ETSI
TECHNICAL
REPORT

ETR 303

January 1997

Source: ETSI TC-MTS

Reference: DTR/MTS-00028

ICS: 33.020

Key words: NIT, testing, TSP1

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

ETSI

European Telecommunications Standards Institute

ETSI Secretariat

Postal address: F-06921 Sophia Antipolis CEDEX - FRANCE
Office address: 650 Route des Lucioles - Sophia Antipolis - Valbonne - FRANCE
X.400: c=fr, a=atlas, p=etsi, s=secretariat - **Internet:** secretariat@etsi.fr

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

Contents

Foreword

This ETSI Technical Report (ETR) has been produced by the Methods for Testing and Specification (MTS) Technical Committee of the European Telecommunications Standards Institute (ETSI) based on work conducted under Eurescom project P.412.

ETRs are informative documents resulting from ETSI studies which are not appropriate for European Telecommunication Standard (ETS) or Interim European Telecommunication Standard (I-ETS) status. An ETR may be used to publish material which is either of an informative nature, relating to the use or the application of ETSs or I-ETSs, or which is immature and not yet suitable for formal adoption as an ETS or an I-ETS.

Blank page

1 Scope

This ETR describes a Test Synchronization Architecture (TSA) and then, within this architecture, the specification of the "TSP1 protocol", which is the language spoken by two TSP1-compliant Test Synchronization Architectural Elements (TSAEs), is given.

The purpose of the TSP1 protocol is to achieve functional co-ordination and timing synchronization between two or more TSAEs involved in a testing session of a distributed nature (where a logical-functional and/or a physical-geographical distribution of different functional testing components takes place).

The TSP1 examples given in this deliverable are related to the application of the TSP1 protocol to the particular type of distributed testing called "Network Integration Testing" (NIT), which is currently used by telecom operators before opening new telecom services in an international (or multi-operator) environment. However the technical applicability of the TSP1 specification goes beyond such particular application.

2 References

For the purposes of this ETR, the following references apply:

2.1 ISO/ETSI References

[1]	ISO/IEC 9646 (1992): "OSI Conformance Testing Methodology and Framework, Part 1 to 5".
[2]	ISO/IEC 9646 - 1 (DAM) (1992): "Multi-Party Testing".
[3]	ETR 141 (1994): "Methods for Testing and Specification (MTS); Protocol and profile conformance testing specifications The Tree and Tabular Combined Notation (TTCN) style guide".
[4]	ISO/IEC 9646-3/AM1: "Concurrent TTCN".

2.2 NIT References

[5]	ETR 193: "Methods for Testing and Specification (MTS); Network Integration Testing (NIT); Methodology aspects; Test Co-ordination Procedure (TCP) style guide".
[6]	ETR 303: "Methods for Testing and Specification (MTS); Test Synchronization; Architectural reference; Test Synchronization Protocol 1 (TSP1) specification".

2.3 OMT References

[7]	Blaha M, Eddy F, Lorensen W, Premerlani W & Rumbaugh J: "Object Oriented Modeling and Design", Prentice-Hall International, Englewood Cliffs, NJ, 1991.

3 Definitions

For the purposes of this ETR, all the definitions in ISO/IEC 9646 [1],[2] and its amendments apply:

configurations: Test components as defined in test component configuration declarations of the C-TTCN.

session: All the information necessary to execute some tests belonging to a given configuration.

4 Symbols and abbreviations

For the purposes of this ETR, all the symbols and abbreviations defined in ISO/IEC 9646 [1] [2] and its amendments apply.

ASN.1	Abstract Syntax Notation One
ATS	Abstract Test Suite
BER	Basic Encoding Rules
CM	Co-ordination Message
CP	Co-ordination Point
C-TTCN	Concurrent TTCN
DSS1	Digital Subscriber Signalling system 1
FE	Front End
ISO	International Organization for Standardization
IUT	Implementation Under Test
LAN	Local Area Network
LT	Lower Tester
LTCF	Lower Tester Control Function
MPTM	Multi Party Testing Method
MTC	Main Test Component
NE	Network Element
NIT	Network Integration Testing
OMT	Object Modelling Technique
OSI	Open System Interconnection
PCO	Point of Control and Observation
PDU	Protocol Data Unit
PNO	Public Network Operator
PT	Protocol Tester
PTC	Parallel Test Component
SDL	Specification and Description Language
SS	System Supervisor
TSA	Test Synchronization Architecture
TSAE	Test Synchronization Architectural Element
TC	Test Component
TCP	Test Co-ordination Procedure
TCP/IP	Transmission Control Protocol / Internet Protocol
TMN	Telecommunication Management Network
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

5 Introduction to Network Integration Testing

When two networks - or two or more Network Elements (NEs) - capable of offering independently some telecommunication services are interconnected - in order to virtually build a wider, "global", network (or to build just a wider set of Network Elements) - it may be useful to be able to run a set of suitable tests in order to verify that the two networks (or the NEs in the wider set) correctly inter-operate and co-operate, with respect to the ability to provide "globally" the expected telecommunication services, at the external borders of the new environment.

This requires that the two networks (NEs) have been correctly "integrated", from the point of view of their *technical* ability to provide globally the expected services, so implementing a global network capable of offering telecom services to the users of both networks in a transparent and homogeneous way. This type of testing is a typical example of so called "Network Integration Testing" (NIT).

NIT can be done in practice, for example, by simulating the behaviour of two end users and checking that the global network behaves as expected (figure 1).

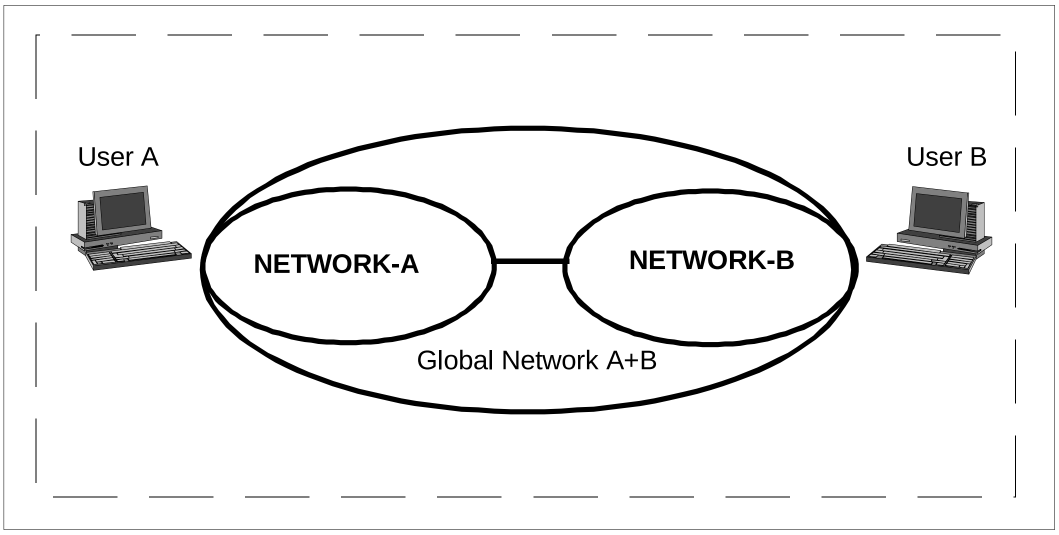


Figure 1: Global network functionality

Such a test campaign would be geographically distributed, because it will be necessary to simulate the behaviour of (for example) an end user (customer) in country X establishing a connection with another end user in country Y. If one assumes that one has at least one tester at each side (A and B), one needs to synchronize each of them when such a distributed test will be running.

For example, one tester - simulating an end user in country X - will start waiting for the test call to be originated by the other side (or country), according to the provisions of an "ad hoc", NIT test specification.

Such synchronization could be carried out in different ways:

- manually, when test operators would phone each other asking when a test case is going to start. This system - currently used by many telecom operators - has some drawbacks:

  - both test operators must speak the same language;

  - both test operators must be available at the same time.

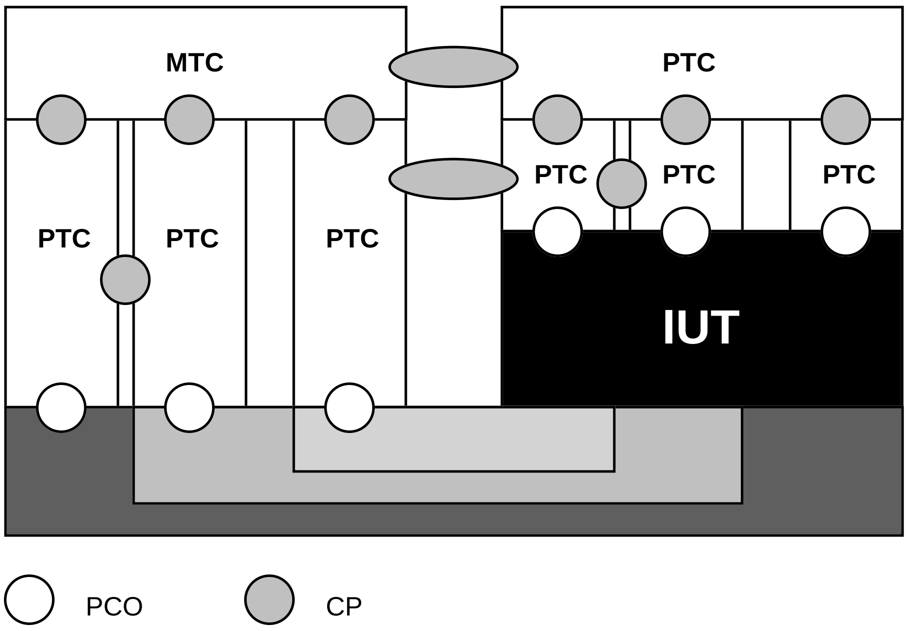
  The procedure is time-consuming.

- automatically, in which both test equipment would be remotely controllable, so that just one test operator could be able to run both parts of the test case.

When it is preferable to use an automatic synchronization, some kind of language "common to all parts" is needed to actually run a distributed test case. The present ETR specifies an example of that common language or protocol, called Test Synchronization Protocol 1 (TSP1).

6 Introduction to parallel testing concept

ISO 9646-3/AM1 [4] (concurrent TTCN) introduces the following scheme and vocabulary:



where:

- MTC stands for Main Test Component;
- PTC stands for Parallel Test Component;
- IUT stands for Implementation Under Test;
- PCO stands for Point of Control and Observation;
- CP stands for Co-ordination Point.

Figure 2: Concurrent TTCN general configuration

For NIT, the abstract configuration of figure 3 may be derived by removing all the Upper Testers (UTs) and the PTC that co-ordinates them. This represents also the Abstract Test Suite (ATS) architecture as specified in ISO/IEC 9646 [1] (MTC, PTC, CP and PCO).

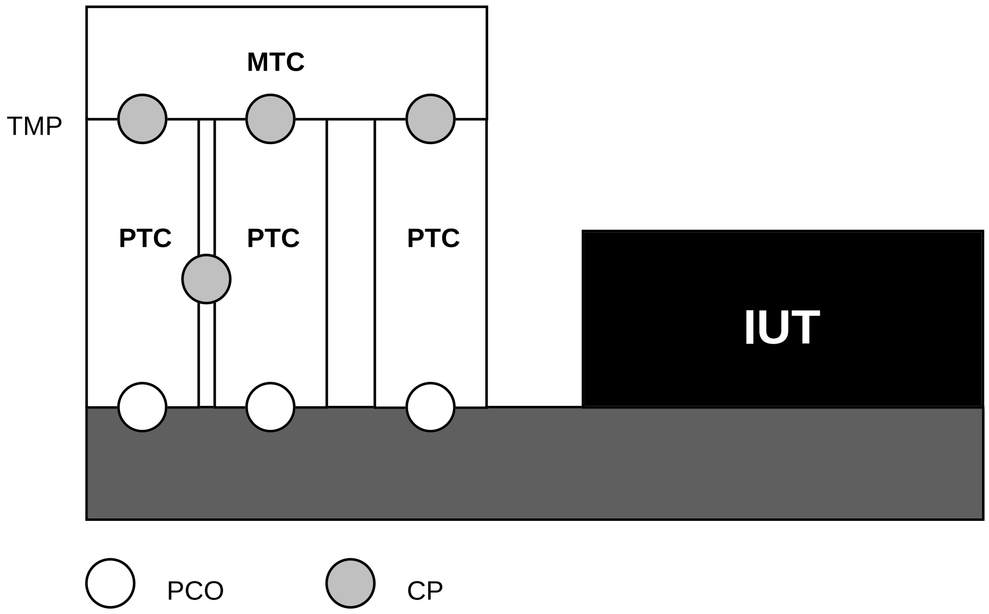


Figure 3: Example of Concurrent TTCN configuration for NIT

Control and synchronization of PTCs is done using Co-ordination Points (CPs) and Co-ordination Messages (CMs). In the ATS, the syntax of CMs and the encoding are defined in ETR 193 [5].

7 NIT implementation aspects

As already mentioned in ETR 193 [5], the testing system for the integration of a national network into the European network should not depend on a local approach. It should rather be a distributed system with the possibility of extending it to WAN-level and of adapting it to various network configurations.

This means it must be a modular structure so that it can integrate other testing systems which themselves act as subsystems. Generally, one can define a list of requirements which can simplify the choice of a testing system:

- possibilities and flexibility of a highly widespread operating system;
- standardized user interfaces and shell environments;
- editors, revision control, protection for resources and data;
- professional development environment with debugger, library and macros;
- programmable protocol-state-machines;
- reliable and consistent even as distributed system;
- availability and support of interpreters and compilers for all relevant programming languages;
- extension onto LAN/WAN level capability;
- possibilities for analysing test results;
- revision control and protection of results;
- piloting aspects (ability to be controlled by an external machine);
- methods for decoding the protocol;
- automatic analysis and report generation;
- user friendly environment.

8 Test Synchronization Architecture

The Test Synchronization Architecture is shown in figure 4. This figure describes how, starting from Multi-Party Testing Method (MPTM), Test Synchronization Architecture is created inserting a middle layer functional entity called Front End (FE). In fact with Multi-Party Testing one can realize all the configurations necessary to check simultaneously several interfaces. Referring to MPTM, System Supervisor has the function of Lower Tester Control Function (LTCF), and each Test Component is a Lower Tester (LT). Front Ends are only a way to solve the communication problems that Tester cannot solve, and a way to give to the System Supervisor a homogeneous and logical view of the testers in terms of test components.

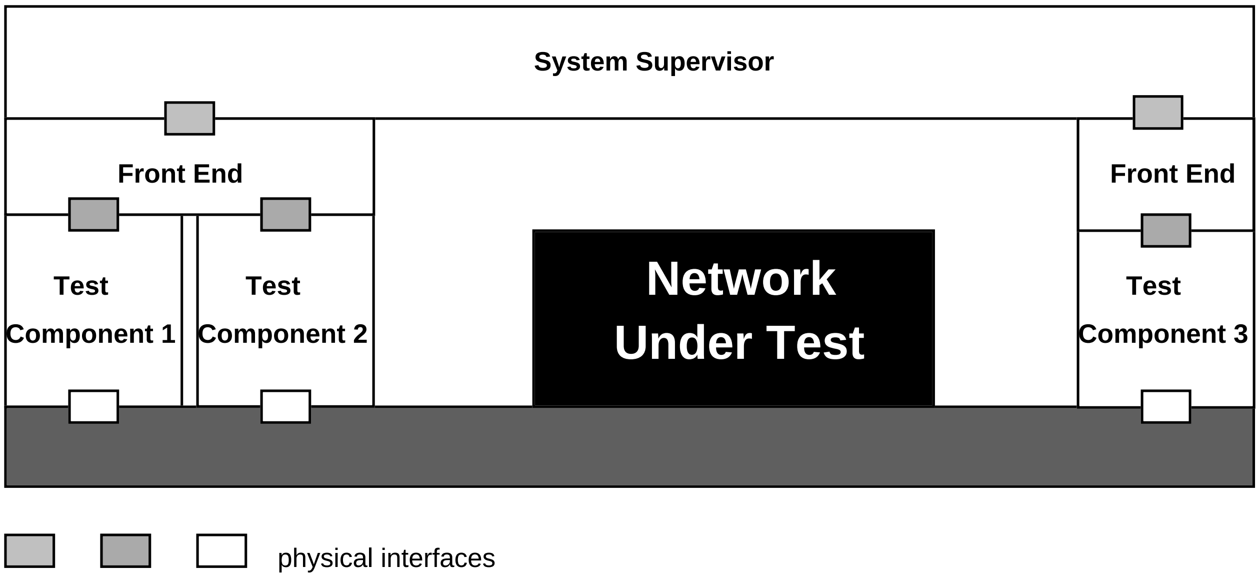


Figure 4: Test Synchronization Architecture

With the introduction of the Front End in the TSA, the concept of "virtual tester" has been introduced. Virtual tester means that each Front End gives a homogeneous view of the tester controlled by the System Supervisor. In this way each tester can be managed as a generic test component(s) without regard to the supplier of the tester machine.

An example of Test Synchronization Architecture applied to Network Integration Testing is shown in figure 5. In this figure one can have various groups of protocol testers (PTs) in different (geographically distributed) places. Each PT is controlled by an FE that is able to communicate with a System Supervisor by means of a high-level protocol. The necessity of a high-level protocol is to solve communication problems between System Supervisor and FE. Each FE can control various PTs that are not far from it with a simple proprietary protocol between FE and PT.

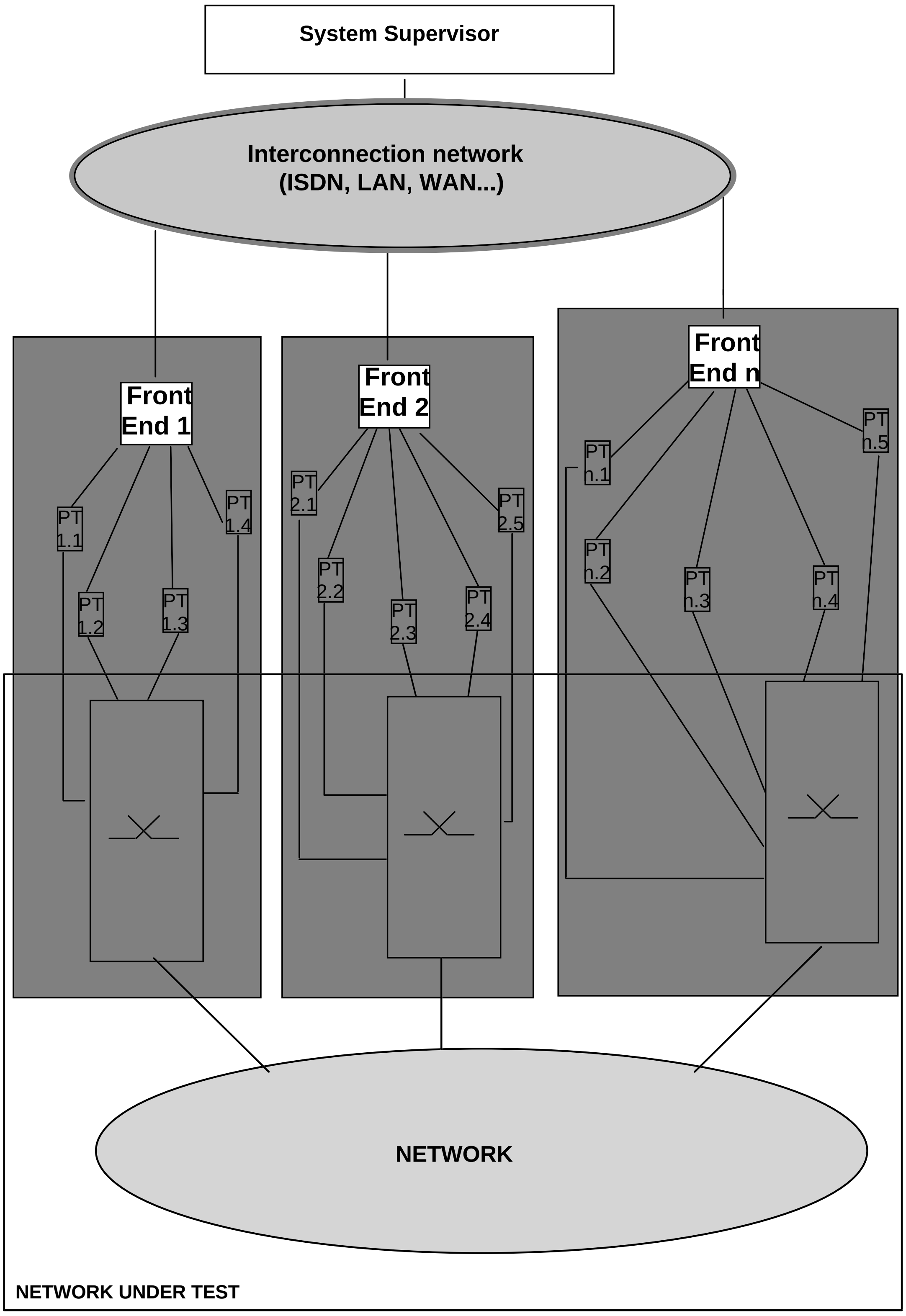


Figure 5: Test Synchronization Architecture applied on NIT

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

The functionality of the different components are:

System Supervisor: Manages the test execution but does not provide any support to implement the necessary configurations on the physical machines like the tester and/or the IUT (as that configuration can be obtained using TMN or a manual approach). The test configuration must be known, well identified, and must be set up before starting the test campaign.

Front End: Has two functions:

1) Translates system supervisor messages to messages known by each physical tester;
2) Avoids a message being transferred to the system supervisor if it is sent from test component 1 to test component 2, such messages being handled by the same Front End. In other cases, the synchronization messages are transferred to the System Supervisor, which provides routeing of the message to the right Front End.

Test Component: Is an element in charge of handling test interfaces.

The communication between these components may be represented as follows:

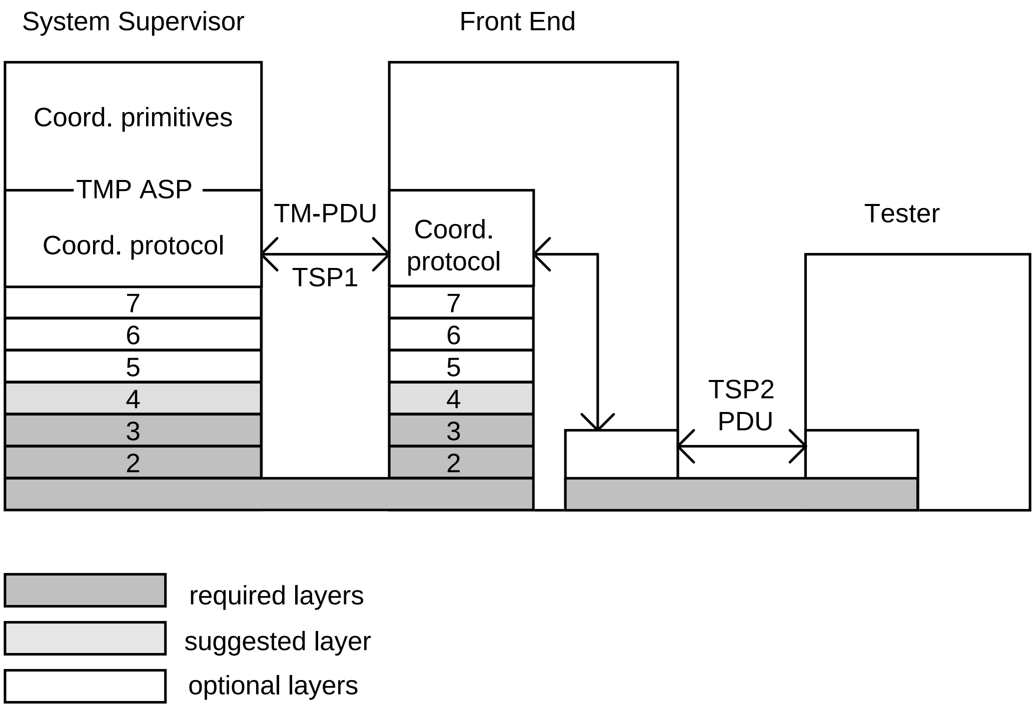


Figure 6: TSP protocols

The co-ordination messages to be described use the services provided by the OSI stack over the 3rd to 7th layers. Then they can be transported by any transport mechanism.

The software architecture for the co-ordination services is shown in the upper part of figure 6.

Some possible alternative solutions of the OSI stack 1 to 7 are showed in the lower part.

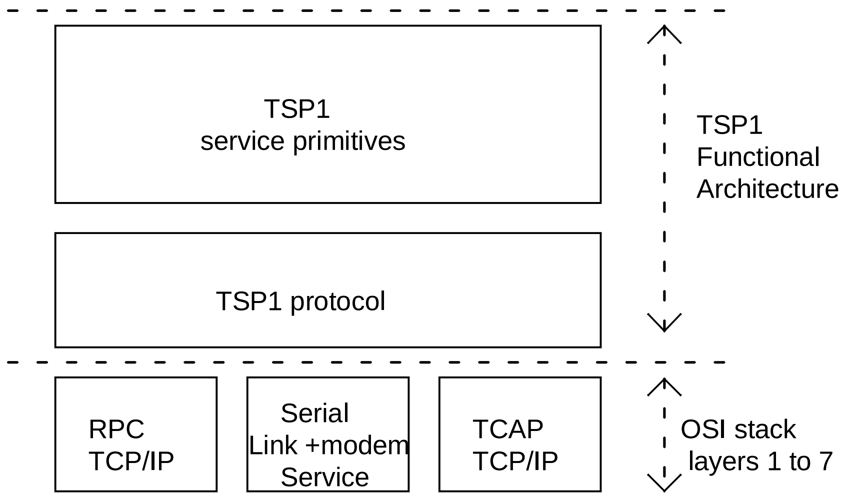


Figure 7: TSP1 functional architecture

Figure 6 shows the structure of the connection between TS elements, and the protocols used to connect each of 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 well known to System Supervisor and each Front End. TSP1 can use the services of many protocols 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 suite for each type of network (e.g. TCP/IP for Internet).

TSP1 messages decoded by Front End are sent to a PT using a protocol (TSP2 in figure 6), whose implementation knowledge concerns a Front End with its PT. Another Front End can use another protocol TSP2 to drive its PT, because this is a local problem.

The structure in figure 4 does not indicate that the System Supervisor has to be in a place far from each Front End. In fact the system can be either in the same machine of one front end in the test architecture, or in a different machine but in the same Front End place. This could allow a "test island" to act in a first instance only as a front end and in a second instance as system supervisor plus front end.

Another feature is that with this three level architecture, different protocol testers coming from different suppliers can be controlled with the same TSP1 set of messages. The Front End is in charge of converting TSP1 to the right proprietary TSP2 messages toward PTs.

8.1 System Supervisor requirements

System Supervisor manages the test case execution. Before running the test, SS can verify that all test components involved in the test are ready to start. After the end of the test or after the end of test suite execution, SS can fetch the trace of each execution. Concerning this behaviour SS will handle:

- management of the address table of the test components (mapping between each test component and its FE);
- routeing capabilities toward its FEs;
- communication with FEs;
- management of test session.

8.2 Front End requirements

FE communicates with SS on one side, and on the other side with the PTs. It routes toward its test components all the synchronization messages that it has to handle. If FE identifies a message that has not been sent to its test component, it forwards that message to SS, which serves to send it to the right destination. So concerning this behaviour, FE has the following requirements:

- routeing capabilities toward its Test Components;
- communication with SS;
- communication with its PTs.

9 Overview of TSP1

Test Synchronization Protocol 1 is a high level synchronization protocol for test procedures. It contains all the primitives and messages in order to manage a complete testing session.

This protocol is made of five groups of services to be provided during the different test phases or occurrences:

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

The description of this protocol begins with the TSP1 interface model (describing the relations among TSP1 elements). Then the TSP1 primitives are described. The PDUs flow between the SS and FE, the ASN.1 message formats and coding, and the SDL specification follow.

10 TSP1 objects models

All the following object models describe a TSP1 architecture from different, complementary, points of view (using OMT notation [7]):

- architectural point of view: system model;
- test suite point of view: TTCN model;
- service primitives point of view: interface model;
- test traces and logs point of view: traces and logs model;
- process point of view: SDL processes model;
- error handling point of view: errors model.

For readers unfamiliar with OMT graphical notation, a brief description in natural language follows each model. However, there is no extensive use of the complete modelling technique: just one (object model) out of the three OMT models (object, dynamic and functional models) is used to describe data and structural parts of the TSP1 architecture.

OMT object models introduce SDL architecture and data.

SDL specification deals with all dynamic aspects of the OMT "active" classes (of process class model).

