY **/
Testers_ Active
tsp1_ _DISPLAY (Display)
From System Supervisor
display_info (Display)
From Front End Management Interface
display_info (Display)
VIA FEManagementIF
tsp1_DISPLAY (Display)
VIA TSP1plus
Testers_ Active
Testers_ Active


PROCEDURE CheckMessageSequence
1(1)
;FPAR
IN Message TSP1_INFO,
IN/OUT Order BOOLEAN;
Representative Procedure for determining whether the supplied Info Message has arrived in its correct sequence
Message := Message
ANY
/*FALSE*/
/*TRUE*/
Order := Out_of_sequence
Order := In_Sequence


MACRODEFINITION Stop_Timer
1(1)
FPAR Tn;
RESET(Tn)


MACRODEFINITION Start_Timer
1(1)
FPAR Tn;
SET(Tn)

Annex E (normative): Additional ASN.1 operations for inter-layer communication

For the purposes of communication between layers of the TSP1+ protocol stack, the following ASN.1 operations shall be used for tagging TSP1+ PDUs (i.e., PDU bodies).

```
  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,

                     tsp1-tco-failure                 [26] IMPLICIT TSP1-TCO-FAILURE }

-- The following operations are included to complete the list of TSP1 PDUs  --
-- by providing NULL definitions for those signals which have no parameters –

  TSP1-INIT-ACK          ::= NULL
  TSP1-INIT-COMPLETE     ::= NULL
  TSP1-CHK-CONF          ::= NULL
  TSP1-CHK-CONF-ACK      ::= NULL
  TSP1-SET-PARAMETER-ACK ::= NULL
  TSP1-SET-TIME-ACK      ::= NULL
  TSP1-LIST-FE-SERVICES  ::= NULL
  TSP1-END               ::= NULL
  TSP1-END-ACK           ::= NULL
```

Bibliography

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

- ETSI ETR 141 (1994): "Methods for Testing and Specification (MTS); Protocol and profile conformance testing specifications The Tree and Tabular Combined Notation (TTCN) style guide".
- ETSI ETR 193 (1995): "Methods for Testing and Specification (MTS); Network Integration Testing (NIT); Methodology aspects; Test Co-ordination Procedure (TCP) style guide".
- ETSI ETR 303 (1997): "Methods for Testing and Specification (MTS); Test Synchronization; Architectural reference; Test Synchronization Protocol 1 (TSP1) specification".
- ETSI TR 101 667 (1999): "Methods for Testing and Specification (MTS); Network Integration Testing (NIT); Interconnection; 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		
V4.2.4	May 2000	Membership Approval Procedure MV 20000714: 2000-05-16 to 2000-07-14
V4.2.4	September 2000	Publication