10.1 System model

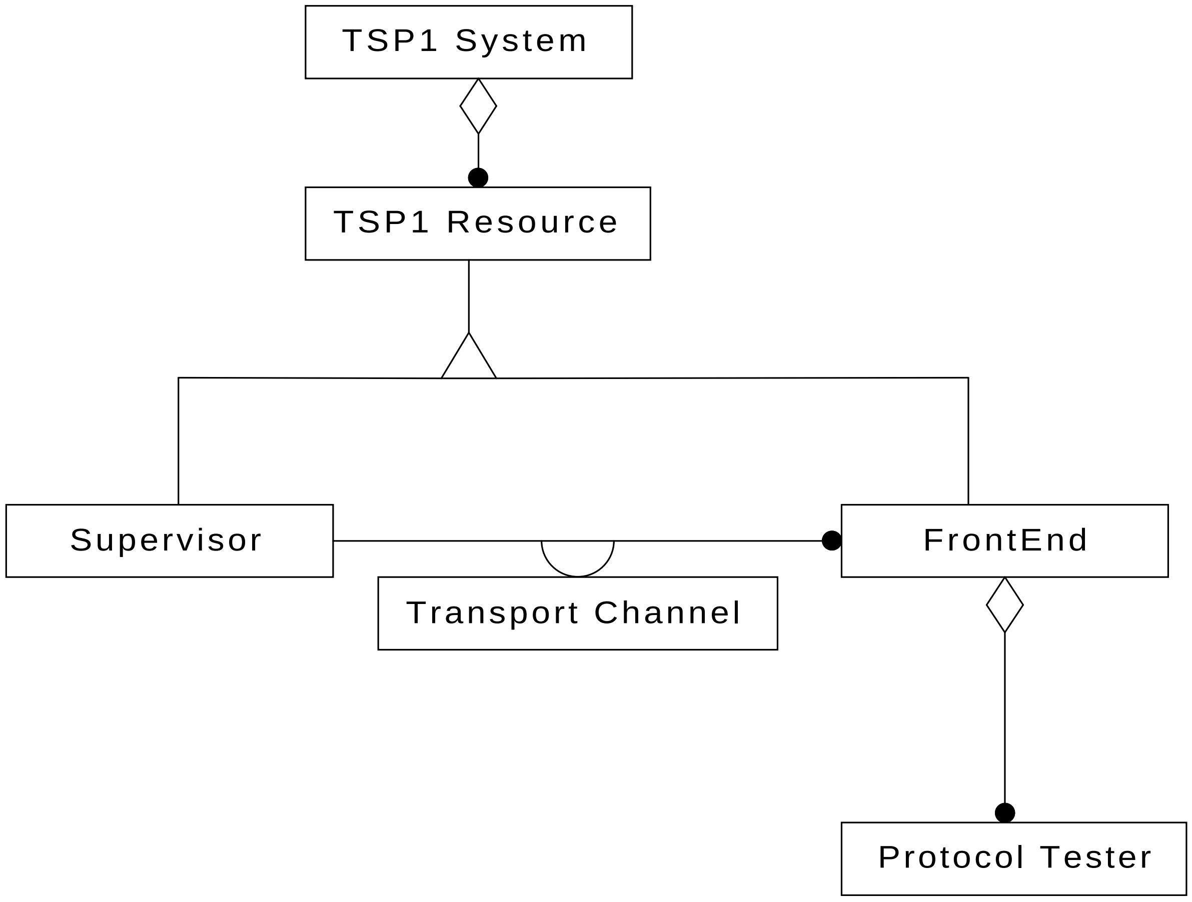


Figure 8

A TSP1 system is composed of several TSP1 resources (aggregation relationship).

There are two kinds of TSP1 resources: supervisor and front end (specialization relationship).

A supervisor can manage several front ends (association relationship).

For each management link, there must be a network transport channel in order to exchange TSP1 PDUs (Transport Channel is an association class).

A front end deals with several protocol testers (aggregation relationship).

10.2 TTCN model

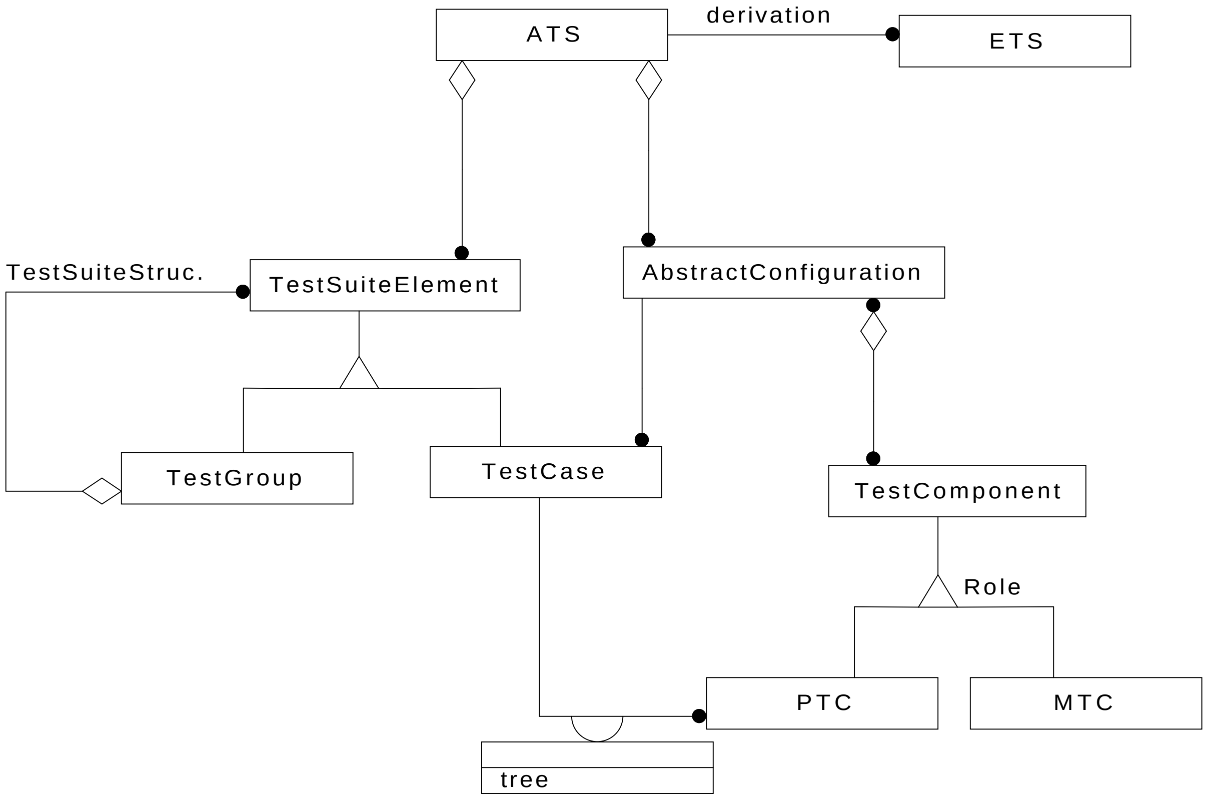


Figure 9

An abstract test suite (ATS) is composed of test suite elements which can be either test groups or test cases. A test group is itself composed of test suite elements. These two aggregation relationships are conceptualizing the TTCN test suite structure.

Several executable test suites (ETS) can be derived from one ATS.

A (concurrent TTCN) ATS defines several abstract configurations which reflect the test distribution choices when using a multi-party testing method.

A test configuration is made of several test components: there are two sorts of test components, the parallel test components (PTC) which are exciting and observing the IUT and the main test component (MTC), essentially in charge of launching the PTC and collecting and consolidating their verdicts.

Each test case is specified in the context of a particular abstract test configuration. A test case behaviour is the behaviour of its MTC. The MTC gives a behaviour to each PTC by assigning it a special tree during the creation phase (tree is an association attribute).

10.3 Interface model

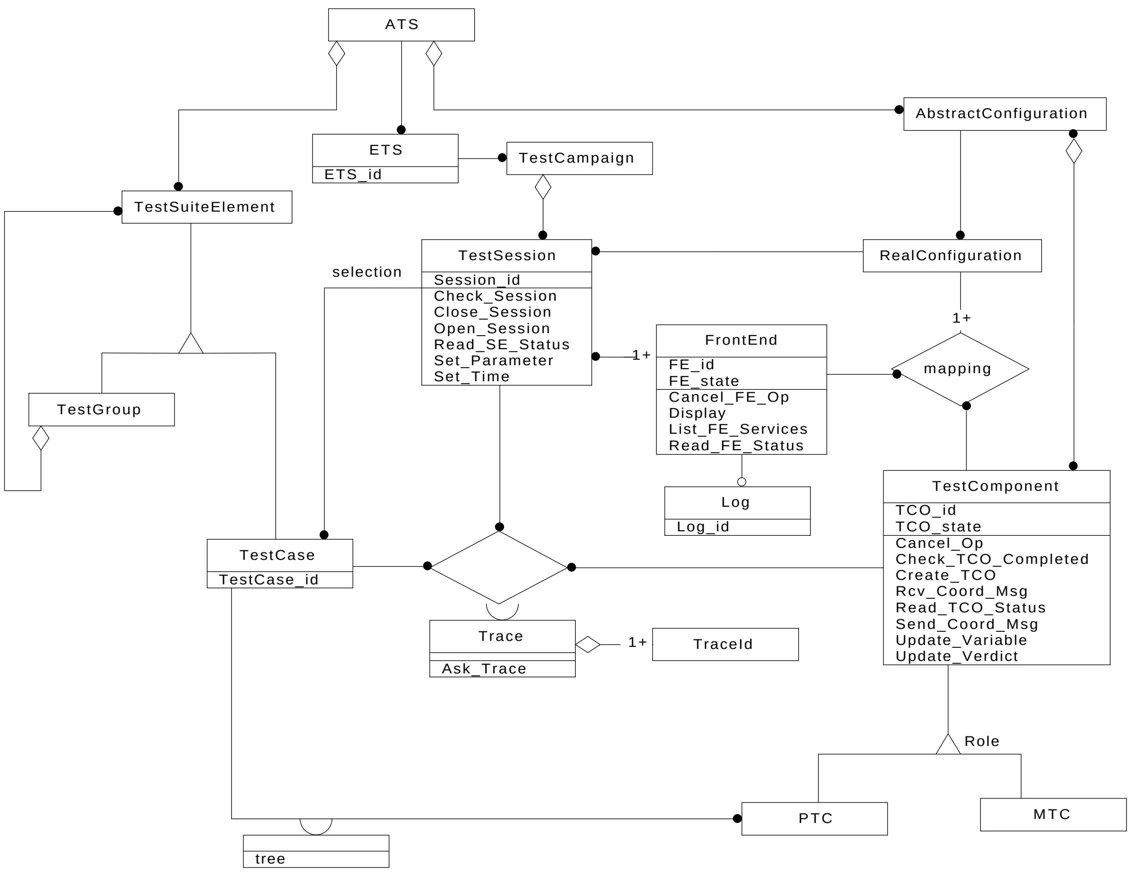


Figure 10

This 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 operations of these classes.

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

From TTCN model:

- ATS, ETS, TestSuiteElement, TestGroup, TestCase;
- AbstractConfiguration, TestComponent, MTC and PTC.

From TSP1 system model:

- FrontEnd.

New classes introduced by this model:

- a TSP1 campaign deals with one TTCN standard;

- 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;

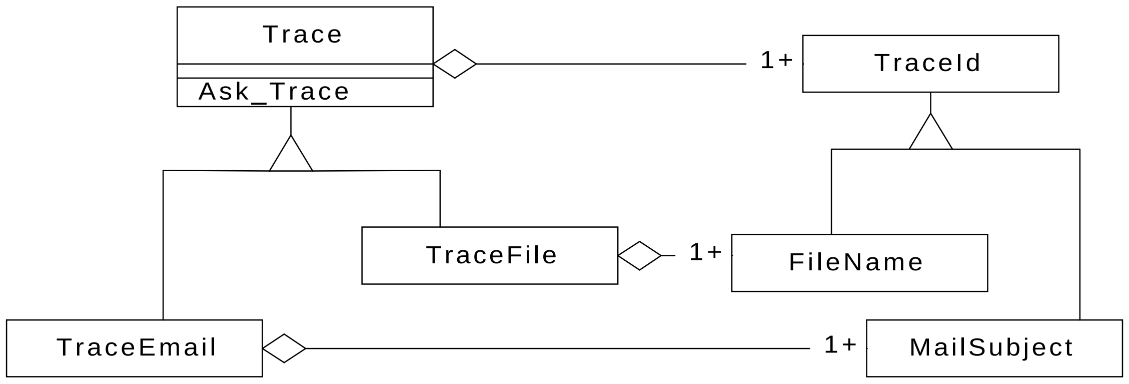
- a real (test) configuration is composed by choosing which front end will be in charge of handling each TTCN test component (this is modelled by the ternary association "mapping");

- a test (component) trace is identified by its test session, its test cases and the test component which produced it (this is modelled by the ternary association class "trace");

- an FE can have an event log (at most one);

- a trace can be composed of several TraceId (at least one).

10.4 Traces and logs model



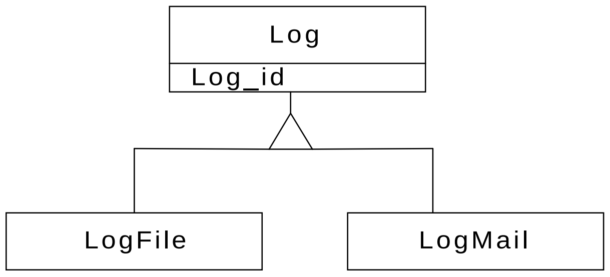


Figure 11

Test synchronization protocol does not deal with trace and log transfer (which can be of multiple kinds).

This model shows two possible sorts of log and traces (it will be able to be extended later). TSP1 is just in charge of transmitting the id and the kind of traces, whatever the real traces contain.

A trace can be a file trace, identified by a file name (unique for a front end).

A trace can be a mail content, identified by a mail subject (for example).

Same thing for the front ends events log.

10.5 SDL processes model

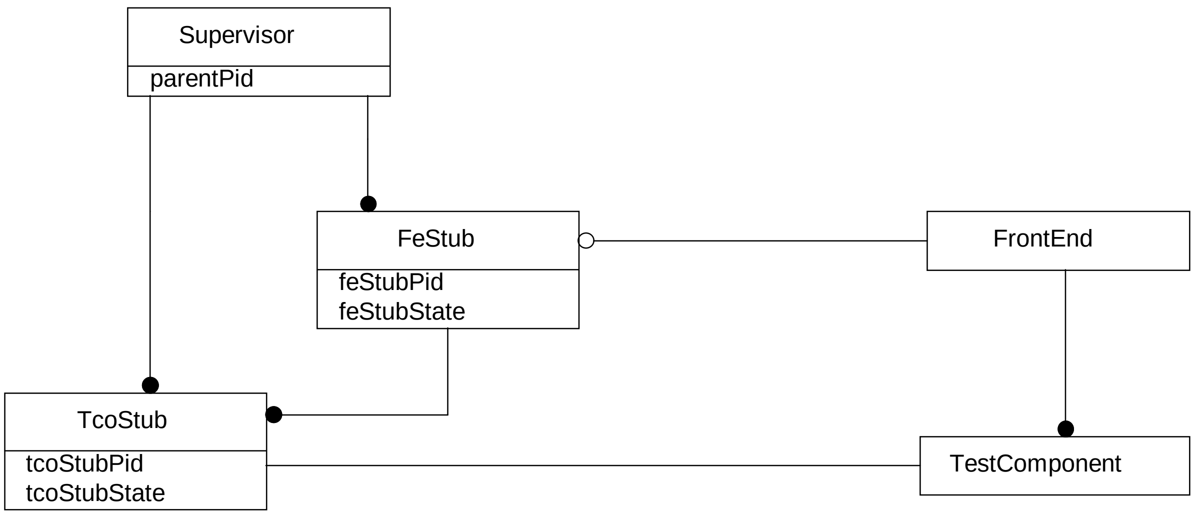


Figure 12

The supervisor process starts:

- as many FeStub (front end stub) processes as needed (one for each front end of the current session);

- as many TcoStub (test component stub) processes as needed (one for each test component of the abstract configuration of the current test session).

A front end process starts:

- as many TestComponent processes as needed (one for each test component of the abstract configuration of the current test session).

Each front end stub is in charge of one front end.

Each test component stub is in charge of one test component.

10.6 Error model

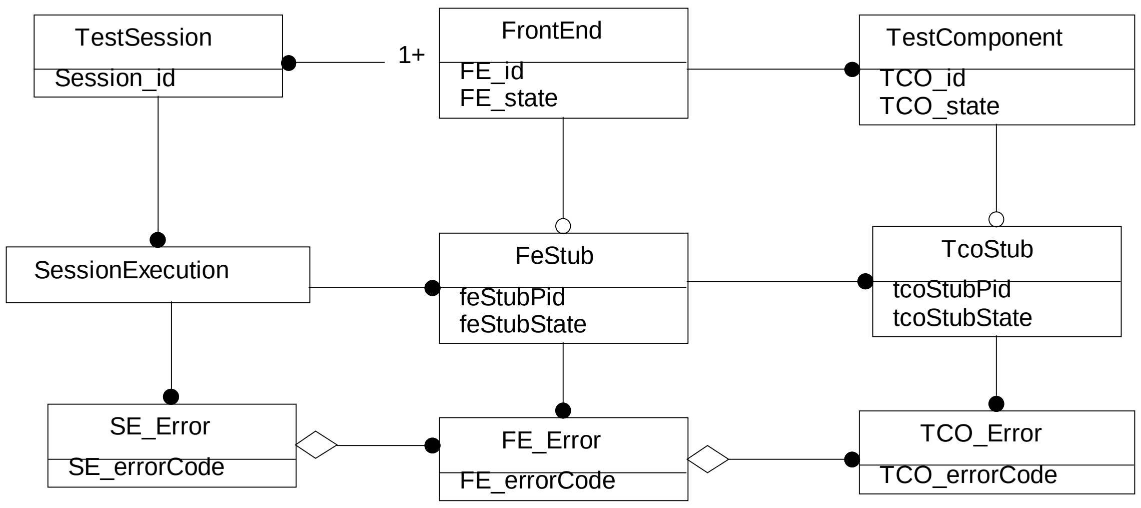


Figure 13

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

During a test execution, the supervisor uses front end stubs and test component stubs processes in order to manage all the real platform processes (FrontEnd and TestComponent).

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

- a session error contains a session error code and comprises several front end errors;

- a front end error contains a front end error code, a front end stub error code and comprises several test component errors;

- a test component error contains a test component error code and test component stub error code.

11 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.

11.1 Group 1

The Group 1 services aim to open a test session, to verify and establish a session of testing, providing all the parameter values.

11.1.1 Test configuration establishment

This function allows the System Supervisor to open a session of testing, as written in ATS test component configuration and in other documents explaining the location of the test components during the test execution.

OPEN_SESSION (ETS_ID, SESSION_ID, SE_ERROR)

where:

ETS_ID	is the executable test suite identification;
SESSION_ID	is the session identifier that has to be opened;
SE_ERROR	is the returned error code for this operation.

11.1.2 Test session checking

This function allows the System Supervisor to verify if a session, which includes the test component configuration as written in ATS, is still established. A session has to be initialized first.

CHECK_SESSION (SE_ERROR)

where:

SE_ERROR	is the returned error code for this operation.

11.1.3 Modification of the test suite parameters

This function allows the System Supervisor to change the values of the test suite parameters if required. A session has to be initialized first.

SET_PARAMETER (PARAM_LIST, SE_ERROR)

where:

PARAM_LIST	is the list of the name and value of the parameter to be changed;
SE_ERROR	is the returned error code for this operation.

11.1.4 Setting a unique time stamp

This function allows the System Supervisor to synchronize all the test components with the same time stamp. A session has to be initialized first.

SET_TIME (TIMESTAMP, SE_ERROR)

where:

TIMESTAMP	is the reference time stamp of the System Supervisor;
SE_ERROR	is the returned error code for this operation.

11.1.5 Looking for the TSP1 services available in the Front End

This function allows the System Supervisor to get the list of services available in the FE which is going to manage such an ETC.

LIST_FE_SERVICES (FE_ID, SERVICE_LIST, FE_ERROR)

where:

FE_ID	is the front end identification;
SERVICE_LIST	is the list of the TSP1 services implemented for the requested ETC;
FE_ERROR	is the returned error code for this operation.

11.1.6 Synthesis

Table 1

Action	ATS	Co-ordination services
OPEN_SESSION primitive is used to open a testing session	None	OPEN_SESSION (ETS_ID, SESSION_ID, SE_ERROR)
CHECK_SESSION primitive is used to check a testing session	None	CHECK_SESSION (SE_ERROR)
SET_PARAMETER primitive is used to change the parameter values in the ETS	None	SET_PARAMETER (PARAM_LIST, SE_ERROR)
SET_TIME primitive is used to fix the same time stamp in all the test component	None	SET_TIME (TIMESTAMP, SE_ERROR)
LIST_FE_SERVICES primitive is used by the SS to know the TSP1 services implemented for the requested FE	None	LIST_FE_SERVICES (FE_ID, SERVICE_LIST, FE_ERROR)

11.2 Group 2

The services provided by Group 2 aims to give the capabilities to run the test. The main objective is to co-ordinate the execution in terms of test launching, synchronization actions, and verdict assignment.

Differently from the previous group, primitives introduced here correspond to concepts and keywords introduced in concurrent ISO/IEC 9646-3/AM1 [4].

The preliminary results in TTCN are passed implicitly. They do not need to be passed explicitly using co-ordination messages but can be implemented using the co-ordination mechanisms between FE and System Supervisor.

11.2.1 Test execution launch

This function allows the executable main test component to load and start execution of an executable parallel test component.

CREATE_TCO (TCO_ID, TEST_CASE_ID, TREE, PARAM_LIST, TCO_ERROR)

where:

TCO_ID	is the executable test component identification;
TEST_CASE_ID	is the test case identification to be launched;
TREE	is the subtree identification to be launched;
PARAM_LIST	is the list of the test case parameters;
TCO_ERROR	is the returned error code for this operation.

11.2.2 Synchronization

This function allows the exchange of synchronization messages among parallel test components and between PTCs and the main test component. Two functions are provided: receive a message from a co-ordination point and send a message to a co-ordination point. The co-ordination point model is a first-in first-out (FIFO) queue, as specified in the ISO/IEC 9646-3/AM1 [4].

RCV_COORD_MSG (TCO_SOURCE_ID, CP_ID, COORD_MSG, TCO_ERROR)

where:

TCO_SOURCE_ID	is the executable test component identification, to which the CM is sent;
CP_ID	is the co-ordination point interface identification;
COORD_MSG	is the message received from the interface;
TCO_ERROR	is the returned error code for this operation.

The RCV_COORD_MSG function waits for a co-ordination message to be received in the local queue related to the specified co-ordination point and extracts the message if any. The transport services are in charge of filling-in the queue.

SEND_COORD_MSG (TCO_DEST_ID, CP_ID, COORD_MSG, TCO_ERROR)

where:

TCO_DEST_ID	is the executable test component identification, to which the CM is sent;
CP_ID	is the co-ordination point interface identification;
COORD_MSG	is the message received from the interface;
TCO_ERROR	is the returned error code for this operation.

The SEND_COORD_MSG function sends a co-ordination message to the queue related to the specified co-ordination point. The transport services are in charge of transmitting the message.

11.2.3 Test completion

The main test component has to check for the execution completion of a parallel test component. This is performed by the DONE TTCN keyword in the ATS. The CHECK_TCO_COMPLETED function checks if the execution of a parallel test component is completed waiting for the conclusive verdict message.

CHECK_TCO_COMPLETED (TCO_ID, TCO_VERDICT_TYPE, TCO_VERDICT_VALUE, TCO_ERROR)

where:

TCO_ID	is the parallel test component identification to be checked for completion;
TCO_VERDICT_TYPE	specifies if the verdict specified is a FINAL local verdict;
TCO_VERDICT_VALUE	is the local verdict value (PASS, FAIL or INCONCLUSIVE);
TCO_ERROR	is the returned error code for this operation.

11.2.4 Temporary verdict assignment

The executable parallel test components have to send information containing the verdict when a temporary local verdict is assigned (the final verdict is treated in test completion part).

UDPATE_VERDICT(TCO_ID, TCO_VERDICT_TYPE, TCO_VERDICT_VALUE, TCO_ERROR)

where:

TCO_ID	is the parallel test component identification to be checked for completion;
TCO_VERDICT_TYPE	specifies if the verdict specified is an INTERMEDIATE local verdict;
TCO_VERDICT_VALUE	is the local verdict value (PASS, FAIL or INCONCLUSIVE);
TCO_ERROR	is the returned error code for this operation.

11.2.5 Global variable update

In the ATS could be present some global variable. During the execution, the content of a global variable could change. In this case is necessary that the new value is updated. ATS global variable modification is no longer supported in TTCN ATS specification. For this reason this service is not specified in SDL.

UPDATE_VARIABLE (TCO_SOURCE_ID, VARIABLE_NAME, VARIABLE_VALUE, TCO_ERROR)

where

TCO_SOURCE_ID	specifies the test component which generate the message;
VARIABLE_NAME	is the name of the ATS global variable that has to be updated;
VARIABLE_VALUE	is the value of the ATS global variable has to be updated;
TCO_ERROR	is the returned error code for this operation.

11.2.6 Synthesis

Table 2

Action	ATS	Co-ordination services
CREATE_TCO primitive is used to load and start a test	CREATE (PTC: TREE)	CREATE (TCO_ID, TEST_CASE_ID, TREE, PARAM_LIST, TCO_ERROR)
SEND_COORD_MSG primitive is used to send a co-ordination message though a co-ordination point	CP!CM	SEND_COORD_MSG (TCO_DEST_ID, CP_ID, COORD_MSG, TCO_ERROR)
RCV_COORD_MSG primitive is used to wait for a co-ordination message though a co-ordination point	CP?CM	RCV_COORD_MSG (TCO_SOURCE_ID, CP_ID, COORD_MSG, TCO_ERROR)
CHECK_TCO_COMPLETED primitive is used to check that a test component ends the running test (waits for a the final verdict)	?DONE (PTC)	CHECK_TCO_COMPLETED (TCO_ID, TCO_VERDICT_TYPE, TCO_VERDICT_VALUE, TCO_ERROR)
UPDATE_VERDICT primitive is used to wait for a partial verdict	Verdict passing is implicit.	UPDATE_VERDICT (TCO_ID, TCO_VERDICT_TYPE, TCO_VERDICT_VALUE, TCO_ERROR)
UPDATE_VARIABLE primitive is used to update all the instances of the same variable in all the test components	ATS global variable modification is no longer supported in TTCN ATS specification.	UPDATE_VARIABLE (TCO_SOURCE_ID, VARIABLE_NAME, VARIABLE_VALUE, TCO_ERROR)

11.3 Group 3

The Group 3 services described in this subclause provide a way of asking for traces and other information related to the result of the execution of the TCs. All these services would be used after the execution of the test components, usually in order to get information to build the test report.

11.3.1 Asking for a trace

This function allows to the System Supervisor to ask for the execution trace of a test component. It is assumed that the complete trace is requested.

ASK_TRACE (SESSION_ID, TEST_CASE_ID, TCO_ID, MEDIA, TRACE_ID_LIST, FE_ERROR)

where:

SESSION_ID	is the session identification;
TEST_CASE_ID	is the test case identification of which the trace is requested;
TCO_ID	is the test component identification;
MEDIA	is the means by which the FE is going to send the trace to the System Supervisor (ftp, e-mail, fax, etc.);
TRACE_ID_LIST	is the list of the trace-ids available in that FE;
FE_ERROR	is the returned error code for this operation.

11.3.2 Closing a test session

This functions allows to the System Supervisor to close a test session with the test components.

CLOSE_SESSION (SE_ERROR)

where:

SE_ERROR	is the returned error code for this operation.

11.3.3 Synthesis

Table 3

Action	ATS	Co-ordination services
ASK_TRACE primitive is used to ask for a trace to a test component	None	ASK_TRACE (SESSION_ID, TEST_CASE_ID, TCO_ID, MEDIA, TRACE_ID_LIST, FE_ERROR)
CLOSE_SESSION primitive is used to close a test session in a test component	None	CLOSE_SESSION (SE_ERROR)

11.4 Group 4

The service provided by Group 4 aims to solve the problem that could happen during the initialization, execution and closing phases.

11.4.1 Rejecting a corrupted or an out of sequence message

The FE has to manage a wrong reception (corrupted or out of sequence) of TSP1 messages. This is done by sending a REJECT_MSG every time FE receives a wrong message. SS is in charge to decide how many times FE can reject the message, and the right procedure to take in this case.

REJECT_MSG (FE_ID, FE_ERROR)

where:

FE_ID	is the FE identification that reject the message;
FE_ERROR	is the returned error code for this operation.

11.4.2 Cancel a running operation

The System Supervisor after having run an operation must have the capability of cancelling it. In fact, that operation could be done by chance or at the very beginning of that operation, the operator could see something wrong and so he could decide to interrupt it.

This is done using CANCEL_OP every time SS decide to interrupt such an operation. After the CANCEL_OP procedure the SS has to return in the state in which it was before that operation.

CANCEL_OP (TCO_ID, TCO_ERROR) which can be used during test execution phase.

where:

TCO_ID	is the parallel test component identification that originates the problem;
TCO_ERROR	is the returned error code for this operation.

CANCEL_FE_OP (FE_ID, FE_ERROR) which can be used during test initialization.

where:

FE_ID	is the front end identification that originates the problem;
FE_ERROR	is the returned error code for this operation.

11.4.3 Synthesis

Table 4

Action	ATS	Co-ordination services
REJECT_MSG primitive is used to reject a corrupted or an out of sequence message	None	REJECT_MSG (FE_ID FE_ERROR)
CANCEL_OP primitive is used to interrupt an operation during execution	None	CANCEL_OP (TCO_ID, TCO_ERROR)
CANCEL_FE_OP primitive is used to interrupt an operation during initialization	None	CANCEL_FE_OP (FE_ID, FE_ERROR)

11.5 Group 5

The service provided by Group 5 aims to solve problems which do not belong to the other groups.

11.5.1 The DISPLAY feature

During test execution there could be the need to display on the screen of the SS or the FE such a message. For example the FE collects data from the PT. The SS could have the need to see what happens during the test execution. With the DISPLAY feature the FE sends to the screen of the SS what it collects from the PT.

DISPLAY (FE_ID, MSG)

where:

FE_ID	is the destination front end;
MSG	is the message that has to be displayed.

11.5.2 Looking for the status of the test components

This function allows the System Supervisor to get the relevant information for the status of the test components. The SS may be allowed to invoke it in order to get a quick look at the status of the test component (if it is running, if it is ready to start, or in any other test component state).

In the following there are three primitives that can be used depending on the number of the test component status that someone wants to get (all the test components at session level, all those controlled by a FE, and only one test component).

READ_SE_STATUS (STATUS_LIST, LOG_ID_LIST, SE_ERROR)

where:

STATUS_LIST	is the list of status for each session's component;
LOG_ID_LIST	is a list of text files containing all major events recorded for each session's test components;
SE_ERROR	is the returned error code for this operation.

READ_FE_STATUS (FE_ID, STATUS_LIST, LOG_ID_LIST, FE_ERROR)

where:

FE_ID	is the FE identification;
STATUS_LIST	is the list of statuses for each TCO of the front end;
LOG_ID_LIST	is a list of text files containing all major events recorded for each test components managed by a front end;
FE_ERROR	is the returned error code for this operation.

READ_TCO_STATUS (TCO_ID, STATUS, LOG_ID, TCO_ERROR)

where:

TCO_ID	is the test component identification;
STATUS	is the status in which the test component is;
LOG_ID	is a textual file in which all major events are recorded;
TCO_ERROR	is the returned error code for this operation.

11.5.3 Synthesis

Table 5

Action	ATS	Co-ordination services
DISPLAY primitive is used to display a message on a remote screen.	None	DISPLAY (FE_ID MSG)
READ_SE_STATUS primitive is used by the SS to know the status of a all test components of the current session.	None	READ_SE_STATUS (STATUS_LIST, LOG_ID_LIST, SE_ERROR)
READ_FE_STATUS primitive is used by the SS to know the status of all test components managed by the front end.	None	READ_FE_STATUS (FE_ID, STATUS_LIST, LOG_ID_LIST, FE_ERROR)
READ_TCO_STATUS primitive is used by the SS to know the status of one particular test component.	None	READ_TCO_STATUS (TCO_ID STATUS, LOG_ID, TCO_ERROR)

12 TSP1 primitives (from interface model methods)

Open_Session (in: ETS_id, in: Session_id, out: SE_Error)

Check_Session (out: SE_Error)

Set_Parameter (in: Param_List, out: SE_Error)

Set_Time (in: TimeStamp, out: SE_Error)

Close_Session (out: SE_Error)

Create_TCO (in: TCO_id, in: Test_Case_id, in: tree, in: Param_List, out: TCO_Error)

Rcv_Coord_Msg (out: TCO_Source_id, out: CP_id, out: Coord_Msg, out: TCO_Error)

Send_Coord_Msg (in: TCO_Dest_id, in: CP_id, in: Coord_Msg, out: TCO_Error)

Update_Verdict (in: TCO_id, in: TCO_Verdict_Type, in: TCO_Verdict_Value, in: TCO_Error)

Update_Variable (in: TCO_Source_id, in: Variable_Name, in: Variable_Value, out: TCO_Error)

Check_TCO_Completed (in: TCO_id, out: TCO_Verdict_Type, out: TCO_Verdict_Value, out: TCO_Error)

Cancel_Op (in: TCO_id, out: TCO_Error)

Cancel_FE_Op (in: FE_id, out: FE_Error)

List_FE_Services (in: FE_id, out: Service_List, out: FE_Error)

Display (in: FE_id, in: Msg)

Ask_Trace (in: Session_id, in: Test_Case_id, in: TCO_id, out: Media, out: Trace_id_List, out: FE_Error)

Read_TCO_Status (in: TCO_id, out: Status, out: Log_id; out: TCO_Error)

Read_FE_Status (in: FE_id, out: Status_List, out: Log_id_List; out: FE_Error)

Read_SE_Status (out: Status_List, out: Log_id_List; out: SE_Error)

Reject_Msg(out: FE_id, out: FE_ERROR)

13 Protocol description

To be able to implement the above described service primitives, a simple protocol and a set of messages have to be specified. This will provide an end to end service between two testers (end to end in the sense of OSI).

The messages described hereafter are defined on top of the OSI layer, from layer 3 (better if layer 4) to layer 7. It is assumed that the transport level provides connection establishment and termination, recovery from transmission errors and flow control strategy.

13.1 Message functional definitions

The TSP1 messages require acknowledgement when they are sent by the System Supervisor. The acknowledge message is used to confirm the correct behaviour of the protocol. For the negative acknowledge, an "ad hoc" PDU called ERROR is used. In this PDU is carried the reason of failure.

The following messages are defined:

TSP1_INIT	This is a message that has the purpose of opening a session of testing. TSP1_INIT message is sent by SS at the beginning of each test session for a given session identifier. FE will enable the test component indicated in TSP1_INIT message and to send to the SS a TSP1_INIT_ACK message as soon as the TSP1_INIT message is received. If the Test Component is not ready, an TSP1_ERROR PDU will be sent with ERROR_CODE other than zero. When the Test Component has completed the load of the configuration, FE sends a TSP1_INIT_COMPLETE to the SS. This message is due to the fact that the initialization procedure could take very long time (10 - 15 minutes).
TSP1_INIT_ACK	This is a message that is sent by the FE to SS as soon as an TSP1_INIT message is received. If FE is not able to initialize the test components, a TSP1_ERROR PDU will be sent with ERROR_CODE other than zero.
TSP1_INIT_COMPLETE	This is a message that is sent by the FE to SS if the Test Component is ready to start a test for a given session. If the session is not initialized, a TSP1_ERROR PDU will be sent with ERROR_CODE other than zero.
TSP1_CHK_CONF	This is a message that has the purpose of verifying the session that has been loaded. TSP1_CHK_CONF message may be sent by SS every time is necessary to verify the session for all the Test Component Involved. FE will permit verification of the software loaded in each Test Component and will send to the SS a TSP1_CHK_CONF_ACK message. If all the software loaded is not coherent with the current session-id, a TSP1_ERROR PDU will be sent with ERROR_CODE other than zero.
TSP1_CHK_CONF_ACK	This is a message that is sent by the FE to SS to give the Test Component status for a given session. If all the software loaded is not coherent with the current session-id, a TSP1_ERROR PDU will be sent with ERROR_CODE other than zero.
TSP1_SET_PARAMETER	This message is used by the SS to change the parameter values in the Test Components without changing session identification.
TSP1_SET_PARAMETER_ACK	This message is used by the FE to acknowledge the modifications of the parameter values in the Test Components. If parameters are not set in the test components, a TSP1_ERROR PDU will be sent with ERROR_CODE other than zero.
TSP1_SET_TIME	This message is used by the SS to fix a time stamp in all the test components.
TSP1_SET_TIME_ACK	This message is used by the FE to acknowledge that a time stamp has been fixed in the test components. If time stamp has not fixed in the test component a TSP1_ERROR PDU will be sent with ERROR_CODE other than zero.
TSP1_LIST_SERVICES	This message is used by the System Supervisor to know what TSP1 services are implemented in the FE.
TSP1_LIST_SERVICES_ACK	This message is used by the FE to acknowledge the TSP1_LIST_SERVICES, and to provide the list of TSP1 services implemented. If problems occurs providing this service, a TSP1_ERROR PDU will be sent with ERROR_CODE other than zero.

TSP1_CREATE	This message is used to send information relating to the CREATE_ETC primitive. Message sent by the system supervisor (SS). Receiving this message, the FE runs the right commands to the testers to load and start the Test Components.
TSP1_CREATE_ACK	This is a supervision message used by the front end (FE) to confirm to SS that the executable test component has been started. If problems occurs providing this service, a TSP1_ERROR PDU will be sent with ERROR_CODE other than zero.
TSP1_INFO	This is a message able to carry the TCP information elements that are not TTCN keyword. For example, all co-ordination messages are carried through this message.
TSP1_VERDICT	This message is sent by the front end each time a temporary or final local verdict is assigned.
TSP1_UPDATE_VARIABLE	This message is used by the SS or FE which identify a particular global variable that has to be updated in all the test component.
TSP1_UPDATE_VARIABLE_ACK	This message is used by the SS or FE to acknowledge or not the updating of the global variable in a test component. If problems occurs providing this service a TSP1_ERROR PDU will be sent with ERROR_CODE other than zero.
TSP1_ASK_TRACE	This message is used to send information relating to the ASK_TRACE primitive. Message sent by the system supervisor (SS).
TSP1_ASK_TRACE_ACK	This is a message used by the front end (FE) to confirm to the SS that the request of a trace from a test component has been received. If problems occurs providing this service a TSP1_ERROR PDU will be sent with ERROR_CODE other than zero.
TSP1_END	This message is used by the SS to request to the FE to close the current test session with the test components.
TSP1_END_ACK	This is a message used by the front end (FE) to confirm to the SS that the current test session has been closed. If problems occurs providing this service a TSP1_ERROR PDU will be sent with ERROR_CODE other than zero.
TSP1_REJECT_MSG	This message is used by the front end (FE), to notify to the System Supervisor that a corrupted or an out of sequence message is received.
TSP1_CANCEL_OP	This message is used by the System Supervisor to cancel the running operation. This message is sent to each FE which is involved in the current session.
TSP1_CANCEL_OP_ACK	This message is used by the FE to acknowledge the TSP1_CANCEL_OP message returning to the status before the last operation was run. If problems occurs providing this service, a TSP1_ERROR PDU will be sent with ERROR_CODE other than zero.
TSP1_DISPLAY	This is a message used by the front end (FE) or the SS to display a message on the screen of the interested party (SS, in the case the message is sent by FE, and FE, PT, in the case the message is sent by SS).
TSP1_ERROR	This message is sent by the FE to the System Supervisor to give a negative acknowledge for TSP1 messages.

14 Message sequences

Here follow the message sequences exchanged among SS & FEs following the issue of a relevant service primitive.

ERROR (SERVICE_ID, ERROR_CODE, ERR-PARAM)

The use of the ERROR PDU is to give a negative acknowledge for a generic TSP1 message when an acknowledge is expected. So the field ERROR in the following primitives is provided with the PDU below.

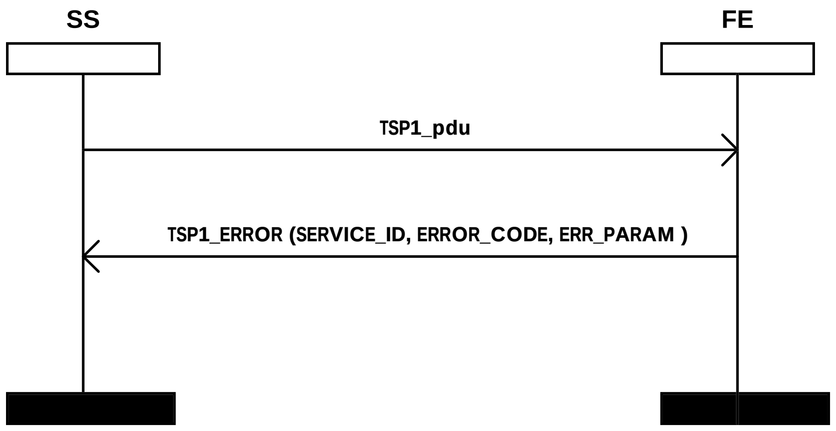


Figure 14

OPEN_SESSION (ETS_ID, SESSION_ID, SE_ERROR)

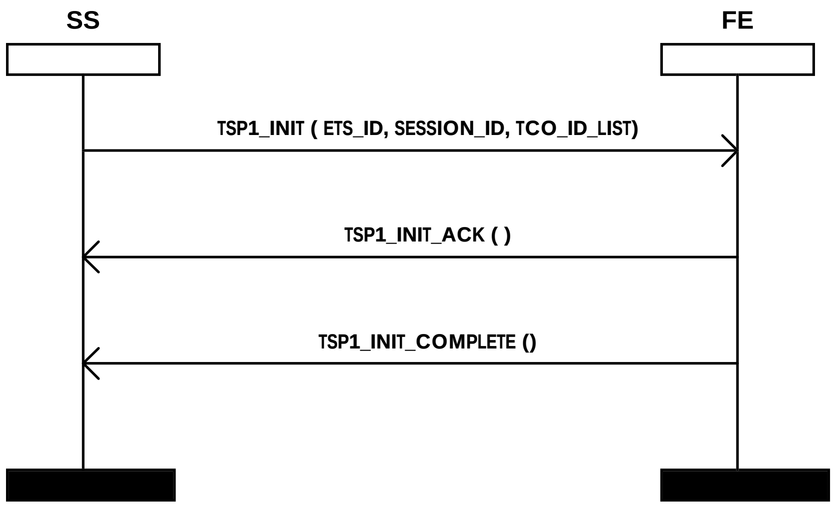


Figure 15

CHECK_SESSION (SE_ERROR)

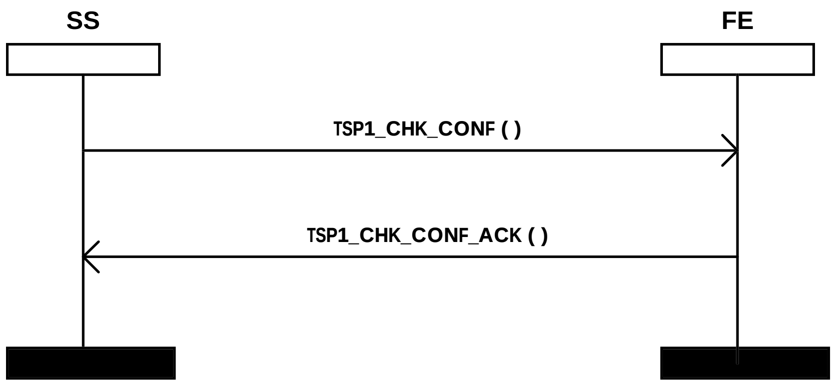


Figure 16

SET_PARAMETER (PARAM_LIST, SE_ERROR)

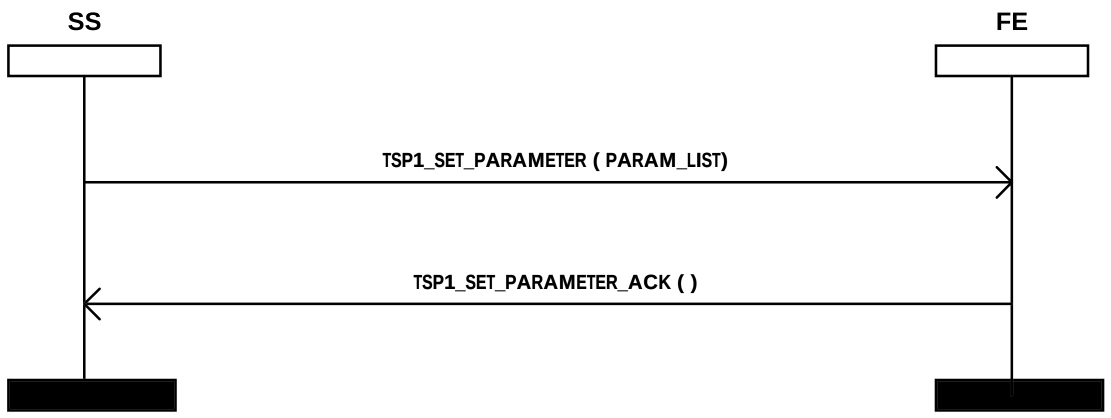


Figure 17

SET_TIME (TIMESTAMP, SE_ERROR)

To fix the same time stamp in all the ETC, is necessary to measure the delay between SS and FE (considering negligible the delay between FE and ETCs). This delay can be measured by the SS for example using the first two messages exchanged with a FE.

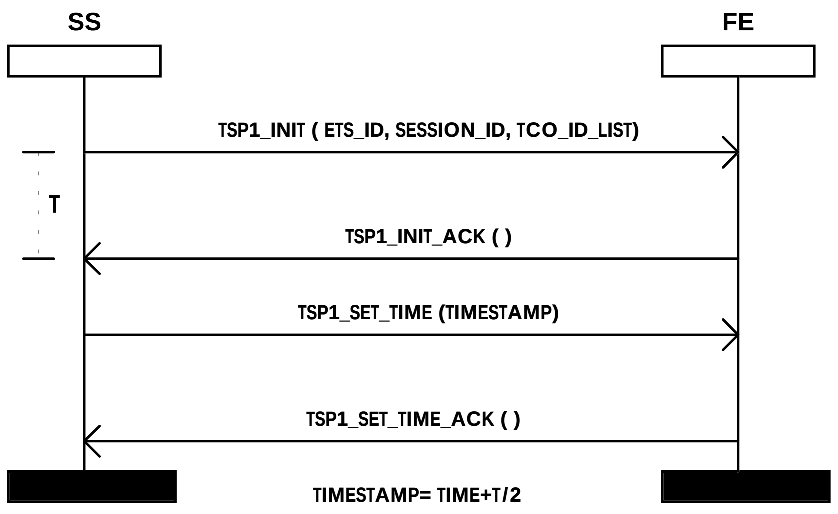

SS
FE
TSP1_INIT (ETS_ID, SESSION_ID, TCO_ID_LIST)
T
TSP1_INIT_ACK ( )
TSP1_SET_TIME (TIMESTAMP)
TSP1_SET_TIME_ACK ( )
TIMESTAMP= TIME+T/2

Figure 18

LIST_FE_SERVICES (FE_ID, SERVICE_LIST, FE_ERROR)

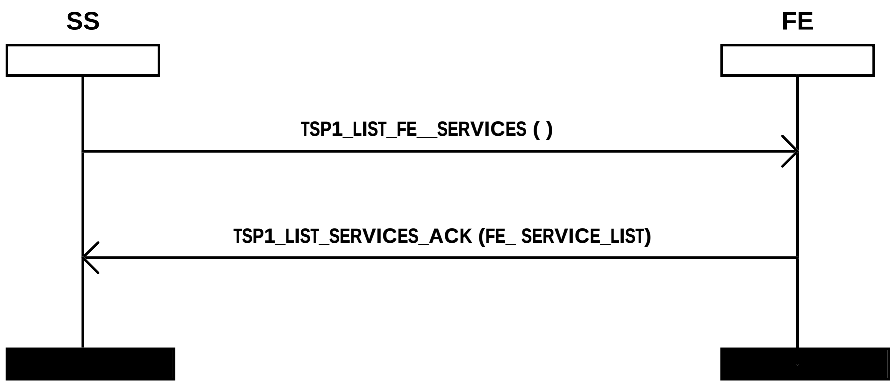

SS
FE
TSP1_LIST_FE__SERVICES ( )
TSP1_LIST_SERVICES_ACK (FE_ SERVICE_LIST)

Figure 19

CREATE_TCO (TCO_ID, TEST_CASE_ID, TREE, PARAM_LIST, TCO_ERROR)

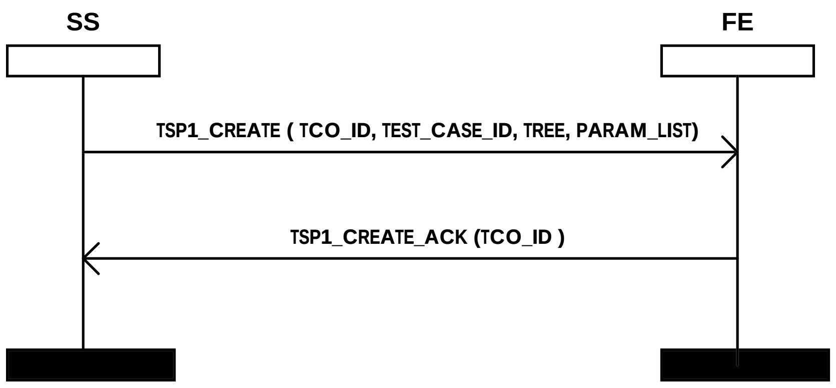

SS
FE
TSP1_CREATE (TCO_ID, TEST_CASE_ID, TREE, PARAM_LIST)
TSP1_CREATE_ACK (TCO_ID)

Figure 20

SEND_COORD_MSG (TCO_DEST_ID, CP_ID, COORD_MSG)

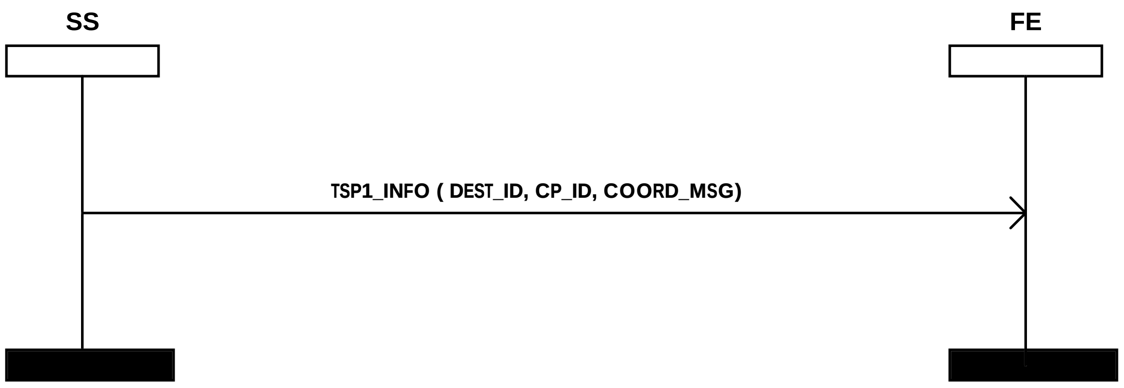


Figure 21

The following case is an example when there is a Co-ordination Message that is sent from a PTC to another PTC. This case is managed from the protocol as a routeing problem. The description of who sends all that is written in the ATS.

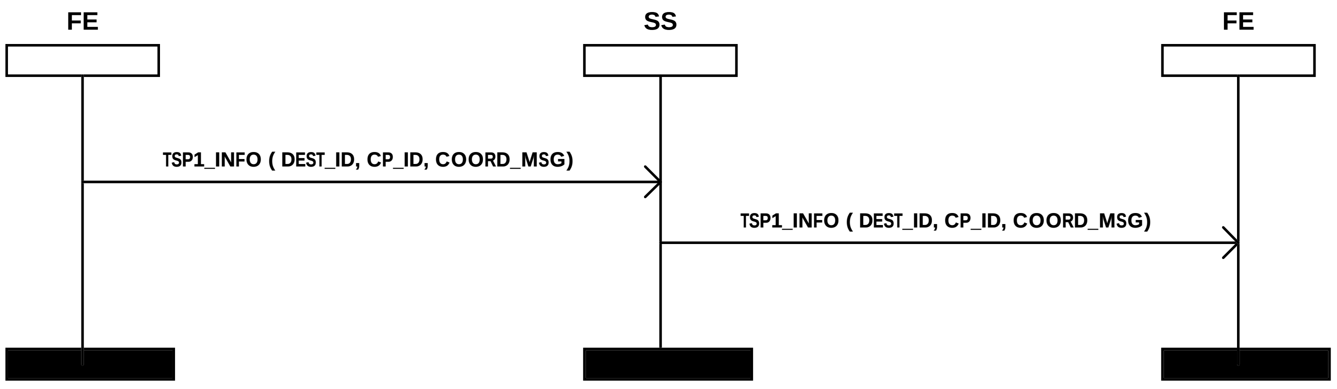

FE
SS
FE
TSP1_INFO (DEST_ID, CP_ID, COORD_MSG)
TSP1_INFO (DEST_ID, CP_ID, COORD_MSG)

Figure 22

RCV_COORD_MSG (TCO_SOURCE_ID, CP_ID, COORD_MSG)

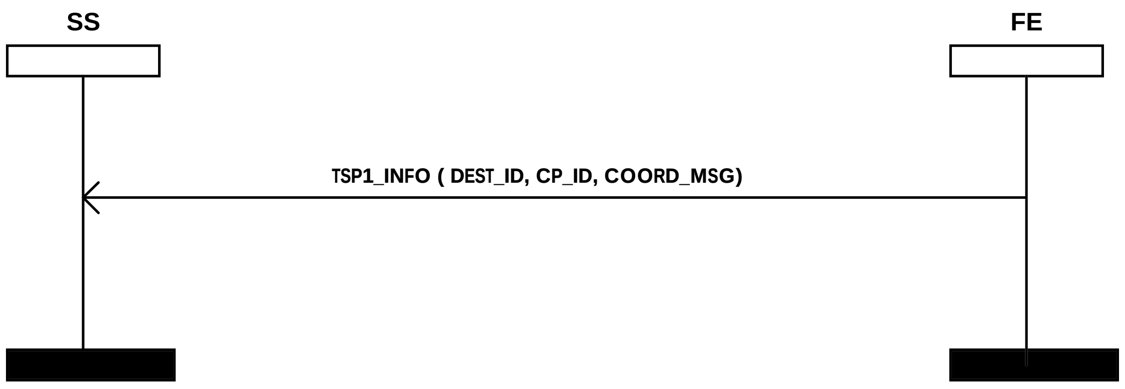


Figure 23

CHECK_TCO_COMPLETED (TCO_ID, TCO_VERDICT_TYPE, TCO_VERDICT_VALUE, TCO_ERROR)

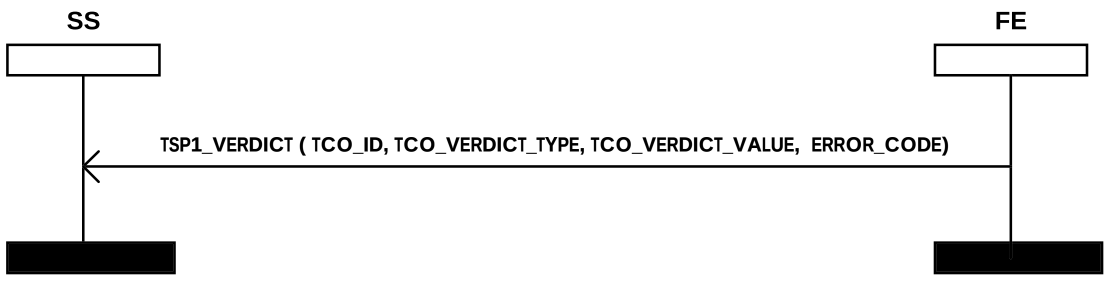

SS
FE
TSP1_VERDICT (TCO_ID, TCO_VERDICT_TYPE, TCO_VERDICT_VALUE, ERROR_CODE)

Figure 24

UPDATE_VERDICT(TCO_ID, TCO_VERDICT_TYPE, TCO_VERDICT_VALUE, TCO_ERROR)

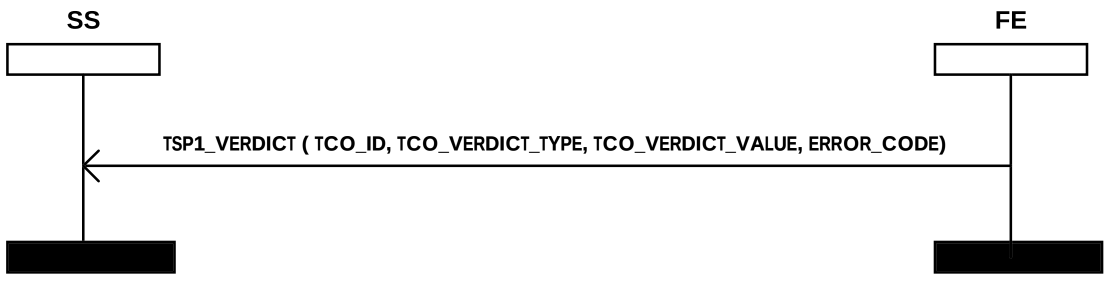

SS
FE
TSP1_VERDICT (TCO_ID, TCO_VERDICT_TYPE, TCO_VERDICT_VALUE, ERROR_CODE)

Figure 25

UPDATE_VARIABLE (TCO_SOURCE_ID, VARIABLE_NAME, VARIABLE_VALUE, TCO_ERROR)

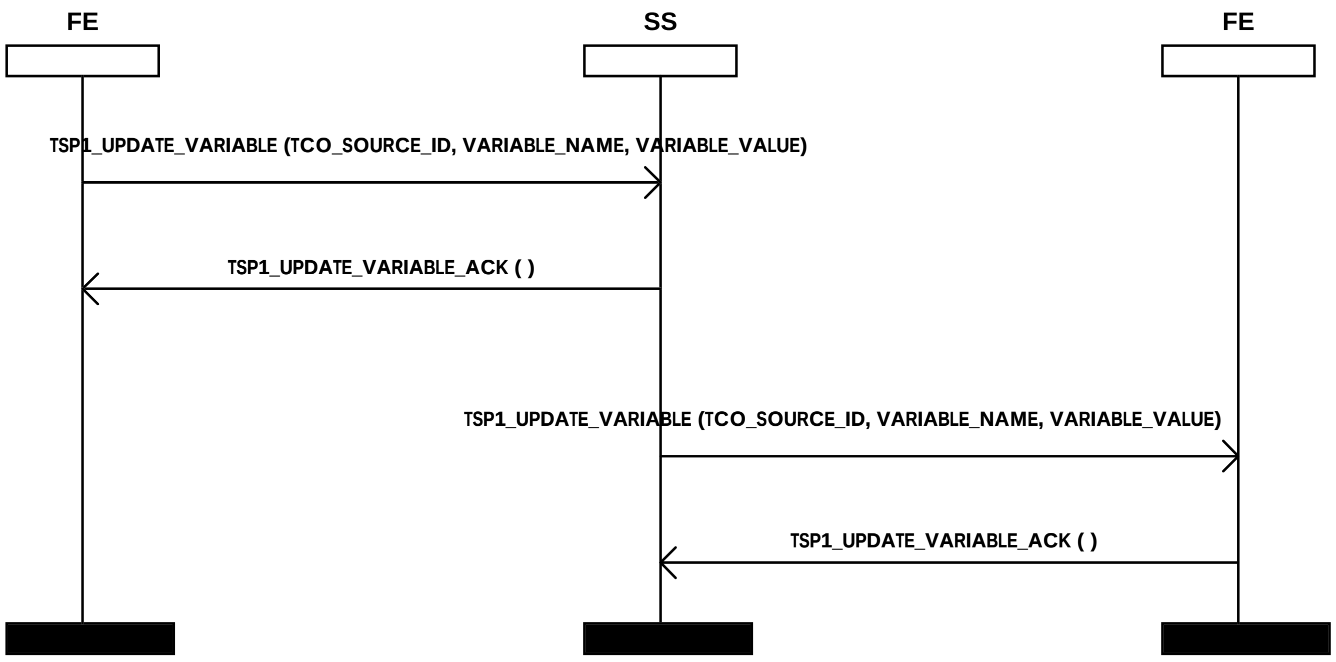

FE
SS
FE
TSP1_UPDATE_VARIABLE (TCO_SOURCE_ID, VARIABLE_NAME, VARIABLE_VALUE)
TSP1_UPDATE_VARIABLE_ACK ( )
TSP1_UPDATE_VARIABLE (TCO_SOURCE_ID, VARIABLE_NAME, VARIABLE_VALUE)
TSP1_UPDATE_VARIABLE_ACK ( )

Figure 26

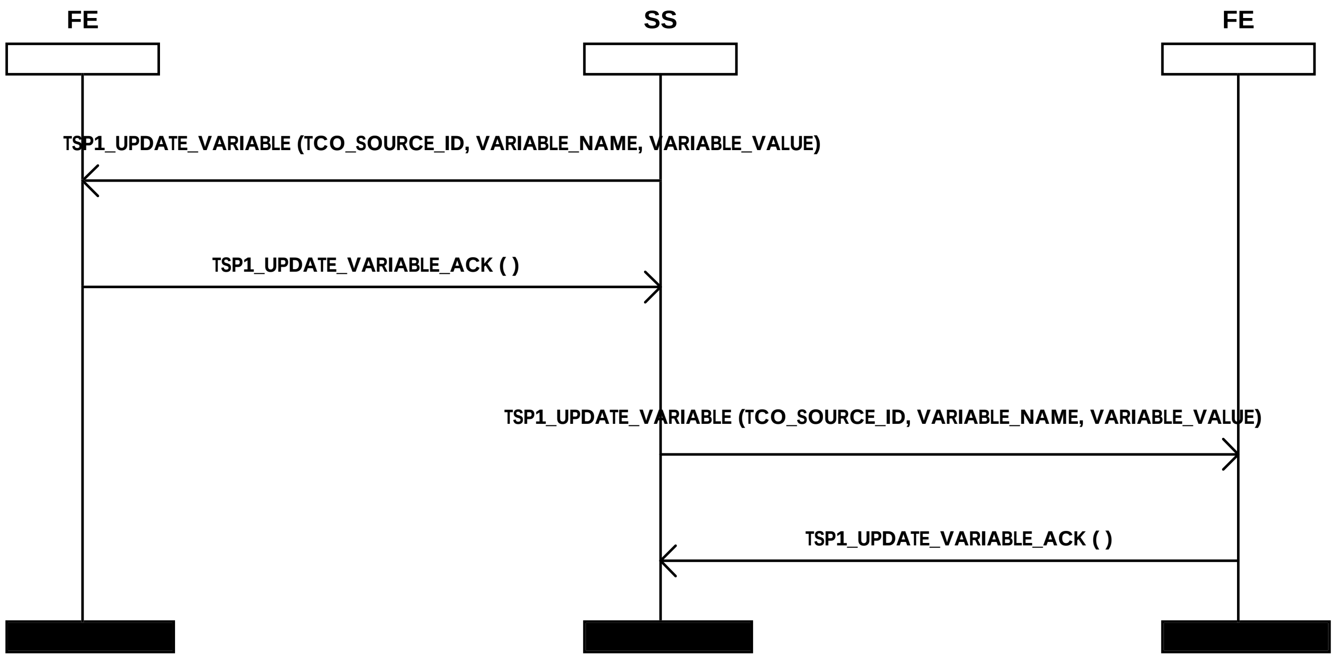

FE
SS
FE
TSP1_UPDATE_VARIABLE (TCO_SOURCE_ID, VARIABLE_NAME, VARIABLE_VALUE)
TSP1_UPDATE_VARIABLE_ACK ( )
TSP1_UPDATE_VARIABLE (TCO_SOURCE_ID, VARIABLE_NAME, VARIABLE_VALUE)
TSP1_UPDATE_VARIABLE_ACK ( )

Figure 27

ASK_TRACE (SESSION_ID, TEST_CASE_ID, TCO_ID, MEDIA, TRACE_ID,
FE_ERROR)

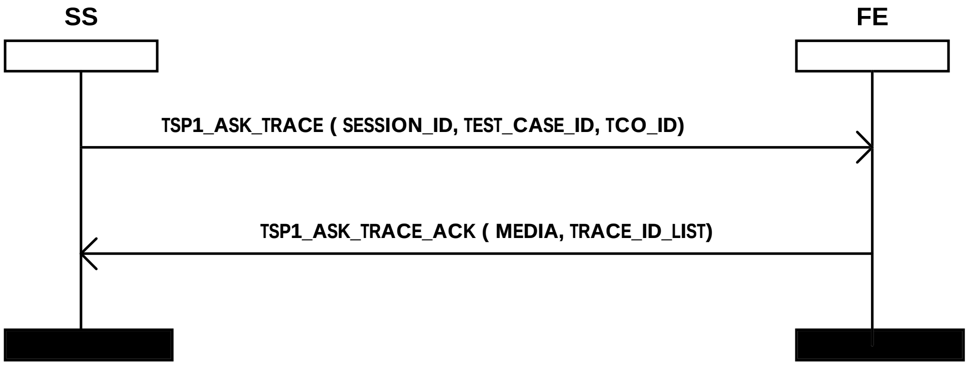

SS
FE
TSP1_ASK_TRACE (SESSION_ID, TEST_CASE_ID, TCO_ID)
TSP1_ASK_TRACE_ACK (MEDIA, TRACE_ID_LIST)

Figure 28

CLOSE_SESSION (SE_ERROR)

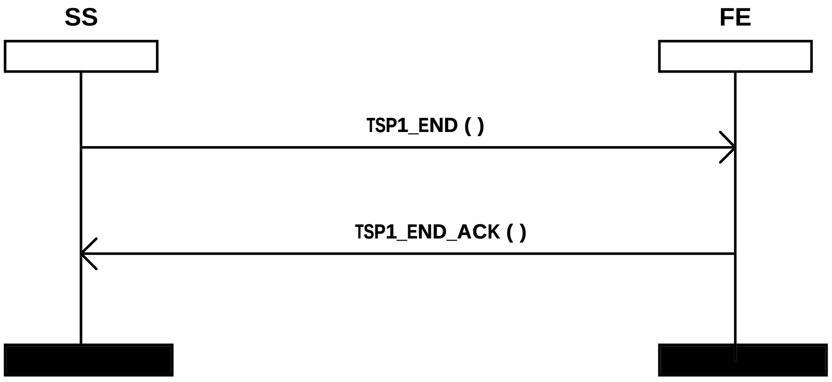


Figure 29

REJECT_MSG (FE_ID, FE_ERROR)

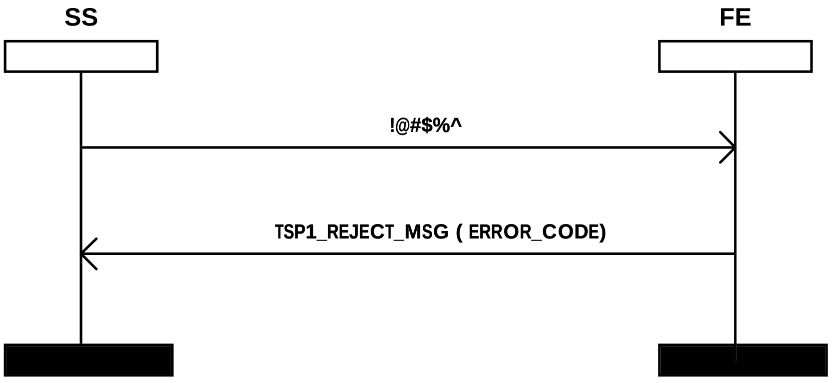


Figure 30

CANCEL_OP (TCO_ID, TCO_ERROR)

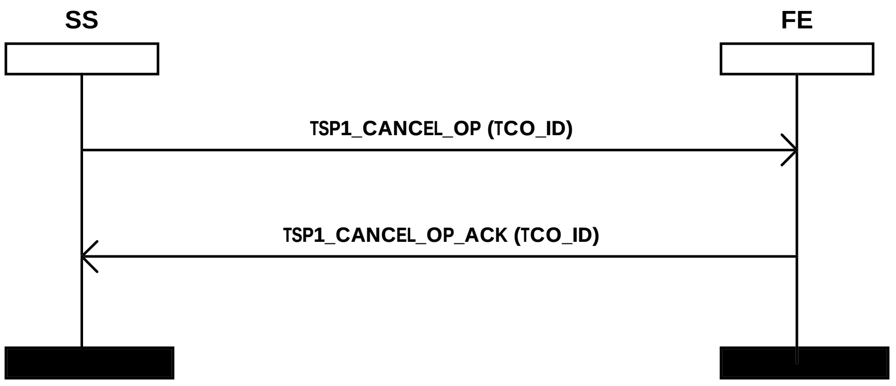


Figure 31

DISPLAY (FE_ID, MSG)

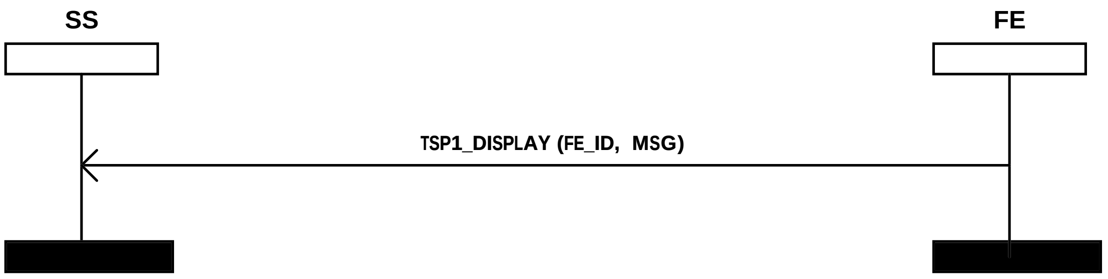


Figure 32

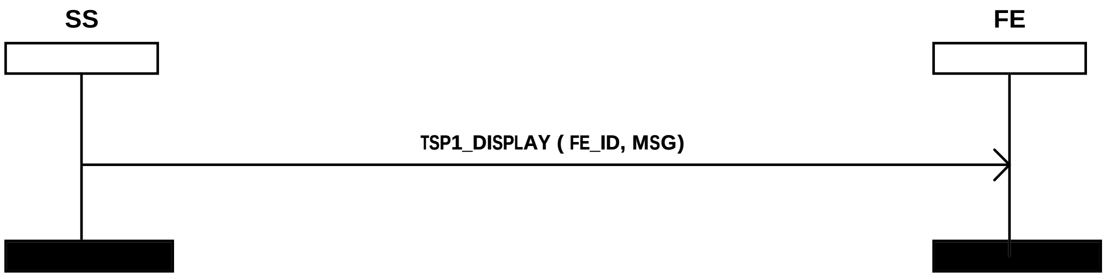


Figure 33

READ_SE_STATUS (STATUS_LIST, LOG_ID_LIST, SE_ERROR)

This service is performed without using any TSP1 messages, because the System Supervisor has already all the information concerning test components.

READ_FE_STATUS (FE_ID, STATUS_LIST, LOG_ID_LIST, FE_ERROR)

This service is performed without using any TSP1 messages, because the System Supervisor has already all the information concerning test components.

READ_TCO_STATUS (TCO_ID, STATUS, LOG_ID, TCO_ERROR)

This service is performed without using any TSP1 messages, because the System Supervisor has already all the information concerning test components.

14.1 Messages and Information Element description

In this subclause, formats and coding of TSP1 messages will be described. This is one of the possible choices that can be done. In particular this is the choice of EURESCOM P412 project.

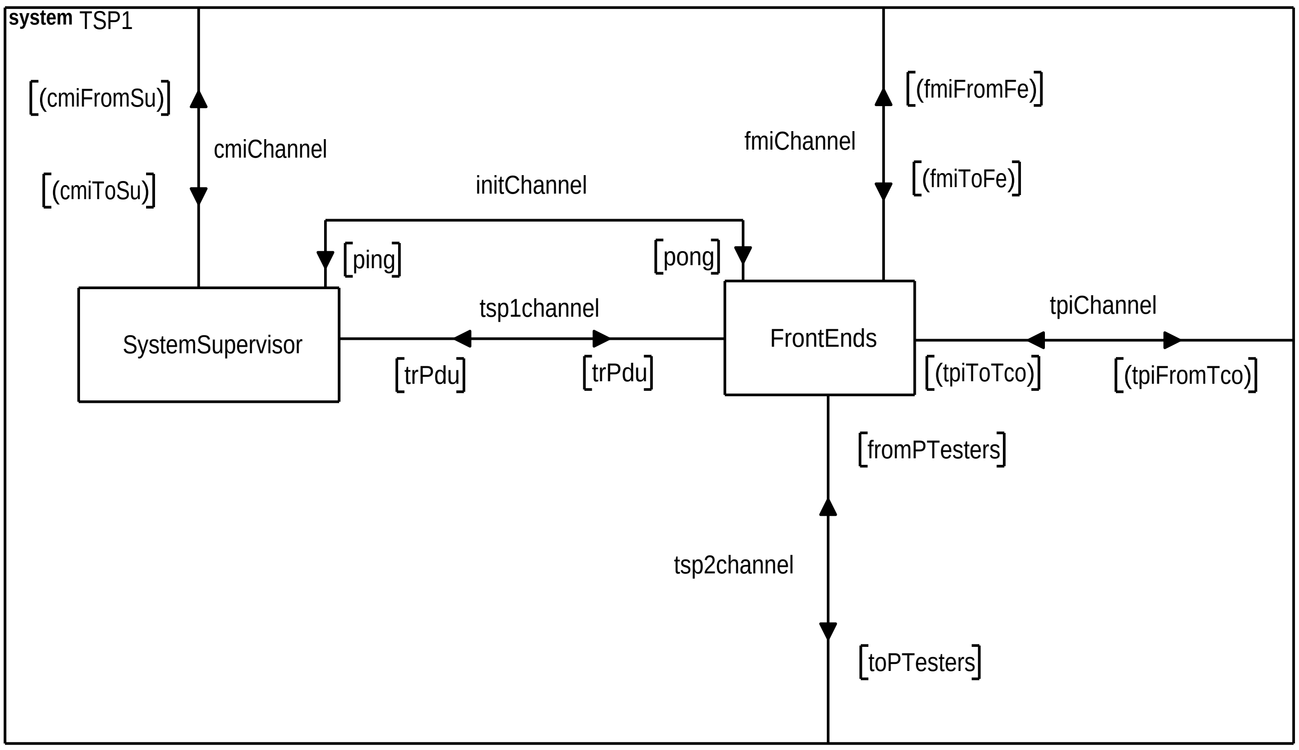
14.1.1 ASN.1 TSP1 description

```
-- TSP1 protocol

TSP1 DEFINITIONS IMPLICIT TAGS::=

BEGIN

TSP1-PDUs::= CHOICE {
        tsp1-error  ERR-Body,
        tsp1-init  INIT-Body,
        tsp1-init-ack  INIT-ACK-Body,
        tsp1-init-complete  INIT-COMPLETE-Body,
        tsp1-chk-conf  CHK-CONF-Body,
        tsp1-chk-conf-ack  CHK-CONF-ACK-Body,
        tsp1-set-parameter  SET-PARAMETER-Body,
        tsp1-set-parameter-ack  SET-PARAMETER-ACK-Body,
        tsp1-set-time  SET-TIME-Body,
        tsp1-set-time-ack  SET-TIME-ACK-Body,
        tsp1-list-fe-services  LIST-FE-SERVICES-Body,
        tsp1-list-fe-services-ack  LIST-FE-SERVICES-ACK-Body,
        tsp1-create  CREATE-Body,
        tsp1-create-ack  CREATE-ACK-Body,
        tsp1-info  INFO-Body,
        tsp1-verdict  UPDATE-VERDICT-Body,
        tsp1-update-variable  UPDATE-VARIABLE-Body,
        tsp1-update-variable-ack  UPDATE-VARIABLE-ACK-Body,
        tsp1-ask-trace  ASK-TRACE-Body,
        tsp1-ask-trace-ack  ASK-TRACE-ACK-Body,
        tsp1-end  END-Body,
        tsp1-end-ack  END-ACK-Body,
        tsp1-reject-msg  REJECT-Body,
        tsp1-cancel-op  CANCEL-OP-Body,
        tsp1-cancel-op-ack  CANCEL-OP-ACK-Body,
        tsp1-display  DISPLAY-Body
        }

-- ******************* PDU body description ***********************************
FeId::= OCTET STRING
EtsId::= OCTET STRING
TcoId::= OCTET STRING
SessionId::= OCTET STRING
TestCaseId::= OCTET STRING
TreeId::= OCTET STRING
Param::= SEQUENCE
        {
        param-id OCTET STRING,
        param-value OCTET STRING
        }
CpId::= OCTET STRING
Msg::= OCTET STRING
TraceId::= OCTET STRING
ServiceId::= INTEGER
             {
        tsp1-init              (1),
        tsp1-chk-conf          (2),
        tsp1-set-parameter     (3),
        tsp1-set-time          (4),
        tsp1-list-fe-services  (5),
        tsp1-create            (6),
        tsp1-info              (7),
        tsp1-verdict           (8),
        tsp1-update-variable   (9),
        tsp1-ask-trace         (10),
        tsp1-end               (11),
        tsp1-reject-msg        (12),
        tsp1-cancel-op         (13),
        tsp1-display           (14)
        }
ServiceList::= SEQUENCE OF ServiceId
VerdictType::= INTEGER {intermediate-verdict (0), final-verdict (1) }
VerdictValue::= INTEGER { fail (0), inconc (1), pass (2) }
MediaValue::= INTEGER
        {
        byMail      (1),
        byE-mail    (2),
        byFax       (3),
        file        (4),
        others      (5)
        }

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

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

        errTraceNotAvailable     (170),

        errCommunicationLost     (200),
        errTesterCrash           (201),
        errUnknownEtc            (255)
        }

Tsp1ErrorParam::= SEQUENCE
        {
        tco-id [0] TcoId OPTIONAL,
        errString [1] OCTET STRING
        }


ERR-Body::= [0] IMPLICIT SEQUENCE
        {
        service-id ServiceId,
        err-code ErrorCode,
        err-param Tsp1ErrorParam
        }

INIT-Body::= [1] IMPLICIT SEQUENCE
        {
        ets-id EtsId,
        session-id SessionId,
        tco-id-list SEQUENCE OF TcoId
        }

INIT-ACK-Body::= [2] IMPLICIT NULL

INIT-COMPLETE-Body::= [3] IMPLICIT NULL

CHK-CONF-Body::= [4] IMPLICIT NULL

CHK-CONF-ACK-Body::= [5] IMPLICIT NULL

SET-PARAMETER-Body::= [6] IMPLICIT SEQUENCE
        {
        param-list SEQUENCE OF Param
        }

SET-PARAMETER-ACK-Body::= [7] IMPLICIT NULL

SET-TIME-Body::= [8] IMPLICIT SEQUENCE
        {
        timestamp UTCTime
        }

SET-TIME-ACK-Body::= [9] IMPLICIT NULL

LIST-FE-SERVICES-Body::= [10] IMPLICIT NULL

LIST-FE-SERVICES-ACK-Body::= [11] IMPLICIT SEQUENCE
        {
        fe-service-list ServiceList
        }

CREATE-Body::= [12] IMPLICIT SEQUENCE
        {
        tco-id [0] TcoId OPTIONAL,
        test-case-id [1] TestCaseId,
        tree TreeId OPTIONAL,
        param-list SEQUENCE OF Param OPTIONAL
        }

CREATE-ACK-Body::= [13] IMPLICIT SEQUENCE
        {
        tco-id TcoId
        }

INFO-Body::= [14] IMPLICIT SEQUENCE
        {
        dest-id TcoId,
        cp-id CpId,
        coord-msg Msg
        }

UPDATE-VERDICT-Body::= [15] IMPLICIT SEQUENCE
        {
        tco-id TcoId,
        tco-verdict-type VerdictType,
        tco-verdict-value VerdictValue,
        error-code ErrorCode
        }

UPDATE-VARIABLE-Body::= [16] IMPLICIT SEQUENCE
        {
        tco-source-id TcoId,
        variable-name OCTET STRING,
        variable-value OCTET STRING
        }

UPDATE-VARIABLE-ACK-Body::= [17] IMPLICIT NULL

ASK-TRACE-Body::= [18] IMPLICIT SEQUENCE
        {
        session-id SessionId,
        test-case-id TestCaseId,
        tco-id TcoId
        }

ASK-TRACE-ACK-Body::= [19] IMPLICIT SEQUENCE
        {
        media MediaValue,
        trace-id-list SEQUENCE OF TraceId
        }

END-Body::= [20] IMPLICIT NULL

END-ACK-Body::= [21] IMPLICIT NULL

REJECT-Body::= [22] IMPLICIT SEQUENCE
        {
        error-code ErrorCode
        }

CANCEL-OP-Body::= [23] IMPLICIT SEQUENCE
        {
        tco-id TcoId OPTIONAL
        }

CANCEL-OP-ACK-Body::= [24] IMPLICIT SEQUENCE
        {
        tco-id TcoId OPTIONAL
        }

DISPLAY-Body::= [25] IMPLICIT SEQUENCE
        {
        fe-id FeId,
        disp-msg Msg
        }

END
```

14.2 SDL TSP1 description

TSP1 is described in the SDL diagrams which follow. (The SDL was produced in the context of Eurescom project p.412).

TSP1 v1.0	EURESCOM P412 Wed Jun 19 1996	View: 1



EURESCOM	tsp1_1_0.pr	Page: 1

```
NEWTYPE TspTag LITERALS
 tspErrorTag,              /*  0 */
 tspInitTag,               /*  1 */
 tspInitAckTag,            /*  2 */
 tspInitCompleteTag,       /*  3 */
 tspChkConfigTag,          /*  4 */
 tspChkConfigAckTag,       /*  5 */
 tspSetParameterTag,       /*  6 */
 tspSetParameterAckTag,    /*  7 */
 tspSetTimeTag,            /*  8 */
 tspSetTimeAckTag,         /*  9 */
 tspListServicesTag,       /* 10 */
 tspListServicesAckTag,    /* 11 */
 tspCreateTag,             /* 12 */
 tspCreateAckTag,          /* 13 */
 tspInfoTag,               /* 14 */
 tspVerdictTag,            /* 15 */
 tspUpdateVariableTag,     /* 16 */
 tspUpdateVariableAckTag,  /* 17 */
 tspAskTraceTag,           /* 18 */
 tspAskTraceAckTag,        /* 19 */
 tspEndTag,                /* 20 */
 tspEndAckTag,             /* 21 */
 tspRejectMsgTag,          /* 22 */
 tspCancelOpTag,           /* 23 */
 tspCancelOpAckTag,        /* 24 */
 tspDisplayTag             /* 25 */
ENDNEWTYPE TspTag;

NEWTYPE ServiceId LITERALS
 tspInitAsp,            /*  1 */
 tspChkConfigAsp,       /*  2 */
 tspSetParameterAsp,    /*  3 */
 tspSetTimeAsp,         /*  4 */
 tspListFeServicesAsp,  /*  5 */
 tspCreateAsp,          /*  6 */
 tspInfoAsp,            /*  7 */
 tspVerdictAsp,         /*  8 */
 tspUdpdateVariableAsp,/*  9 */
 tspAskTraceAsp,        /* 10 */
 tspEndAsp,             /* 11 */
 tspRejectMsgAsp,       /* 12 */
 tspCancelOpAsp,        /* 13 */
 tspDisplayAsp          /* 14 */
ENDNEWTYPE ServiceId;

NEWTYPE TspPdu
STRUCT
 tag tspTag;
 tspError          TspErrorPdu;
 tspInit           TspInitPdu;
 tspSetParameter   TspSetParameterPdu;
 tspSetTime        TspSetTimePdu;
 tspListServices   TspListServicesAckPdu;
 tspCreate         TspCreatePdu;
 tspCreateAck      TspCreateAckPdu;
 tspInfo           TspInfoPdu;
 tspVerdict        TspVerdictPdu;
 tspUpdateVariable TspUpdateVariablePdu;
 tspAskTrace       TspAskTracePdu;
 tspAskTraceAck    TspAskTraceAckPdu;
 tspRejectMsg      TspRejectMsgPdu;
 tspCancelOp       TspCancelOpPdu;
 tspCancelOpAck    TspCancelOpAckPdu;
 tspDisplay        TspDisplayPdu;
ENDNEWTYPE TspPdu;
```

```
/* Supervisor
   Primitives */

SIGNALLIST TspSuReq  =
 TspInitReq,
 TspChkConfigReq,
 TspSetParameterReq,
 TspSetTimeReq,
 TspCreateReq,
 TspInfoReq,
 TspVerdictReq,
 TspUpdateVariableReq,
 TspEndReq,
 TspCancelOpReq;

SIGNALLIST TspSuComReq =
 TspDisplayReq,
 TspAskTraceReq,
 TspListServicesReq;

SIGNALLIST TspSuInd  =
 TspErrorInd,
 TspInitCompleteInd,
 TspCreateInd,
 TspInfoInd,
 TspVerdictInd,
 TspUpdateVariableInd,
 TspRejectMsgInd;

SIGNALLIST TspSuComInd =
 TspDisplayInd;

SIGNALLIST TspSuResp =
 TspUpdateVariableResp;

SIGNALLIST TspSuConf =
 TspInitConf,
 TspChkConfigConf,
 TspSetParameterConf,
 TspSetTimeConf,
 TspCreateConf,
 TspUpdateVariableConf,
 TspEndConf,
 TspCancelOpConf;

SIGNALLIST TspSuComConf =
 TspAskTraceConf,
 TspListServicesConf;
```

```
/* FrontEnd
   Primitives */

SIGNALLIST TspFeInd  =
 TspInitInd,
 TspChkConfigInd,
 TspSetParameterInd,
 TspSetTimeInd,
 TspCreateInd,
 TspInfoInd,
 TspVerdictInd,
 TspUpdateVariableInd,
 TspEndInd,
 TspCancelOpInd;

SIGNALLIST TspFeComInd =
 TspDisplayInd,
 TspAskTraceInd,
 TspListServicesInd;

SIGNALLIST TspFeReq  =
 TspErrorReq,
 TspInitCompleteReq,
 TspCreateReq,
 TspInfoReq,
 TspVerdictReq,
 TspUpdateVariableReq,
 TspRejectMsgReq;

SIGNALLIST TspFeComReq =
 TspDisplayReq;

SIGNALLIST TspFeConf =
 TspUpdateVariableConf;

SIGNALLIST TspFeResp =
 TspInitResp,
 TspChkConfigResp,
 TspSetParameterResp,
 TspSetTimeResp,
 TspCreateResp,
 TspUpdateVariableResp,
 TspEndResp,
 TspCancelOpResp;

SIGNALLIST TspFeComResp =
 TspAskTraceResp,
 TspListServicesResp;
```

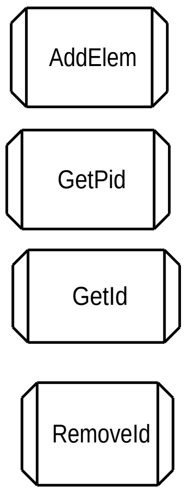
```
/* ASN1-like type declaration */

SYNTYPE FeId = Charstring ENDSYNTYPE;

SYNTYPE EtsId = Charstring ENDSYNTYPE;

SYNTYPE SessionId = Charstring ENDSYNTYPE;

SYNTYPE TestCaseId = Charstring ENDSYNTYPE;

SYNTYPE TcoId = Charstring ENDSYNTYPE;

NEWTYPE TcoIdList String(TcoId,noTco) ENDNEWTYPE;

SYNTYPE TreeId = Charstring ENDSYNTYPE;

NEWTYPE ServiceList String(ServiceId,noService) ENDNEWTYPE;

SYNTYPE UTCTime = Time ENDSYNTYPE;

NEWTYPE MediaValue LITERALS
 byMail,
 byEmail,
 byFax,
 file,
 others
ENDNEWTYPE MediaValue;

SYNTYPE VariableName = CharString ENDSYNTYPE;
SYNTYPE VariableValue = Charstring ENDSYNTYPE;

NEWTYPE VerdictType LITERALS
 intermediateVerdict,
 finalVerdict
ENDNEWTYPE;

NEWTYPE VerdictValue LITERALS
 fail,
 inconc,
 pass
ENDNEWTYPE;

SYNTYPE CpId = Charstring ENDSYNTYPE;


NEWTYPE Param
STRUCT
 paramId    Charstring;
 paramValue Charstring;
ENDNEWTYPE Param;

NEWTYPE ParamList String(Param,noParam) ENDNEWTYPE;

SYNTYPE TraceId = Charstring ENDSYNTYPE;

NEWTYPE TraceIdList String(TraceId, noTrace) ENDNEWTYPE;
```

```
/* Test Synchronization Protocol */

NEWTYPE TspErrorParam
STRUCT
 tcoId     TcoId;
 errString Charstring;
ENDNEWTYPE TspErrorParam;

SYNONYM NullError = (.’’,’’.);

NEWTYPE TspErrorPdu
STRUCT
 serviceId ServiceId;
 errCode   ErrorCode;
 errParam  TspErrorParam;
ENDNEWTYPE TspErrorPdu;

NEWTYPE TspInitPdu
STRUCT
 etsId EtsId;
 sessionId SessionId;
 tcoIdList TcoIdList;
ENDNEWTYPE TspInitPdu;

NEWTYPE TspSetParameterPdu
STRUCT
 paramList ParamList;
ENDNEWTYPE TspSetParameterPdu;

NEWTYPE TspSetTimePdu
STRUCT
 timeStamp UTCTime;
ENDNEWTYPE TspSetTimePdu;

NEWTYPE TspListServicesAckPdu
STRUCT
 feServices ServiceList;
ENDNEWTYPE TspListServicesAckPdu;

NEWTYPE TspCreatePdu
STRUCT
 tcoId      TcoId;
 testCaseId TestCaseId;
 tree       TreeId;
 paramList  ParamList;
ENDNEWTYPE TspCreatePdu;

NEWTYPE TspCreateAckPdu
STRUCT
 tcoId TcoId;
ENDNEWTYPE TspCreateAckPdu;

NEWTYPE TspInfoPdu
STRUCT
 destId    TcoId;
 cpId     CpId;
 coordMsg Msg;
ENDNEWTYPE TspInfoPdu;

NEWTYPE TspVerdictPdu
STRUCT
 tcoId        TcoId;
 verdictType  VerdictType;
 verdictValue VerdictValue;
 errorCode    ErrorCode;
ENDNEWTYPE TspVerdictPdu;
```

```
NEWTYPE TspUpdateVariablePdu
STRUCT
 tcoSourceId   TcoId;
 variableName  VariableName;
 variableValue VariableValue;
ENDNEWTYPE TspUpdateVariablePdu;

NEWTYPE TspAskTracePdu
STRUCT
 sessionId  SessionId;
 testCaseId TestCaseId;
 tcoId      TcoId;
ENDNEWTYPE TspAskTracePdu;

NEWTYPE TspAskTraceAckPdu
STRUCT
 media MediaValue;
 traceIdList TraceIdList;
ENDNEWTYPE TspAskTraceAckPdu;

NEWTYPE TspRejectMsgPdu
STRUCT
 errorCode ErrorCode;
ENDNEWTYPE TspRejectMsgPdu;

NEWTYPE TspCancelOpPdu
STRUCT
 tcoId TcoId;
ENDNEWTYPE TspCancelOpPdu;

NEWTYPE TspCancelOpAckPdu
STRUCT
 tcoId TcoId;
ENDNEWTYPE TspCancelOpAckPdu;

NEWTYPE TspDisplayPdu
STRUCT
 feId    FeId;
 dispMsg Msg;
ENDNEWTYPE TspDisplayPdu;
```

```
SIGNAL TrPdu(TspPdu);

SIGNAL TspErrorReq(TspErrorPdu),                   TspErrorInd(TspErrorPdu);

SIGNAL TspInitReq(TspInitPdu),                     TspInitInd(TspInitPdu),
       TspInitResp,                                TspInitConf;

SIGNAL TspInitCompleteReq,                         TspInitCompleteInd;

SIGNAL TspChkConfigReq,                            TspChkConfigInd,
       TspChkConfigResp,                           TspChkConfigConf;

SIGNAL TspSetParameterReq(TspSetParameterPdu),     TspSetParameterInd(TspSetParameterPdu),
       TspSetParameterResp,                        TspSetParameterConf;

SIGNAL TspSetTimeReq(TspSetTimePdu),               TspSetTimeInd(TspSetTimePdu),
       TspSetTimeResp,                             TspSetTimeConf;

SIGNAL TspListServicesReq,                         TspListServicesInd,
       TspListServicesResp(TspListServicesAckPdu), TspListServicesConf(TspListServicesAckPdu);

SIGNAL TspCreateReq(TspCreatePdu),                 TspCreateInd(TspCreatePdu),
       TspCreateResp(TspCreateAckPdu),              TspCreateConf(TspCreateAckPdu);

SIGNAL TspInfoReq(TspInfoPdu),                     TspInfoInd(TspInfoPdu);

SIGNAL TspVerdictReq(TspVerdictPdu),               TspVerdictInd(TspVerdictPdu);

SIGNAL TspUpdateVariableReq(TspUpdateVariablePdu), TspUpdateVariableInd(TspUpdateVariablePdu),
       TspUpdateVariableResp,                      TspUpdateVariableConf;

SIGNAL TspAskTraceReq(TspAskTracePdu),             TspAskTraceInd(TspAskTracePdu),
       TspAskTraceResp(TspAskTraceAckPdu),         TspAskTraceConf(TspAskTraceAckPdu);

SIGNAL TspEndReq,                                  TspEndInd,
       TspEndResp,                                 TspEndConf;

SIGNAL TspRejectMsgReq(TspRejectMsgPdu),           TspRejectMsgInd(TspRejectMsgPdu);

SIGNAL TspCancelOpReq(TspCancelOpPdu),             TspCancelOpInd(TspCancelOpPdu),
       TspCancelOpResp(TspCancelOpAckPdu),         TspCancelOpConf(TspCancelOpAckPdu);

SIGNAL TspDisplayReq(TspDisplayPdu),               TspDisplayInd(TspDisplayPdu);
```

```
SYNONYM nFE = 3;
SYNONYM nTCo = 10;

SYNONYM testTime = 120;
SYNONYM defaultTime = 10;

SIGNAL fromPTesters, toPTesters, ping, pong(FeId),
       startCommonServices, stopCommonServices,
       killCommonServices, killDecoder, killEncoder;

SYNTYPE Index = Natural CONSTANTS >0 ENDSYNTYPE;

NEWTYPE RoutingElement STRUCT
  id   Charstring;
  pid  Pid;
ENDNEWTYPE;

NEWTYPE RoutingList
  String(RoutingElement,empty)
ENDNEWTYPE;

SYNONYM initAckTime = defaultTime;
SYNONYM initCompleteTime = defaultTime;
SYNONYM endAckTime = defaultTime;
SYNONYM chkAckTime = defaultTime;
SYNONYM paramAckTime = defaultTime;
SYNONYM timeAckTime = defaultTime;
SYNONYM createAckTime = defaultTime;
SYNONYM cancelAckTime = defaultTime;
SYNONYM listServicesAckTime = defaultTime;
SYNONYM askTraceAckTime = defaultTime;
```



```
SYNTYPE Msg = Charstring ENDSYNTYPE;

SYNTYPE Tree = Charstring ENDSYNTYPE;
NEWTYPE ErrorCode LITERALS errOK, errTimeOut, errPT, errBadSe, errBadTco ENDNEWTYPE;
/* ---------- Front-end Synchronizing Interface’s Types ---------- */
NEWTYPE StErrorCode LITERALS errStOK, errStFe, errStArg, errStTimeOut ENDNEWTYPE;
NEWTYPE FeError STRUCT
   feId        FeId;
   stubError   StErrorCode;
   feError     ErrorCode;
   retryNumber Natural;
ENDNEWTYPE;
/* ---------- Campaign Management Interface’s Types ---------- */

NEWTYPE FeErrorList String(FeError,empty) ENDNEWTYPE;

NEWTYPE SeErrorCode LITERALS errSeOK, errSeFailed ENDNEWTYPE;
NEWTYPE SeError STRUCT
   seId     SessionId;
   seError  SeErrorCode;
   feErrors FeErrorList;
ENDNEWTYPE;

SYNTYPE MtcId = TcoId ENDSYNTYPE;

NEWTYPE TestComponent STRUCT
   id TcoId;
   feId  FeId;
ENDNEWTYPE;
NEWTYPE TcoList String(TestComponent,noTco) ENDNEWTYPE;

NEWTYPE FeIdSet PowerSET(FeId) ENDNEWTYPE;

NEWTYPE TcoErrorCode LITERALS errTcoOK, errTcoNotStarted, errTcoTimeOut, errTcoFe
ENDNEWTYPE;
NEWTYPE TcoError STRUCT
   tcoId TcoId;
   tcoError TcoErrorCode;
   feError FeError;
ENDNEWTYPE;
NEWTYPE TcoErrorList String(TcoError,empty) ENDNEWTYPE;
NEWTYPE EtcErrorCode LITERALS errEtcOK, errEtcError, errEtcTimeOut ENDNEWTYPE;
NEWTYPE EtcError STRUCT
   etcId TestCaseId;
   etcError  EtcErrorCode;
   tcoErrors TcoErrorList;
ENDNEWTYPE;
SYNTYPE TcoIndex = Natural CONSTANTS 1 : nTCo ENDSYNTYPE;
```

```
/* Campaign Magagement Interface (cmi) */

SIGNALLIST cmiToSu = (cmiReqToSu), (cmiCommonReqToSu);
SIGNALLIST cmiCommonReqToSu =
   cmiDisplay, cmiListServices, cmiAskTrace;
SIGNALLIST cmiReqToSu =
   cmiLoadETS, cmiOpenSession, cmiCloseSession, cmiCheckSession, cmiSetParameter,
   cmiSetTime, cmiUnloadETS, cmiExecETC, cmiCancelExec;
SIGNALLIST cmiFromSu = (cmiConfFromSu), (cmiCommonConfFromSu);
SIGNALLIST cmiCommonConfFromSu =
   cmiDisplay, cmiListedServices, cmiAskedTrace;
SIGNALLIST cmiConfFromSu = cmiAllFeConnected,
   cmiOpenedSession, cmiClosedSession, cmiCheckedSession, cmiSentParameter,
   cmiSentTime, cmiEtcEnd, cmiExecutedETC, cmiTcoVerdict, cmiEtcVerdict;

SIGNAL cmiLoadETS(EtsId);
SIGNAL cmiOpenSession(SessionId,TcoList,MtcId);
SIGNAL cmiCloseSession;
SIGNAL cmiDisplay(FeId,Msg);
SIGNAL cmiCheckSession;
SIGNAL cmiSetParameter(ParamList);
SIGNAL cmiSetTime(TIME);
SIGNAL cmiUnloadETS;
SIGNAL cmiExecETC(TestCaseId);
SIGNAL cmiCancelExec;
SIGNAL cmiAskTrace(FeId,TspAskTracePdu);
SIGNAL cmiListServices(FeId);

SIGNAL cmiAllFeConnected(SeError);
SIGNAL cmiOpenedSession(SeError);
SIGNAL cmiClosedSession(SeError);
SIGNAL cmiCheckedSession(SeError);
SIGNAL cmiSentParameter(SeError);
SIGNAL cmiSentTime(SeError);
SIGNAL cmiExecutedETC(EtcError);
SIGNAL cmiTcoVerdict(TcoId,VerdictValue);
SIGNAL cmiEtcEnd(EtcError);
SIGNAL cmiEtcVerdict(VerdictValue,EtcError);
SIGNAL cmiAskedTrace(TspAskTraceAckPdu,FeError);
SIGNAL cmiListedServices(TspListServicesAckPdu,FeError);
```

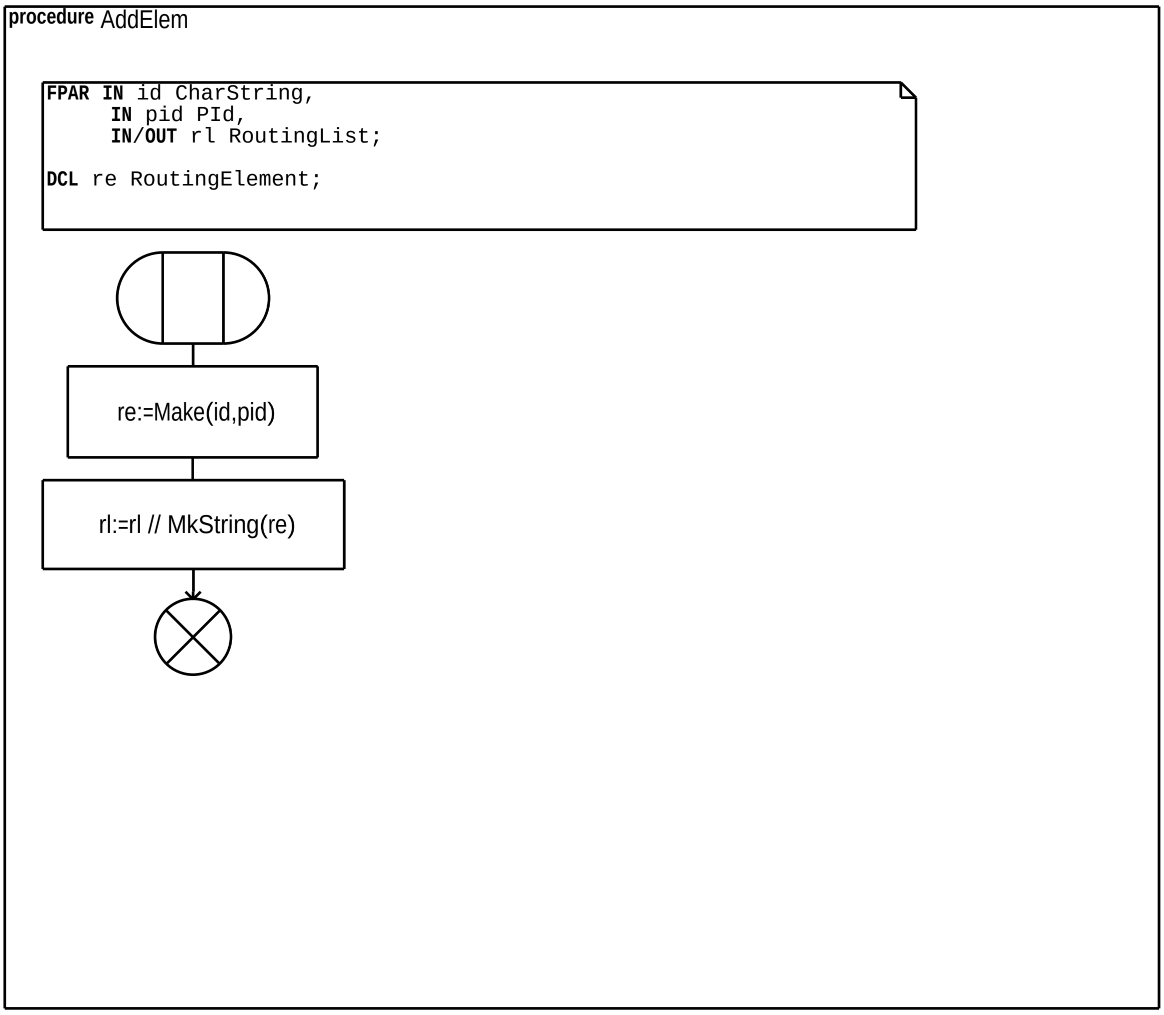
```
/* Test Programming Interface (tpi) */

SIGNALLIST tpiToTco = tpiCreate, tpiCreated, tpiCanceled, tpiUpdateVerdict,
                      tpiFinalVerdict, tpiInfoReq;
SIGNALLIST tpiFromTco = tpiCreate, tpiCancel, tpiUpdatedVerdict, tpiLastVerdict, tpiInfoInd;

SIGNAL tpiCreated(ErrorCode);
SIGNAL tpiCreate(TcoId,TestCaseId,Tree,ParamList);
SIGNAL tpiCancel;
SIGNAL tpiCanceled(ErrorCode);
SIGNAL tpiFinalVerdict(VerdictValue,ErrorCode);
SIGNAL tpiUpdateVerdict(VerdictValue);
SIGNAL tpiUpdatedVerdict(TcoId,VerdictValue);
SIGNAL tpiLastVerdict(TcoId,VerdictValue,ErrorCode);
SIGNAL tpiInfoReq(TcoId,CpId,Msg);
SIGNAL tpiInfoInd(CpId,Msg);
```

```
/* Test Managing Interface (tmi) */

SIGNALLIST tmiToSu = tmiEnd, tmiStopped, tmiLocalVerdict, tmiVerdict;
SIGNALLIST tmiToTco = tmiStart, tmiStop, tmiKill;

SIGNAL tmiStart(TestCaseId,ParamList);
SIGNAL tmiStop;
SIGNAL tmiKill;
SIGNAL tmiEnd(TcoError);
SIGNAL tmiVerdict(VerdictValue,TcoError);
SIGNAL tmiLocalVerdict(TcoId,VerdictValue);
SIGNAL tmiStopped(TcoError);
```

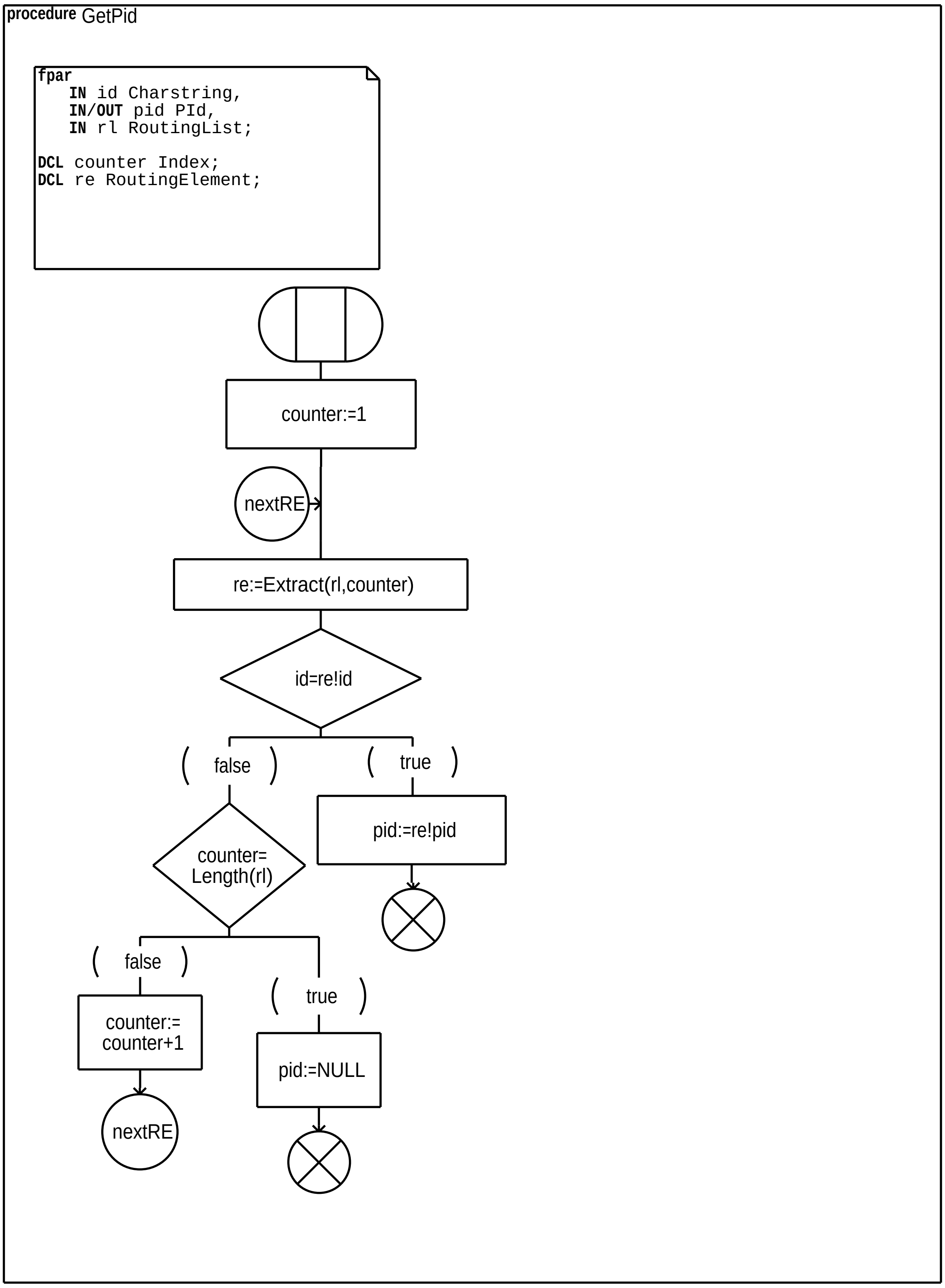
```
/* Test Synchronizing Interface (tsi) */

SIGNALLIST tsiToTco = tsiCanceled, tsiCreated, tsiCancel, tsiUpdateVerdict, tsiFinalVerdict,
                      tsiCreate, tsiKill, tsiInfo;
SIGNALLIST tsiToFe = tsiCreate, tsiCancel, tsiCanceled, tsiCreated, tsiFinalVerdict,
                     tsiUpdateVerdict, tsiInfo;

SIGNAL tsiCreate(TcoId,TestCaseId,Tree,ParamList);
SIGNAL tsiCancel(TcoId);
SIGNAL tsiCanceled(FeError);
SIGNAL tsiCreated(FeError);
SIGNAL tsiFinalVerdict(TcoId,VerdictValue,FeError);
SIGNAL tsiUpdateVerdict(TcoId,VerdictValue,FeError);
SIGNAL tsiKill;
SIGNAL tsiInfo(TcoId,CpId,Msg);
```

```
/* Front-end Management Interface (fmi) */

SIGNALLIST fmiFromFe = fmiDisplay;
SIGNALLIST fmiToFe = fmiDisplay;

SIGNAL fmiDisplay(FeId,Msg);
```

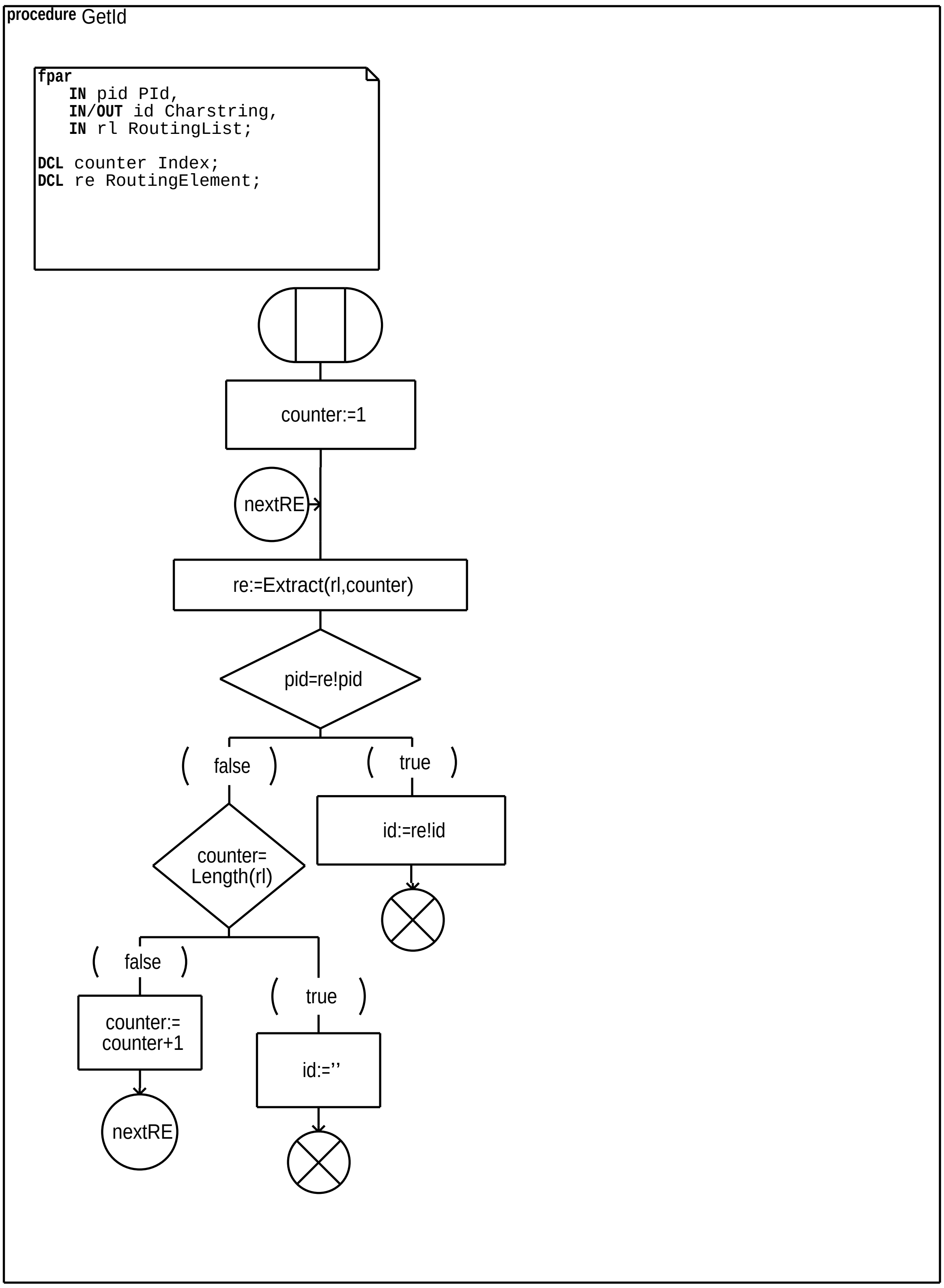
```
/* Front-end Synchronizing Interface (fsi) */

SIGNALLIST fsiToSu = (fsiCommonConf), (fsiConfirm), fsiDisconnected;
SIGNALLIST fsiCommonConf = fsiToBeDisplayed, fsiAskedTrace, fsiListedServices;
SIGNALLIST fsiConfirm =
     fsiConnected, fsiOpenedSession, fsiClosedSession, fsiCheckedSession,
     fsiSentParameter, fsiSentTime;

SIGNALLIST fsiToSt = (fsiCommonReq), (fsiRequest);
SIGNALLIST fsiCommonReq = fsiDisplay, fsiAskTrace, fsiListServices;
SIGNALLIST fsiRequest =
     fsiConnect, fsiOpenSession, fsiCloseSession,  fsiCheckSession, fsiSetParameter,
     fsiSetTime, fsiDisconnect, fsiDeclareTco, fsiEndTest;

SIGNAL fsiConnect;
SIGNAL fsiConnected(FeError);
SIGNAL fsiOpenedSession(FeError);
SIGNAL fsiCloseSession;
SIGNAL fsiToBeDisplayed(FeId,FeId,Msg);
SIGNAL fsiClosedSession(FeError);
SIGNAL fsiEndTest;

SIGNAL fsiDisplay(FeId,Msg);

SIGNAL fsiOpenSession;
SIGNAL fsiCheckSession;
SIGNAL fsiCheckedSession(FeError);
SIGNAL fsiSetParameter;
SIGNAL fsiSentParameter(FeError);
SIGNAL fsiSetTime;
SIGNAL fsiSentTime(FeError);

SIGNAL fsiDisconnect;
SIGNAL fsiDisconnected(FeError);
SIGNAL fsiDeclareTco(TcoId);

SIGNAL fsiListServices;
SIGNAL fsiAskTrace(TspAskTracePdu);
SIGNAL fsiListedServices(TspListServicesAckPdu,FeError);
SIGNAL fsiAskedTrace(TspAskTraceAckPdu,FeError);
```



```
FPAR IN id CharString,
     IN pid PId,
     IN/OUT rl RoutingList;

DCL re RoutingElement;
```



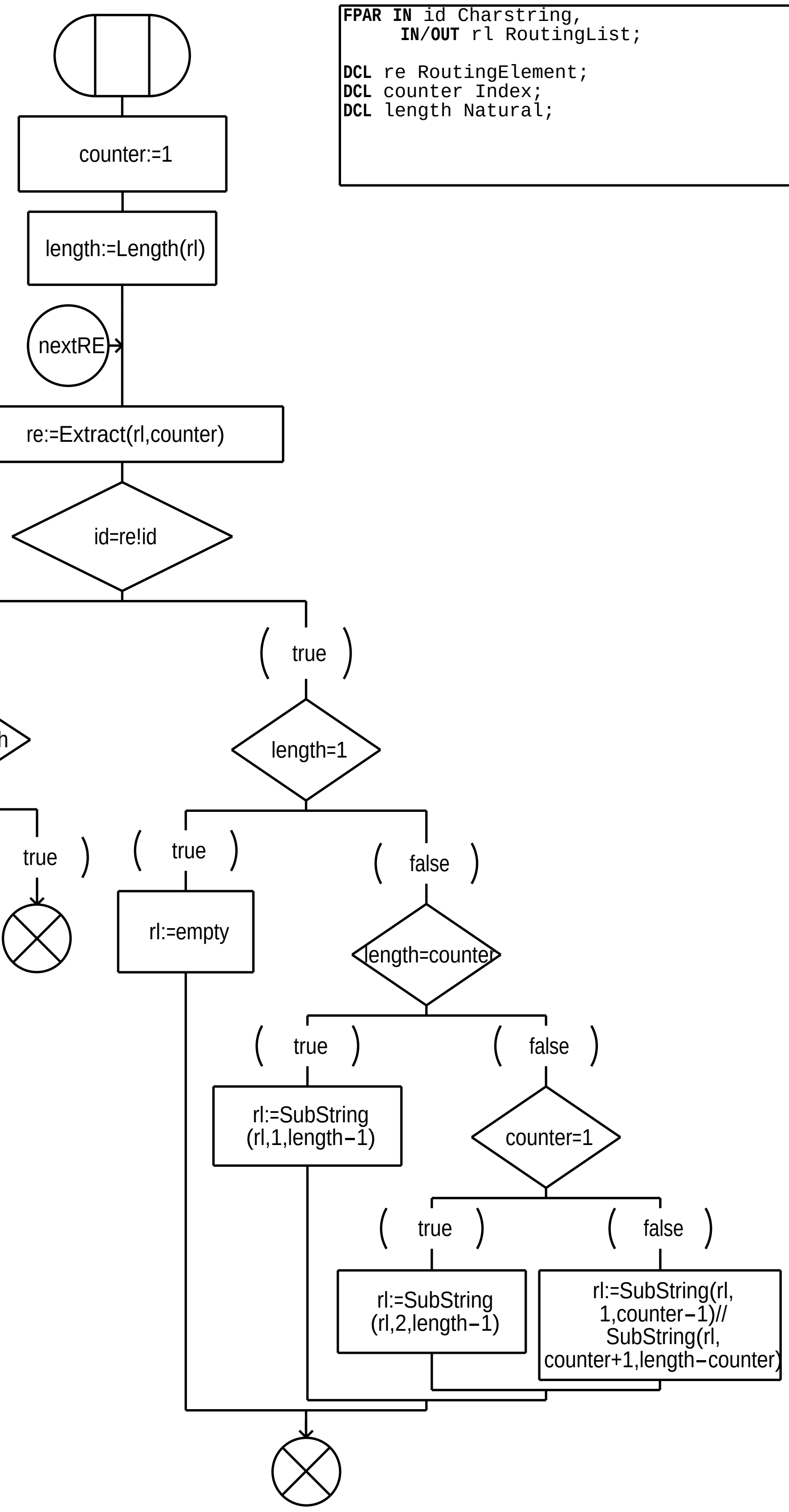
```
fpar
   IN id Charstring,
   IN/OUT pid PId,
   IN rl RoutingList;

DCL counter Index;
DCL re RoutingElement;
```

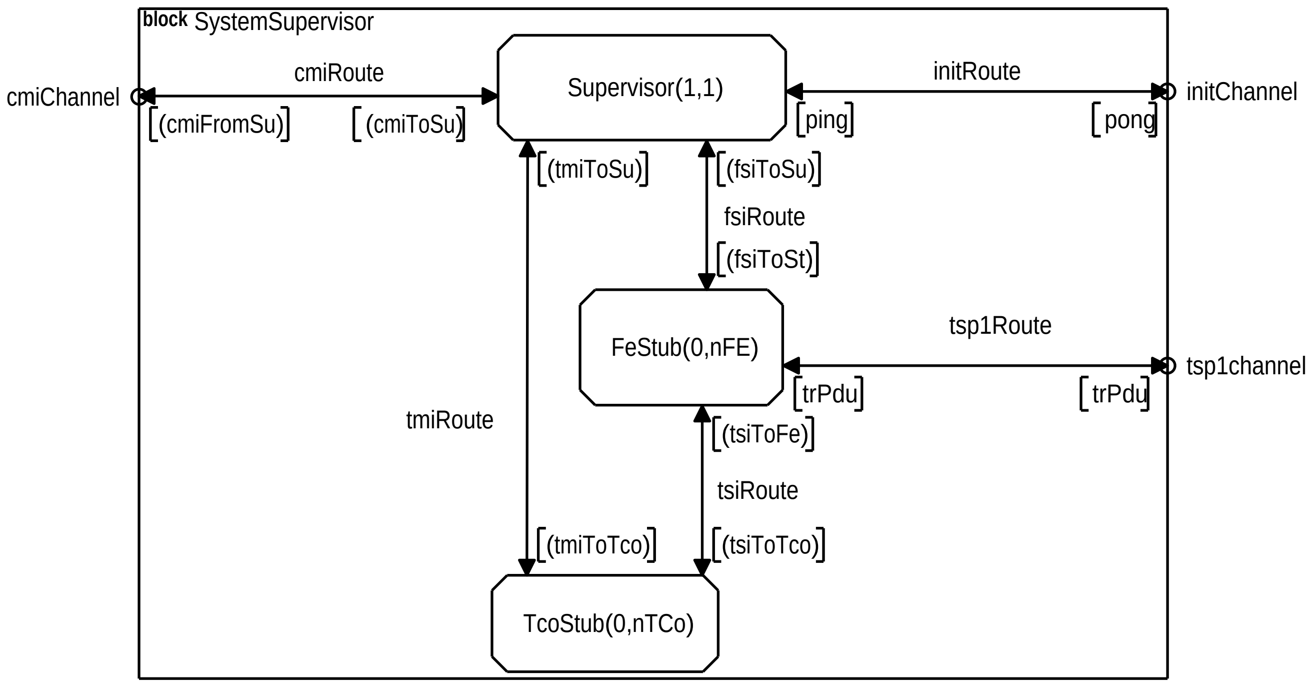


```
fpar
    IN pid PId,
    IN/OUT id Charstring,
    IN rl RoutingList;

DCL counter Index;
DCL re RoutingElement;
```

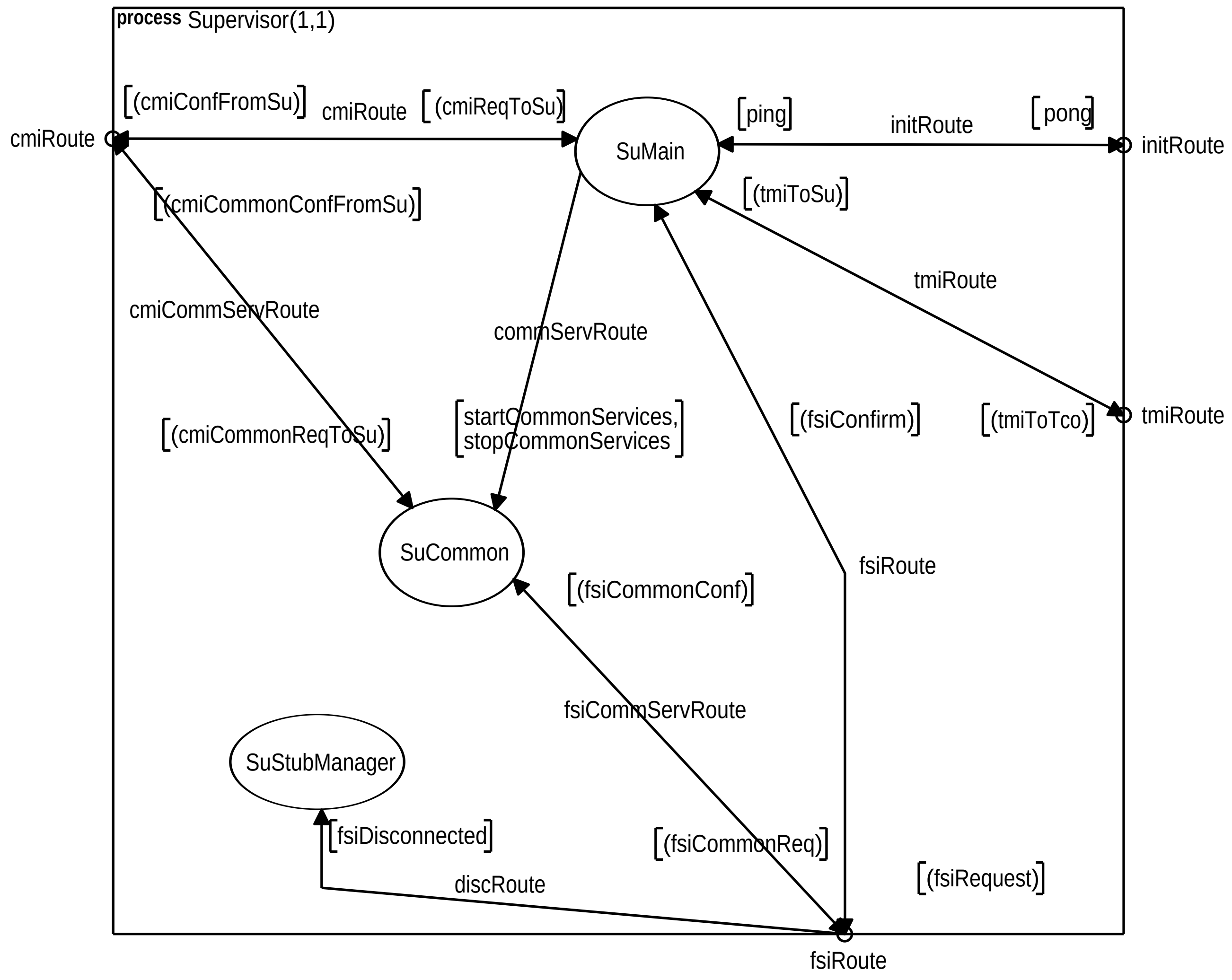
procedure RemoveId

```
FPAR IN id Charstring,
     IN/OUT rl RoutingList;

DCL re RoutingElement;
DCL counter Index;
DCL length Natural;
```



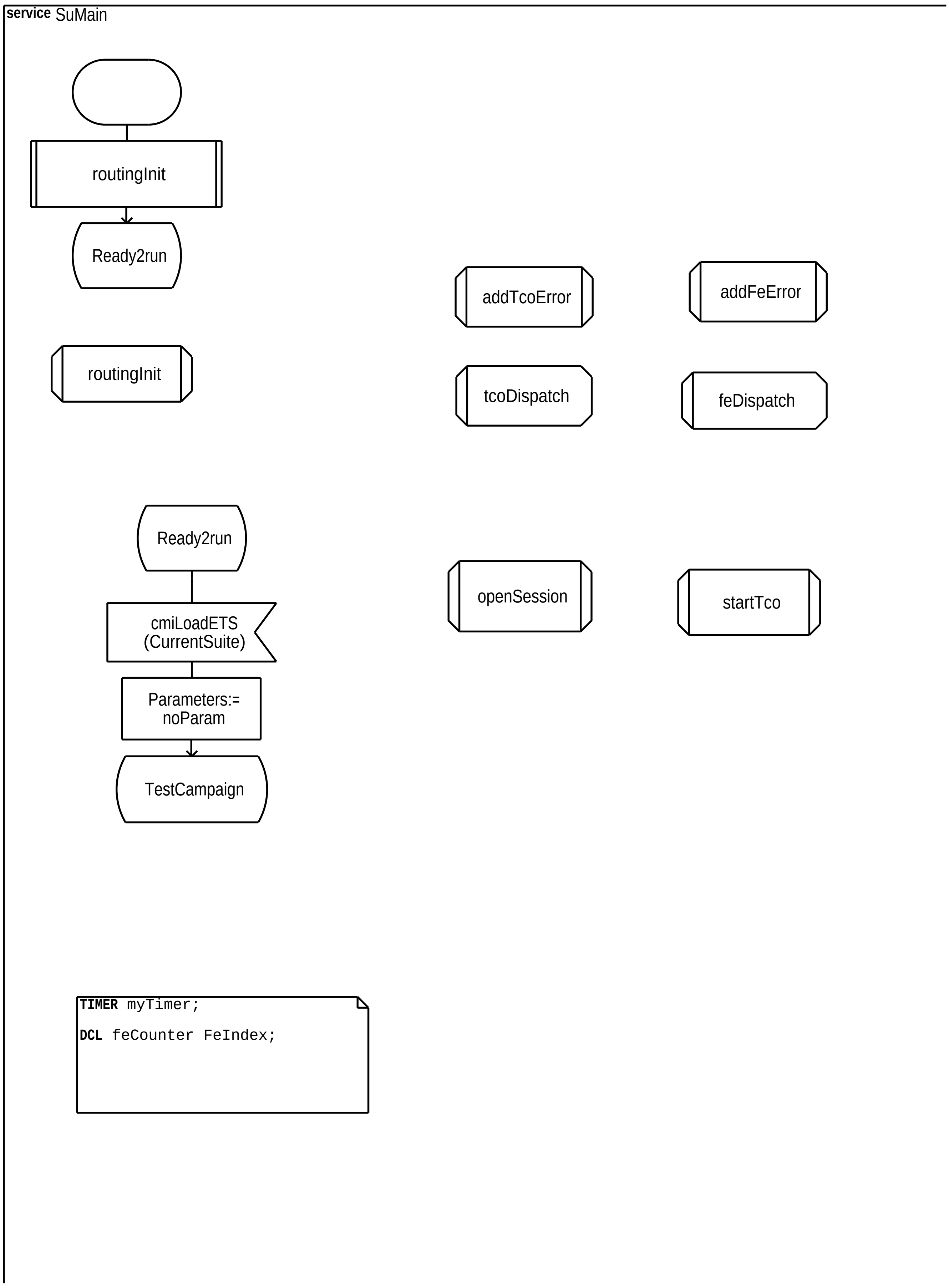


```
SYNTYPE FeIndex = Natural CONSTANTS 1 : nFE ENDSYNTYPE;

NEWTYPE FeIdList String(FeId,empty) ENDNEWTYPE;

NEWTYPE FrontEnd
STRUCT
 id   FeId;
 pid  PId;
 stub PId;
ENDNEWTYPE FrontEnd;

NEWTYPE RoutingTable
   array(FeIndex,FrontEnd);
ENDNEWTYPE RoutingTable;
```

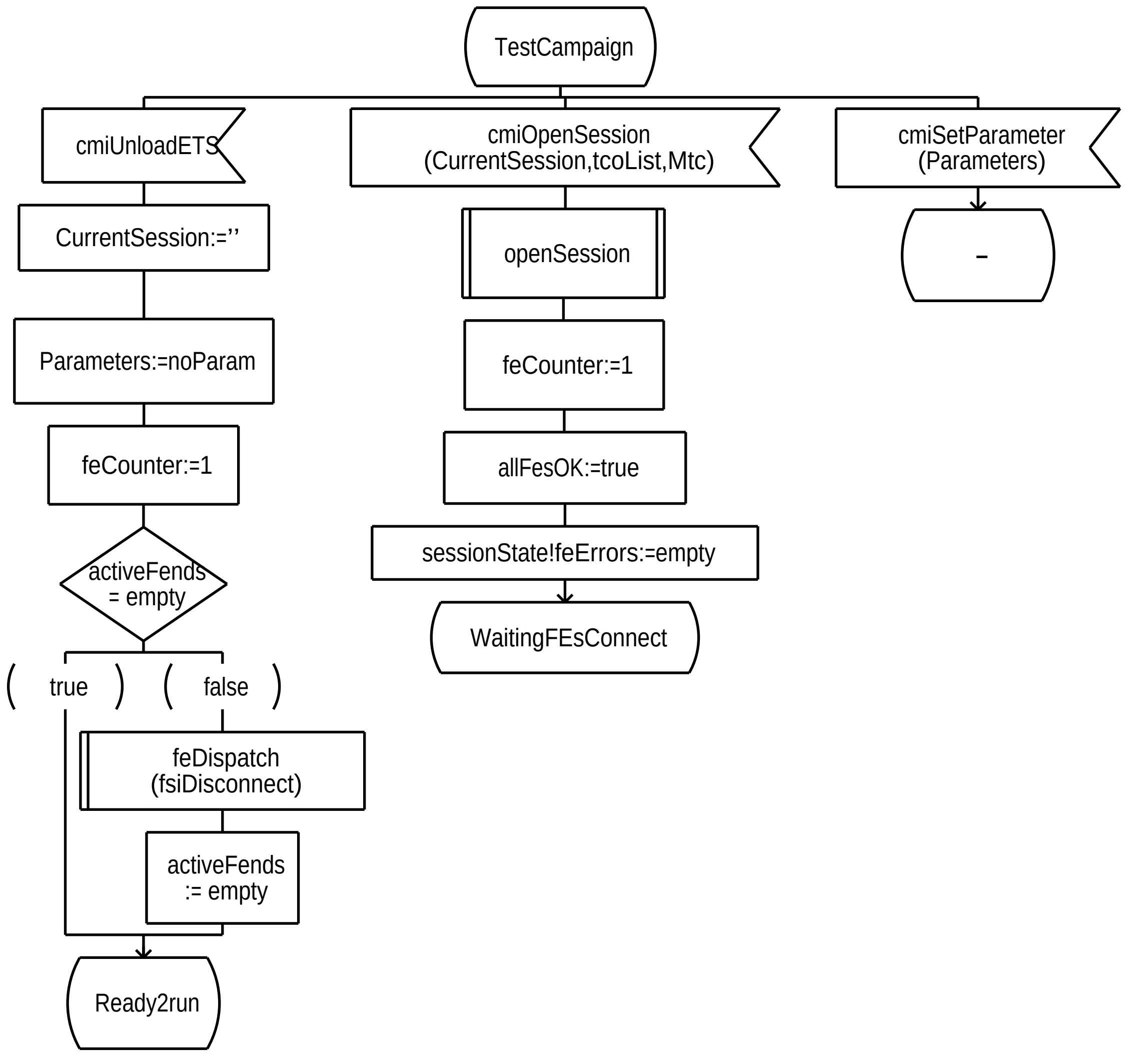

process Supervisor(1,1)
cmiRoute
(cmiConfFromSu)
cmiRoute
(cmiReqToSu)
SuMain
ping
initRoute
pong
initRoute
(cmiCommonConfFromSu)
(tmiToSu)
tmiRoute
cmiCommServRoute
commServRoute
startCommonServices, stopCommonServices
(cmiCommonReqToSu)
(fsiConfirm)
(tmiToTco)
tmiRoute
SuCommon
fsiRoute
(fsiCommonConf)
fsiCommServRoute
SuStubManager
fsiDisconnected
(fsiCommonReq)
(fsiRequest)
discRoute
fsiRoute

```
DCL REVEALED CurrentSession SessionId;
DCL REVEALED CurrentSuite EtsId;
DCL REVEALED Parameters ParamList;
DCL REVEALED Time TIME;
DCL REVEALED CurrentTest TestCaseId;
DCL REVEALED RunningTcos RoutingList;
DCL REVEALED Mtc MtcId;
DCL REVEALED MtcFePid PId;
DCL REVEALED SeFeIdList FeIdList;
```

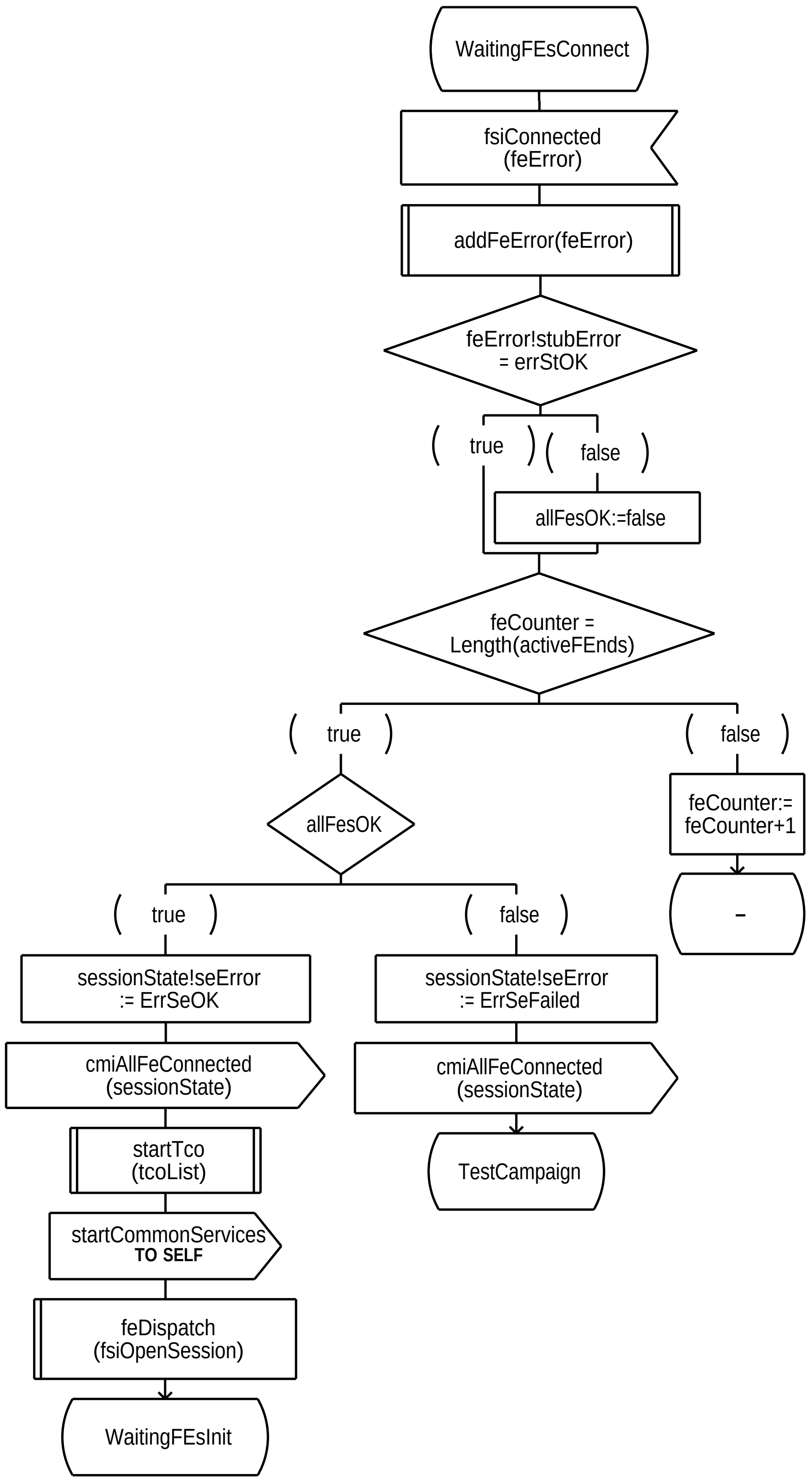
```
DCL myId FeId;
DCL myRoutingTable RoutingTable;

DCL frontEnd FrontEnd;
DCL activeFEnds RoutingList;

DCL feId, senderId, destId FeId;
DCL feError FeError;
DCL allFesOK, allTcosOK Boolean;
DCL sessionOK Boolean;
DCL sessionState SeError;
DCL msg Msg;

DCL testState EtcError;
DCL tcoCounter TcoIndex;
DCL tcoList TcoList;
DCL tcoId TcoId;
DCL fePid PId;
DCL tcoPid PId;
DCL stubPid PId;

DCL routingElem RoutingElement;

DCL err TcoError;
DCL verdict VerdictValue;
```

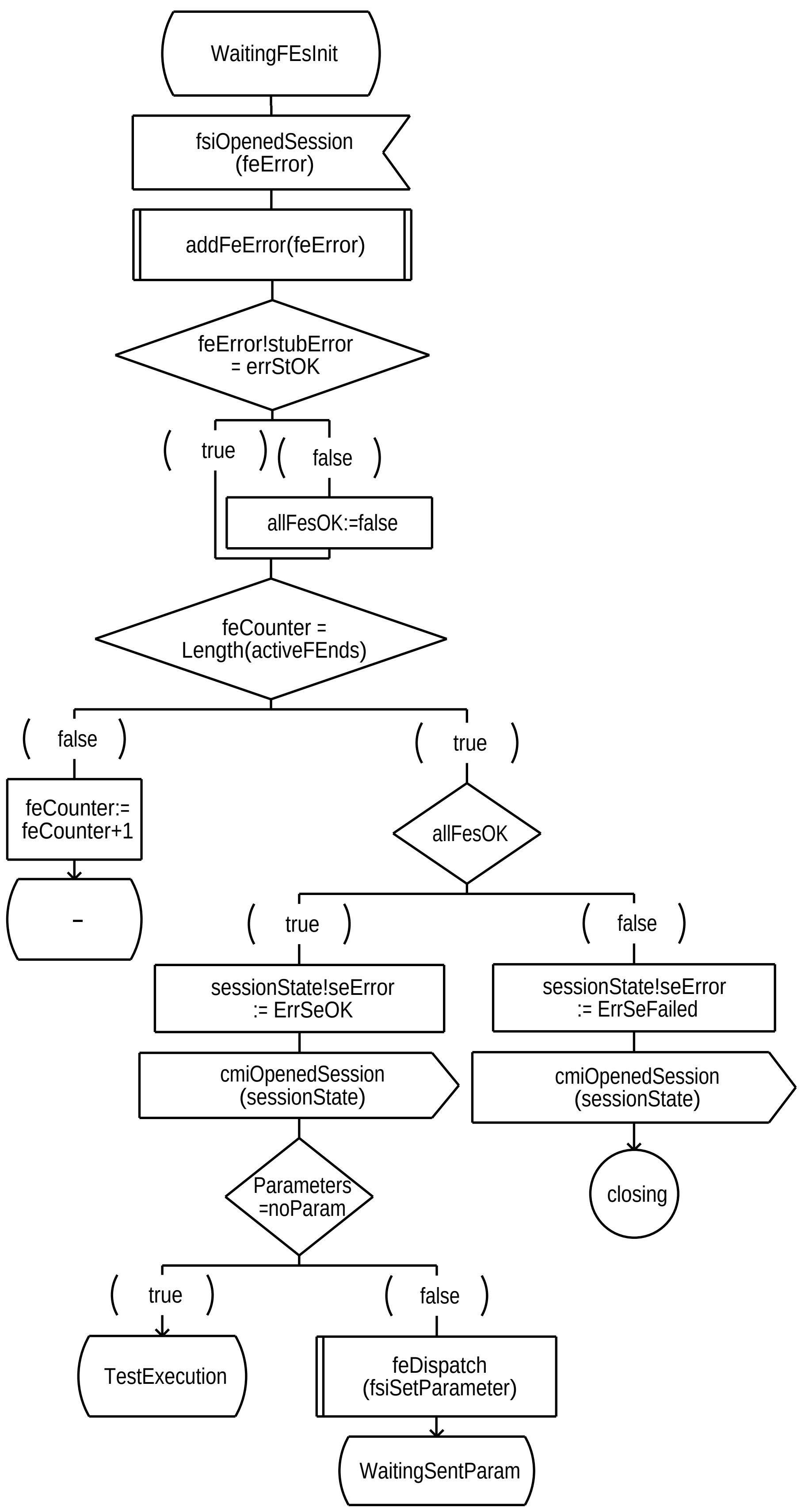


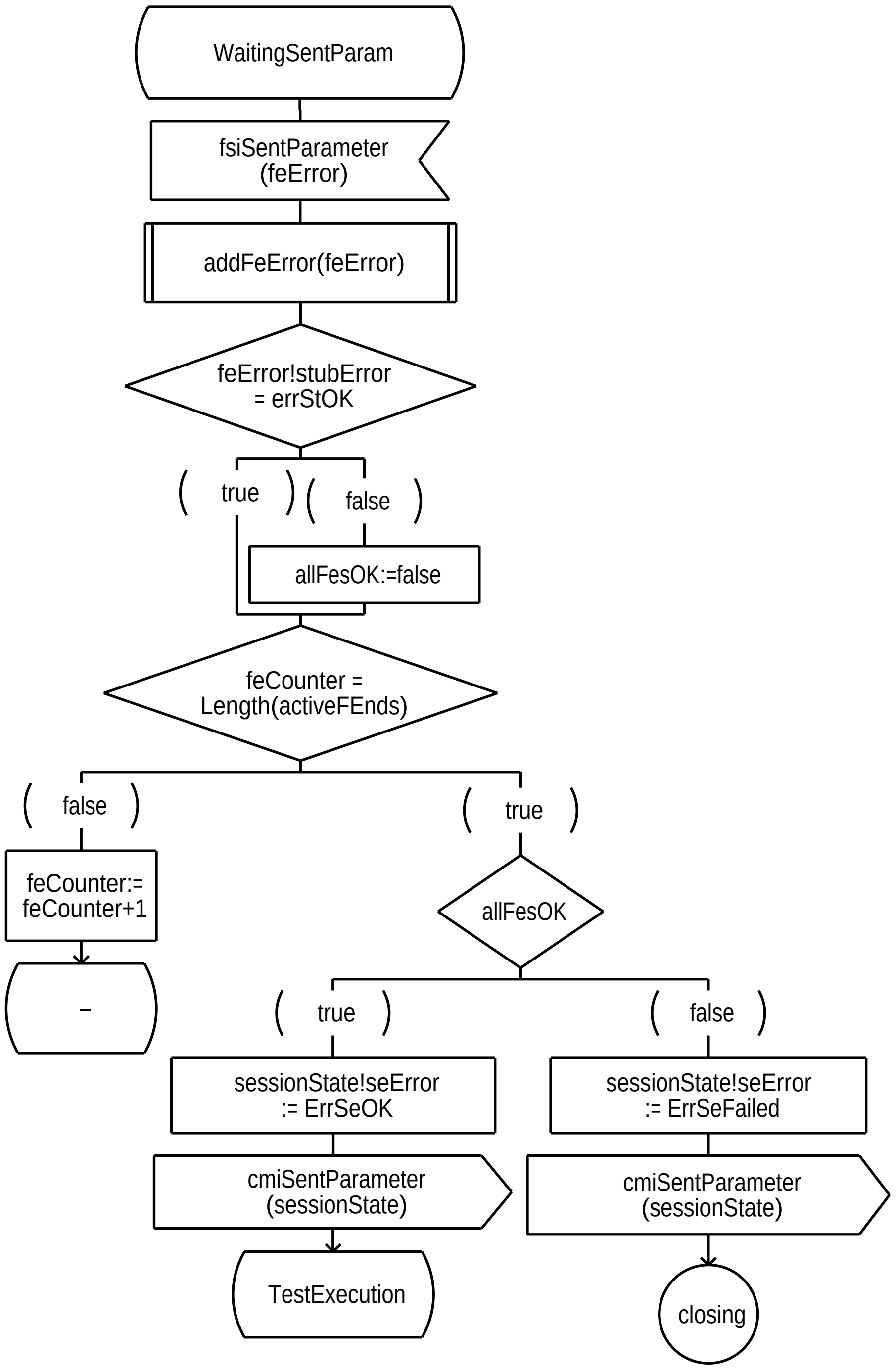
```
TIMER myTimer;

DCL feCounter FeIndex;
```

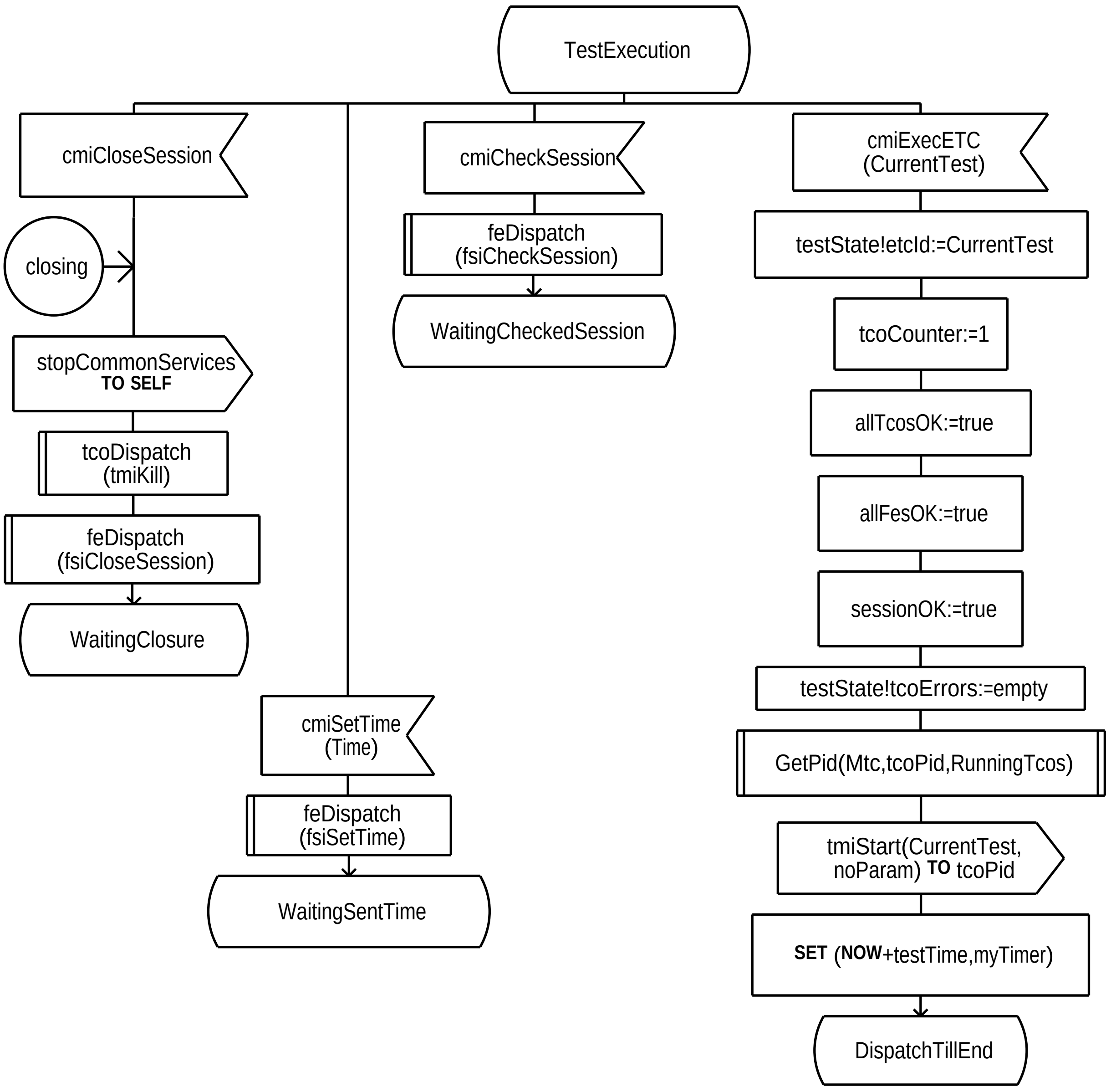


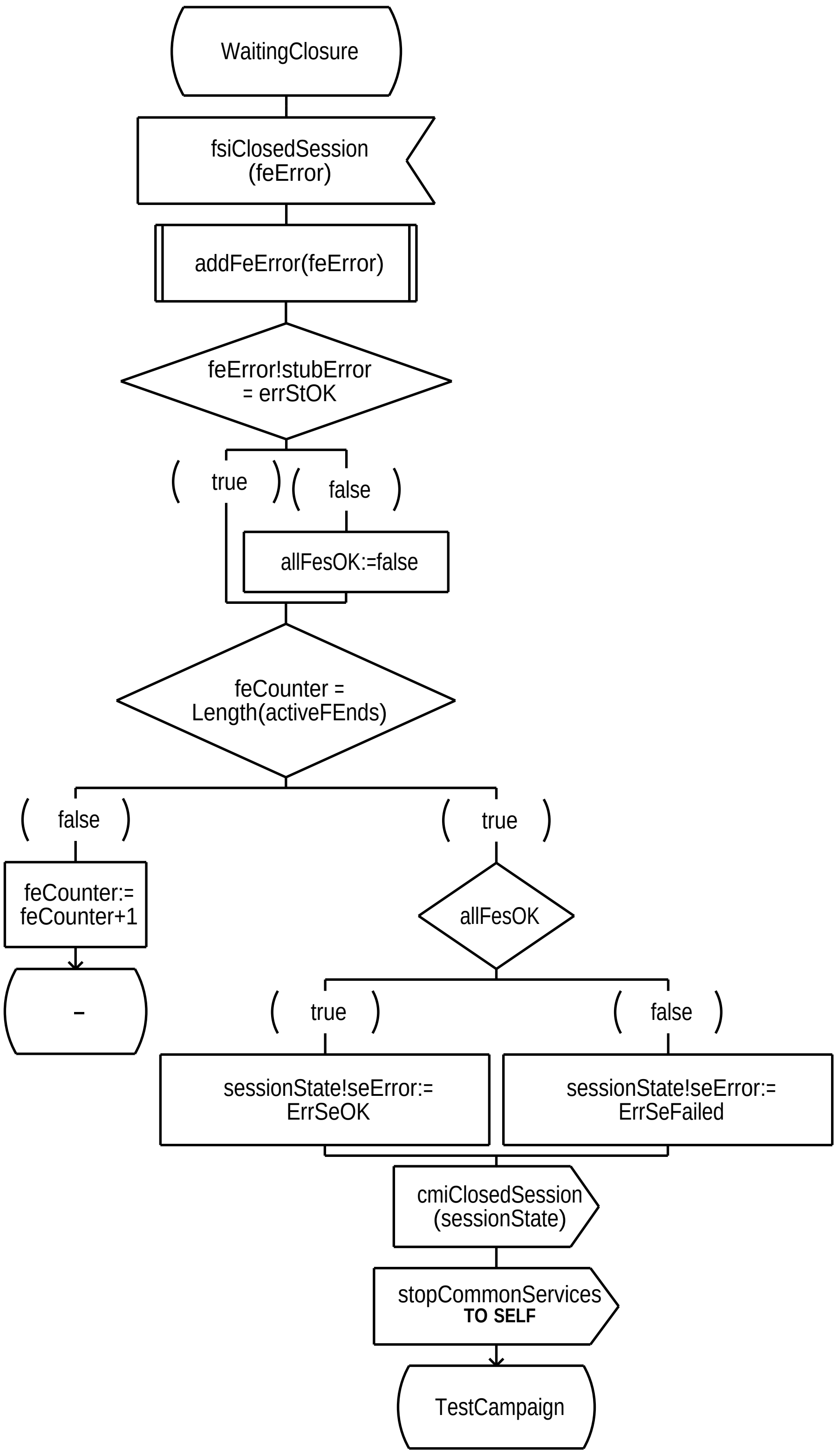
TestCampaign
cmiUnloadETS
cmiOpenSession (CurrentSession,tcoList,Mtc)
cmiSetParameter (Parameters)
CurrentSession:=’’
Parameters:=noParam
feCounter:=1
activeFends = empty
true
false
feDispatch (fsiDisconnect)
activeFends := empty
Ready2run
openSession
feCounter:=1
allFesOK:=true
sessionState!feErrors:=empty
WaitingFEsConnect
-

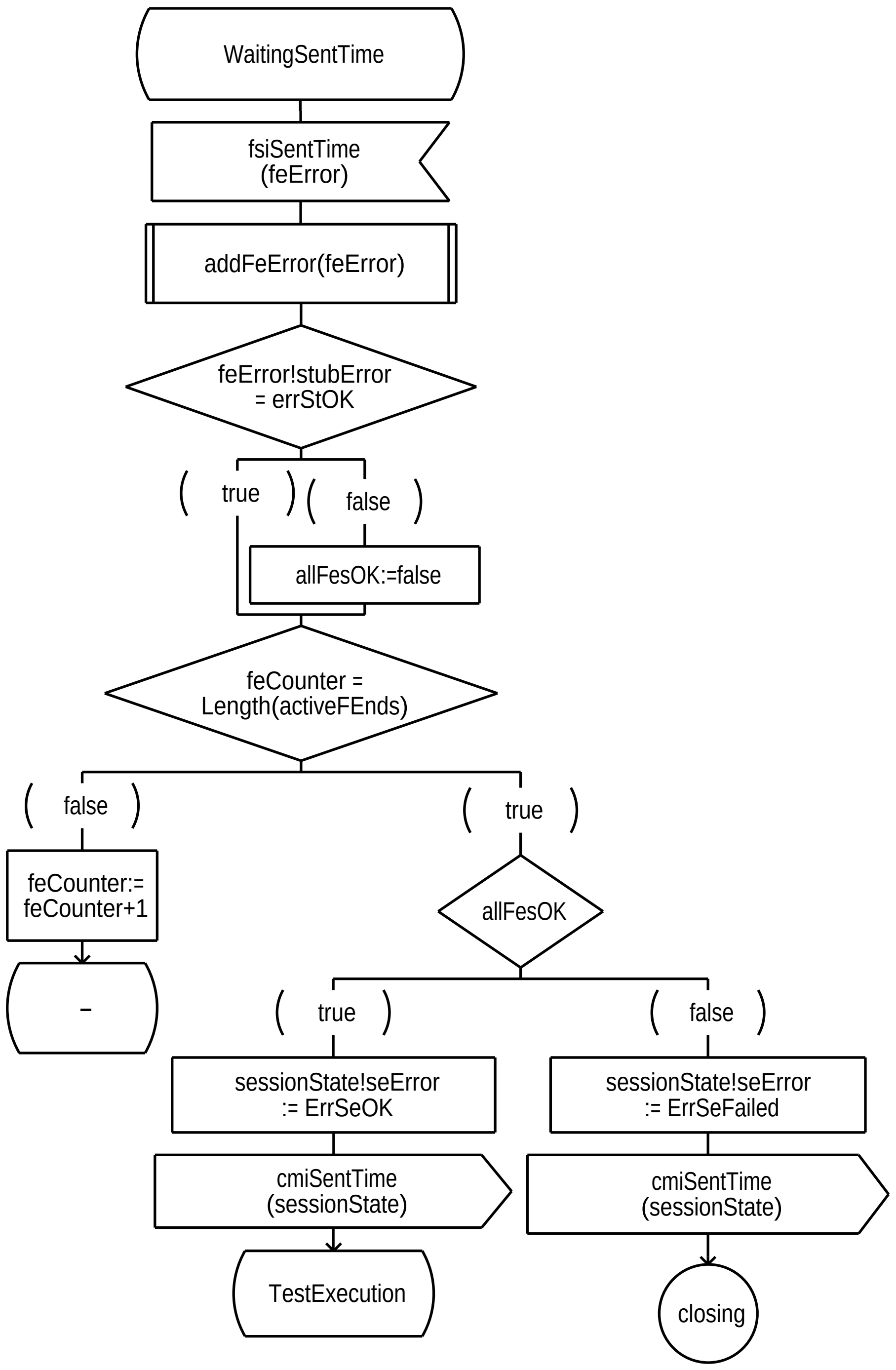

WaitingFEsConnect
fsiConnected (feError)
addFeError(feError)
feError!stubError = errStOK
true
false
allFesOK:=false
feCounter = Length(activeFEnds)
true
false
allFesOK
feCounter:= feCounter+1
-
true
false
sessionState!seError := ErrSeOK
sessionState!seError := ErrSeFailed
cmiAllFeConnected (sessionState)
cmiAllFeConnected (sessionState)
startTco (tcoList)
TestCampaign
startCommonServices TO SELF
feDispatch (fsiOpenSession)
WaitingFEsInit

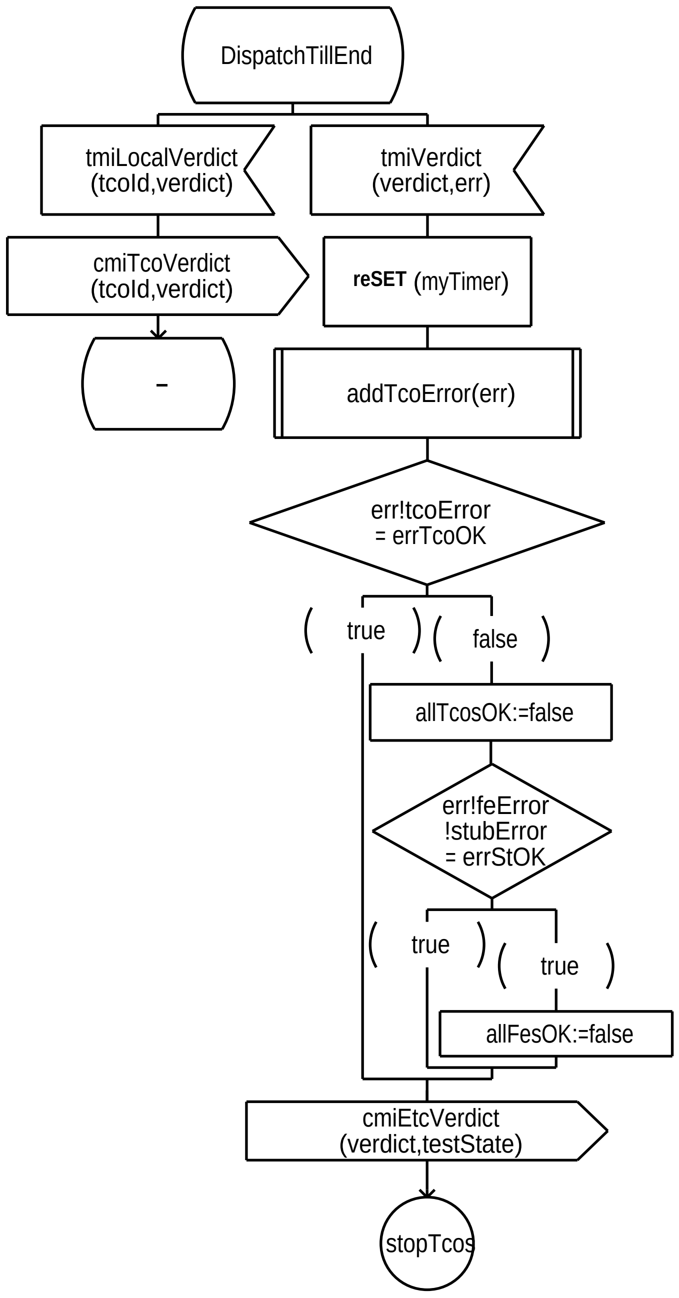

WaitingFEsInit
fsiOpenedSession
(feError)
addFeError(feError)
feError!stubError
= errStOK
true
false
allFesOK:=false
feCounter =
Length(activeFEnds)
false
true
feCounter:=
feCounter+1
-
allFesOK
true
false
sessionState!seError
:= ErrSeOK
sessionState!seError
:= ErrSeFailed
cmiOpenedSession
(sessionState)
cmiOpenedSession
(sessionState)
Parameters
=noParam
closing
true
false
TestExecution
feDispatch
(fsiSetParameter)
WaitingSentParam

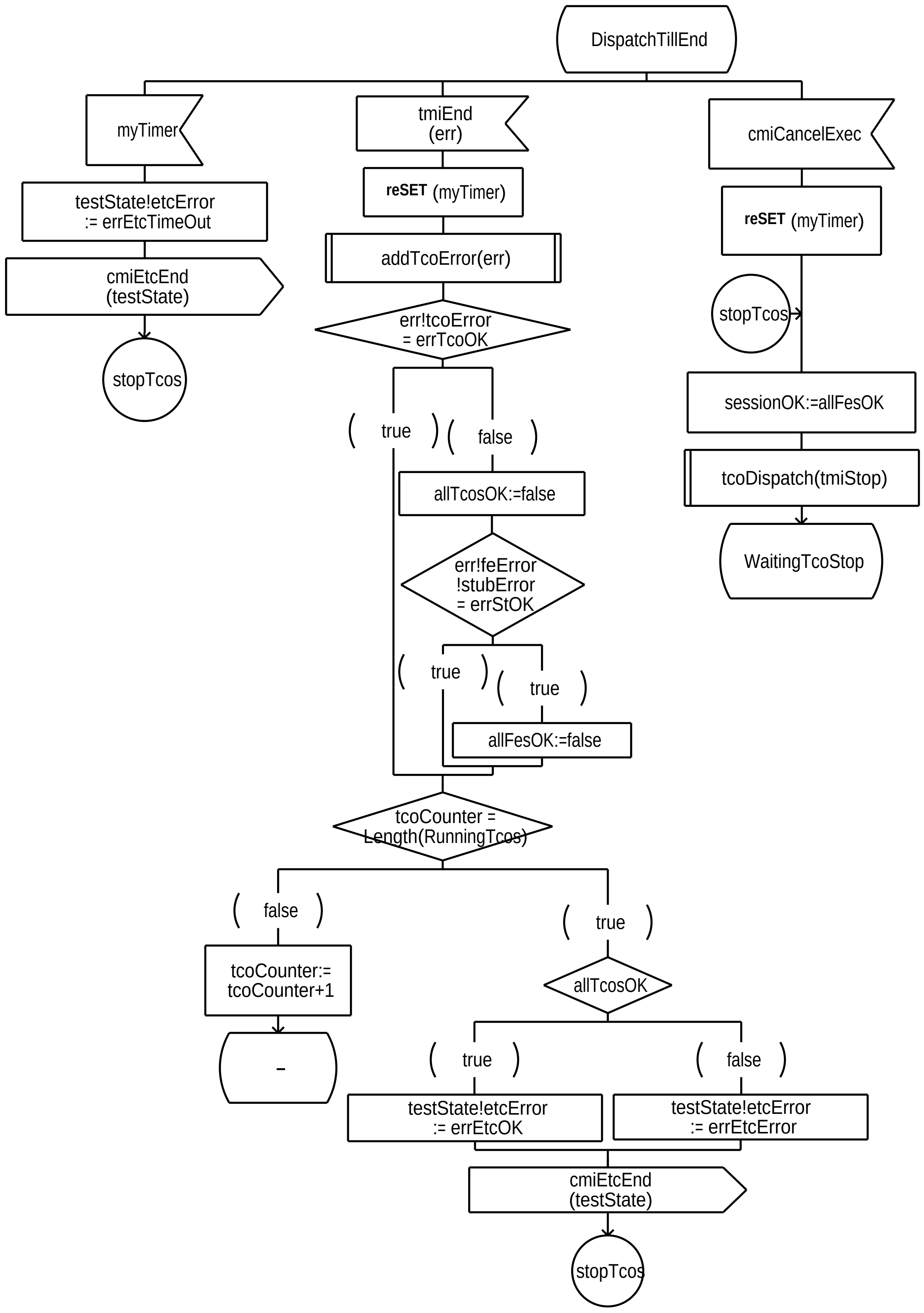

WaitingSentParam
fsiSentParameter (feError)
addFeError(feError)
feError!stubError = errStOK
true
false
allFesOK:=false
feCounter = Length(activeFEnds)
false
true
feCounter:= feCounter+1
-
allFesOK
true
false
sessionState!seError := ErrSeOK
sessionState!seError := ErrSeFailed
cmiSentParameter (sessionState)
cmiSentParameter (sessionState)
TestExecution
closing

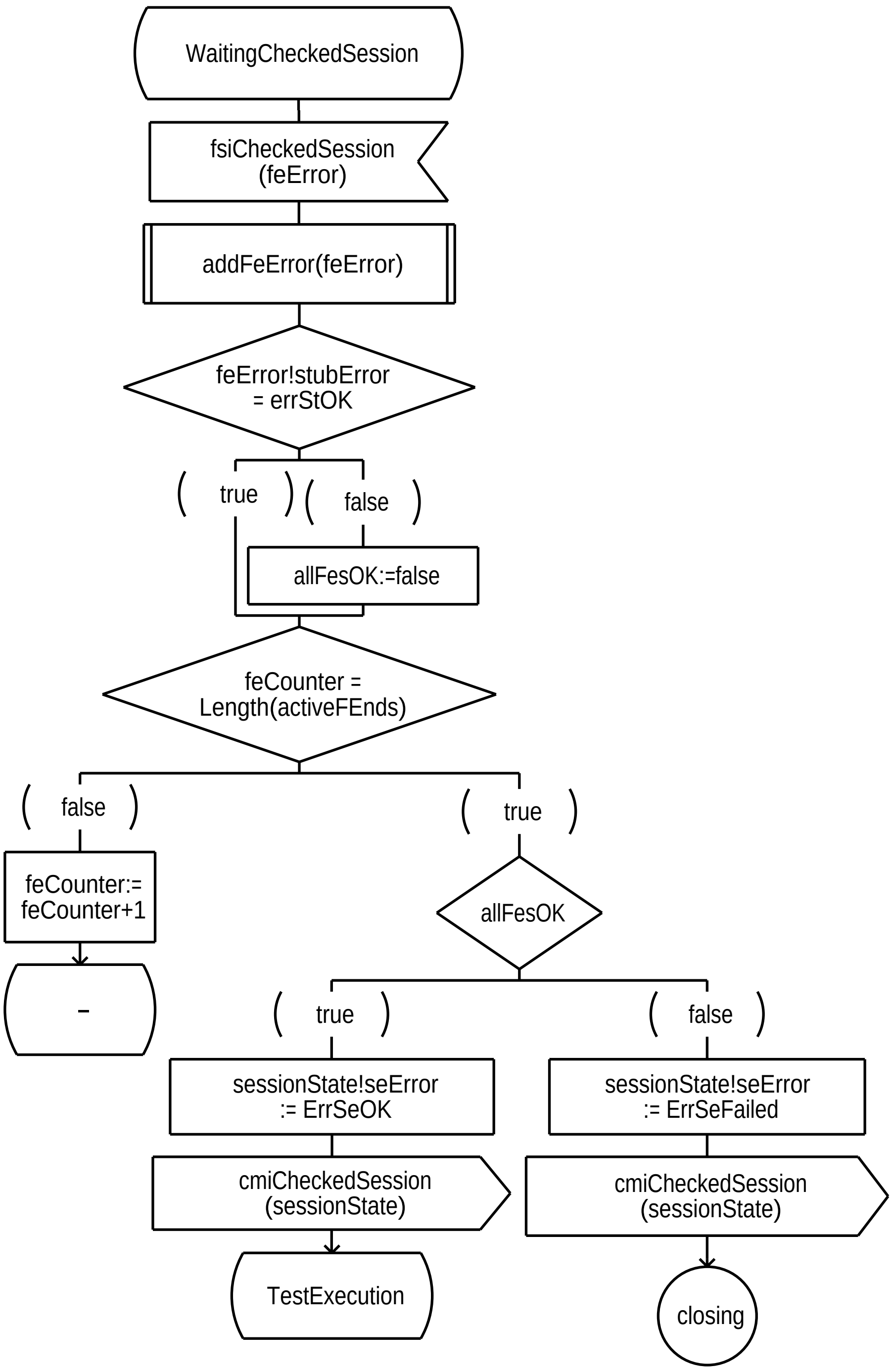

TestExecution
cmiCloseSession
closing
stopCommonServices TO SELF
tcoDispatch (tmiKill)
feDispatch (fsiCloseSession)
WaitingClosure
cmiSetTime (Time)
feDispatch (fsiSetTime)
WaitingSentTime
cmiCheckSession
feDispatch (fsiCheckSession)
WaitingCheckedSession
cmiExecETC (CurrentTest)
testState!etcId:=CurrentTest
tcoCounter:=1
allTcosOK:=true
allFesOK:=true
sessionOK:=true
testState!tcoErrors:=empty
GetPid(Mtc,tcoPid,RunningTcos)
tmiStart(CurrentTest, noParam) TO tcoPid
SET (NOW+testTime,myTimer)
DispatchTillEnd


WaitingClosure
fsiClosedSession
(feError)
addFeError(feError)
feError!stubError
= errStOK
true
false
allFesOK:=false
feCounter =
Length(activeFEnds)
false
true
feCounter:=
feCounter+1
-
allFesOK
true
false
sessionState!seError:=
ErrSeOK
sessionState!seError:=
ErrSeFailed
cmiClosedSession
(sessionState)
stopCommonServices
TO SELF
TestCampaign

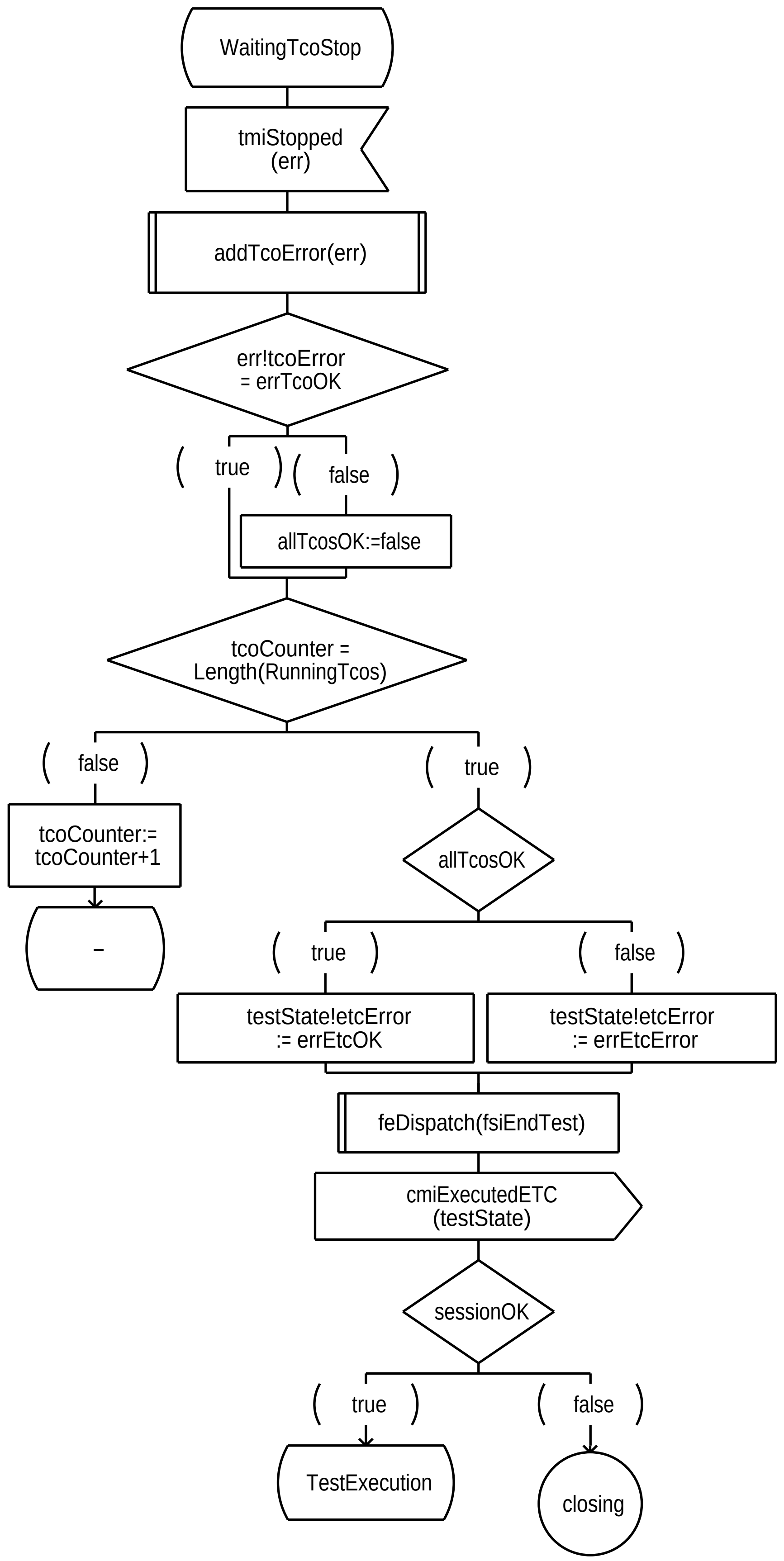

WaitingSentTime
fsiSentTime (feError)
addFeError(feError)
feError!stubError = errStOK
true
false
allFesOK:=false
feCounter = Length(activeFEnds)
false
true
feCounter:= feCounter+1
-
allFesOK
true
false
sessionState!seError := ErrSeOK
sessionState!seError := ErrSeFailed
cmiSentTime (sessionState)
cmiSentTime (sessionState)
TestExecution
closing


DispatchTillEnd
tmiLocalVerdict
(tcoId,verdict)
tmiVerdict
(verdict,err)
cmiTcoVerdict
(tcoId,verdict)
reSET (myTimer)
–
addTcoError(err)
err!tcoError
= errTcoOK
true
false
allTcosOK:=false
err!feError
!stubError
= errStOK
true
true
allFesOK:=false
cmiEtcVerdict
(verdict,testState)
stopTcos

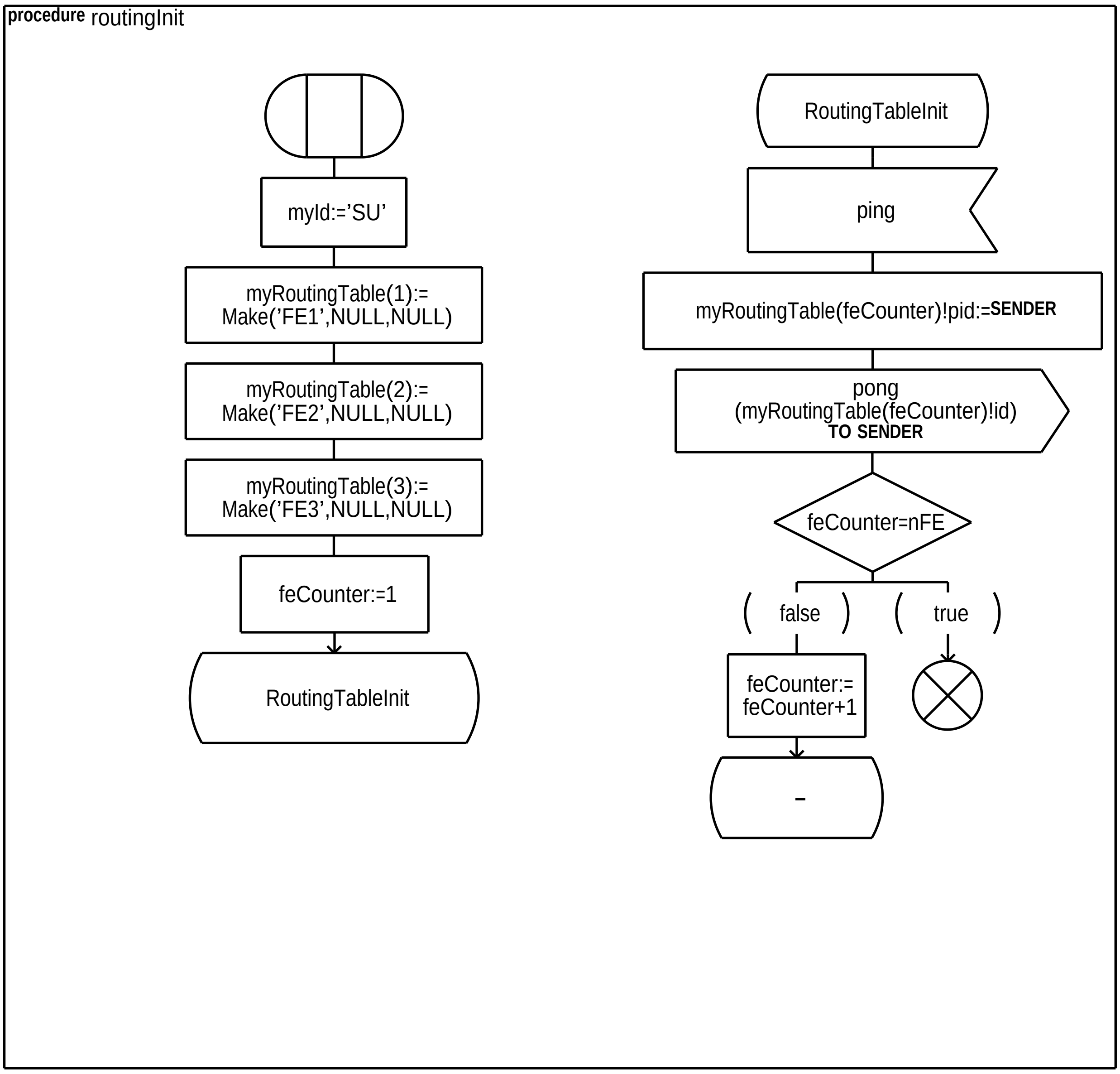

DispatchTillEnd
myTimer
testState!etcError := errEtcTimeOut
cmiEtcEnd (testState)
stopTcos
tmiEnd (err)
reSET (myTimer)
addTcoError(err)
err!tcoError = errTcoOK
true
false
allTcosOK:=false
err!feError !stubError = errStOK
true
true
allFesOK:=false
tcoCounter = Length(RunningTcos)
false
tcoCounter:= tcoCounter+1
-
true
allTcosOK
true
false
testState!etcError := errEtcOK
testState!etcError := errEtcError
cmiEtcEnd (testState)
stopTcos
cmiCancelExec
reSET (myTimer)
stopTcos
sessionOK:=allFesOK
tcoDispatch(tmiStop)
WaitingTcoStop

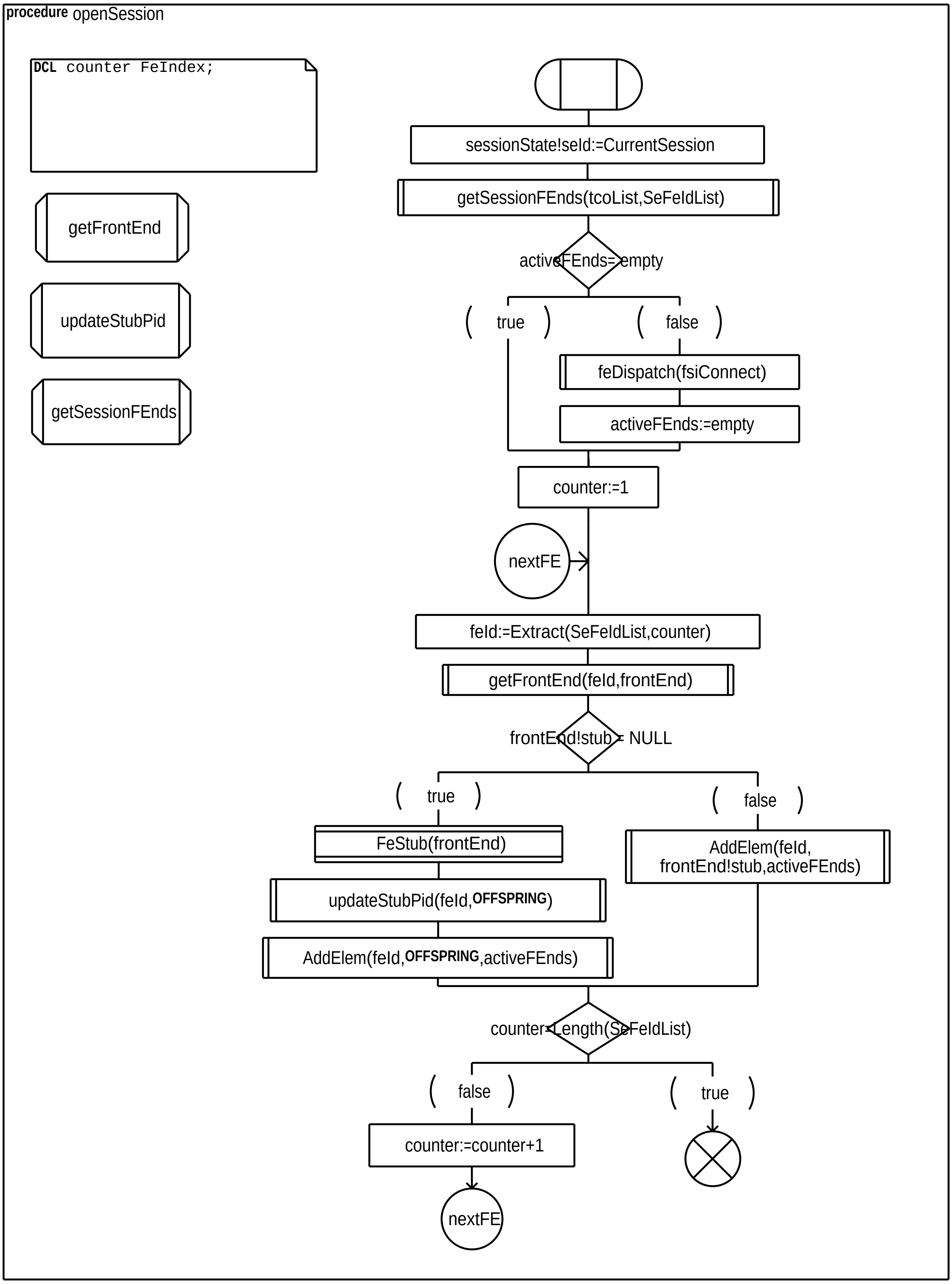


WaitingCheckedSession
fsiCheckedSession (feError)
addFeError(feError)
feError!stubError = errStOK
true
false
allFesOK:=false
feCounter = Length(activeFEnds)
false
true
feCounter:= feCounter+1
-
allFesOK
true
false
sessionState!seError := ErrSeOK
sessionState!seError := ErrSeFailed
cmiCheckedSession (sessionState)
cmiCheckedSession (sessionState)
TestExecution
closing

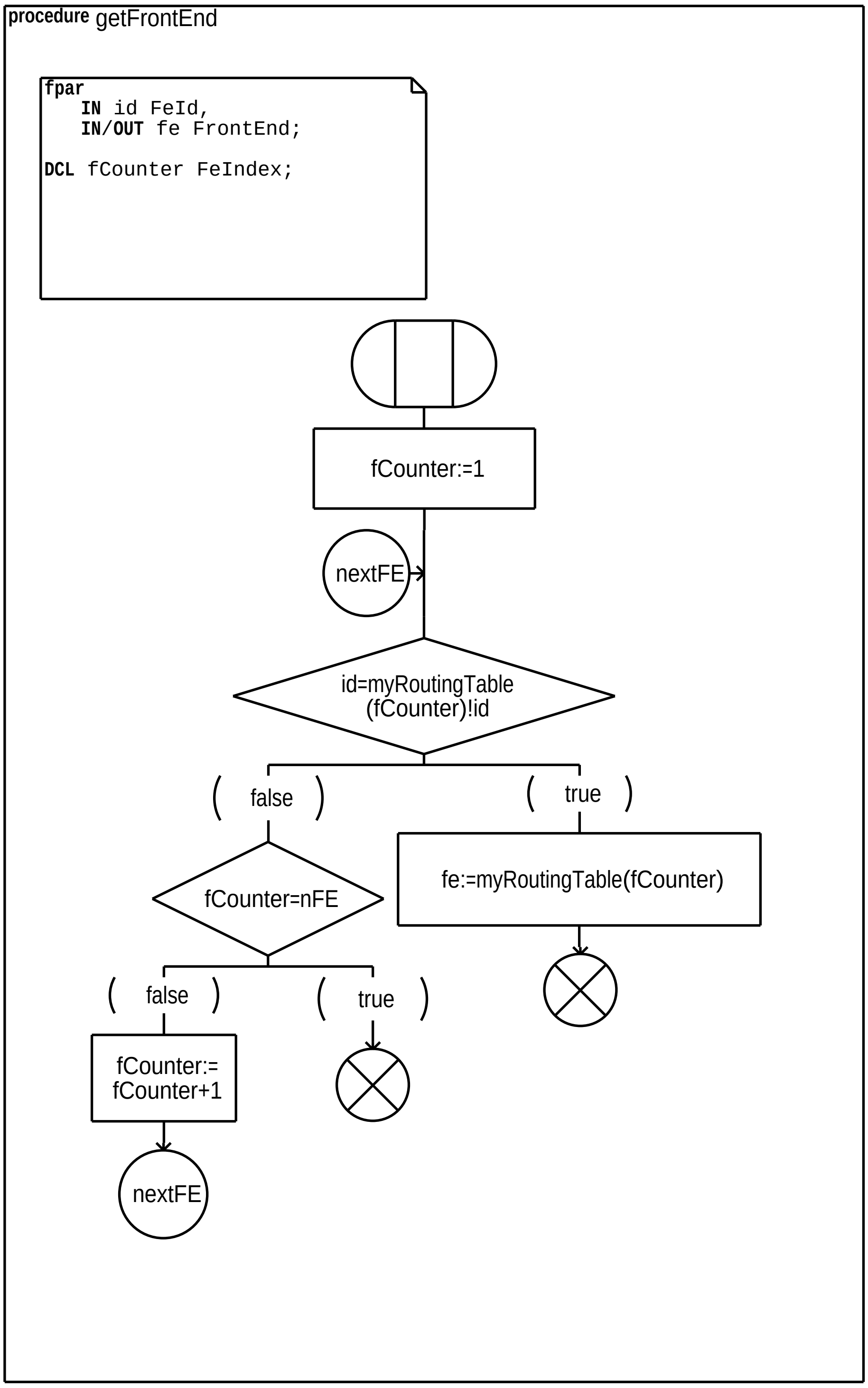


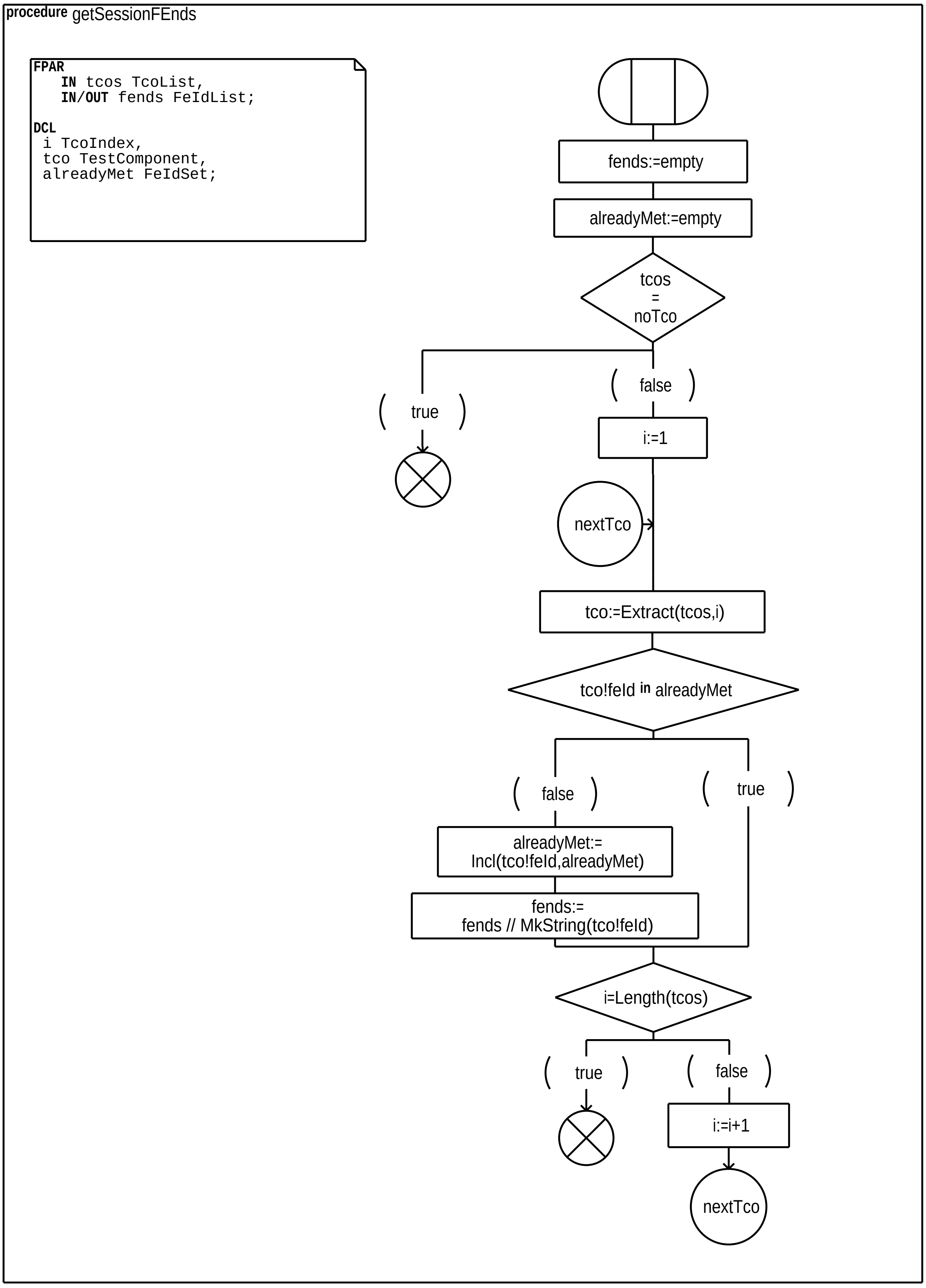
WaitingTcoStop
tmiStopped (err)
addTcoError(err)
err!tcoError = errTcoOK
true
false
allTcosOK:=false
tcoCounter = Length(RunningTcos)
false
true
tcoCounter:= tcoCounter+1
-
allTcosOK
true
false
testState!etcError := errEtcOK
testState!etcError := errEtcError
feDispatch(fsiEndTest)
cmiExecutedETC (testState)
sessionOK
true
false
TestExecution
closing

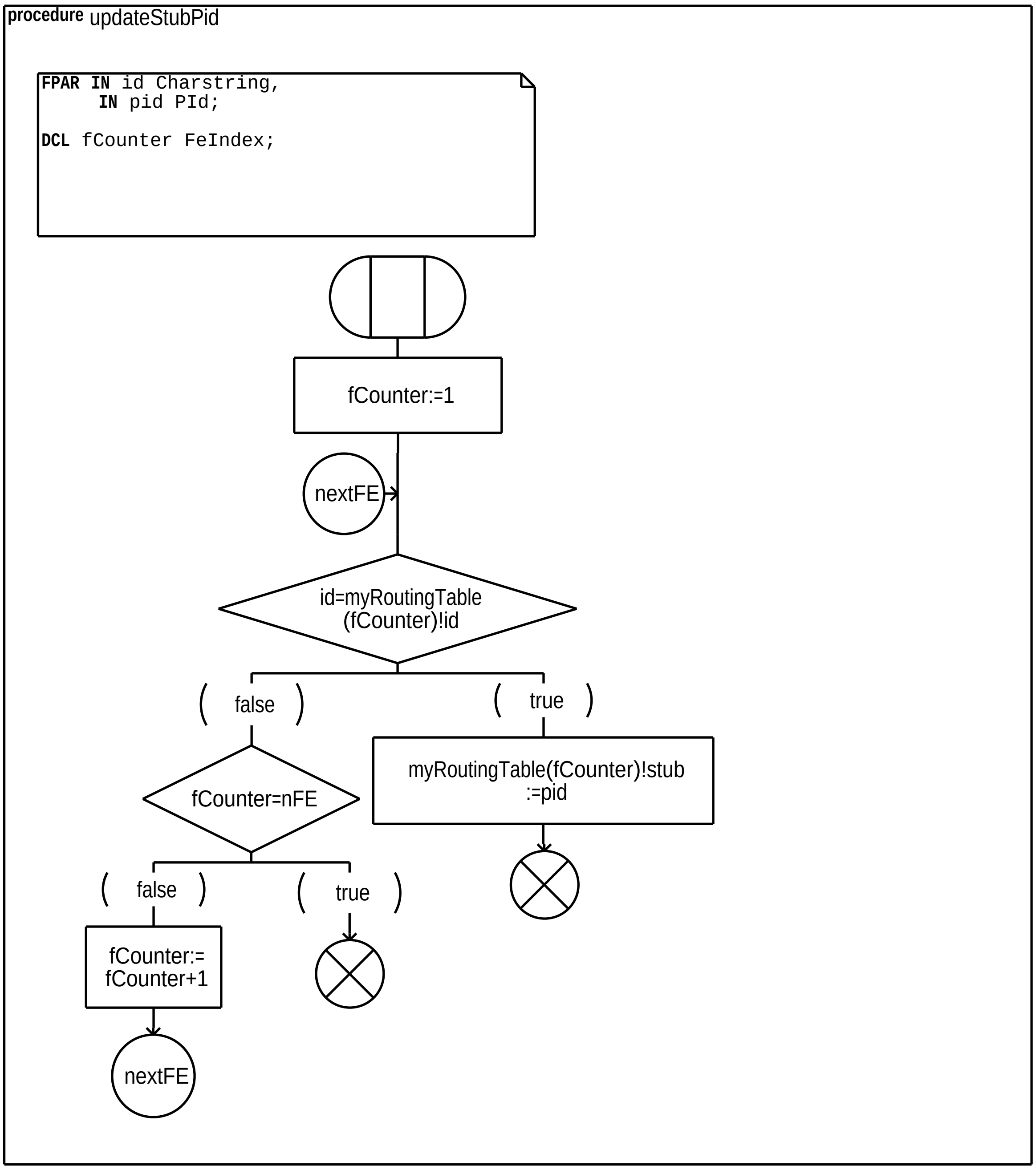

procedure routingInit
myId:='SU'
myRoutingTable(1):=
Make('FE1',NULL,NULL)
myRoutingTable(2):=
Make('FE2',NULL,NULL)
myRoutingTable(3):=
Make('FE3',NULL,NULL)
feCounter:=1
RoutingTableInit
RoutingTableInit
ping
myRoutingTable(feCounter)!pid:=SENDER
pong
(myRoutingTable(feCounter)!id)
TO SENDER
feCounter=nFE
false
true
feCounter:=
feCounter+1
-

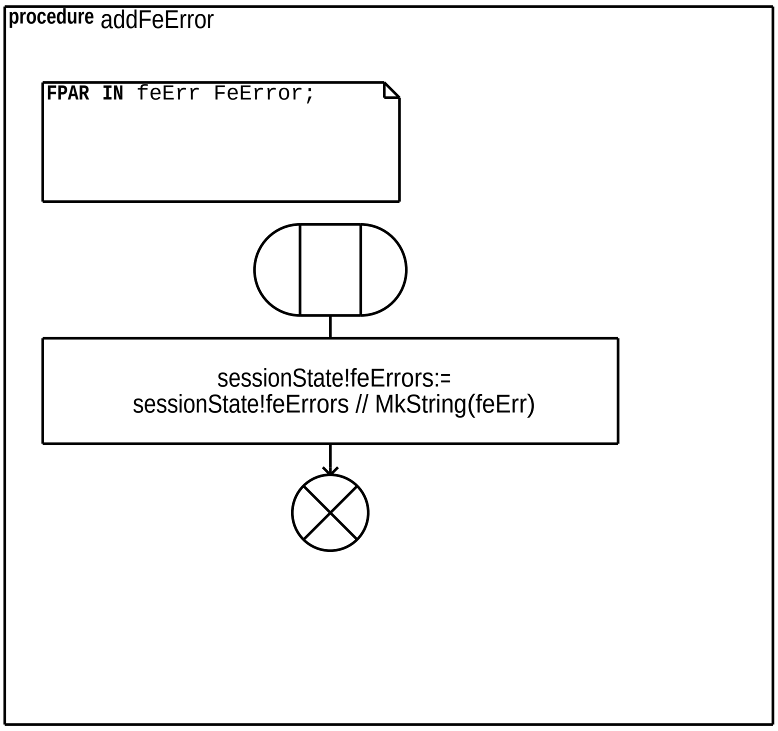


procedure openSession
DCL counter FeIndex;
getFrontEnd
updateStubPid
getSessionFEnds
sessionState!seId:=CurrentSession
getSessionFEnds(tcoList,SeFeIdList)
activeFEnds=empty
true
false
feDispatch(fsiConnect)
activeFEnds:=empty
counter:=1
nextFE
feId:=Extract(SeFeIdList,counter)
getFrontEnd(feId,frontEnd)
frontEnd!stub = NULL
true
false
FeStub(frontEnd)
updateStubPid(feId,OFFSPRING)
AddElem(feId,OFFSPRING,activeFEnds)
AddElem(feId, frontEnd!stub,activeFEnds)
counter=Length(SeFeIdList)
false
true
counter:=counter+1
nextFE

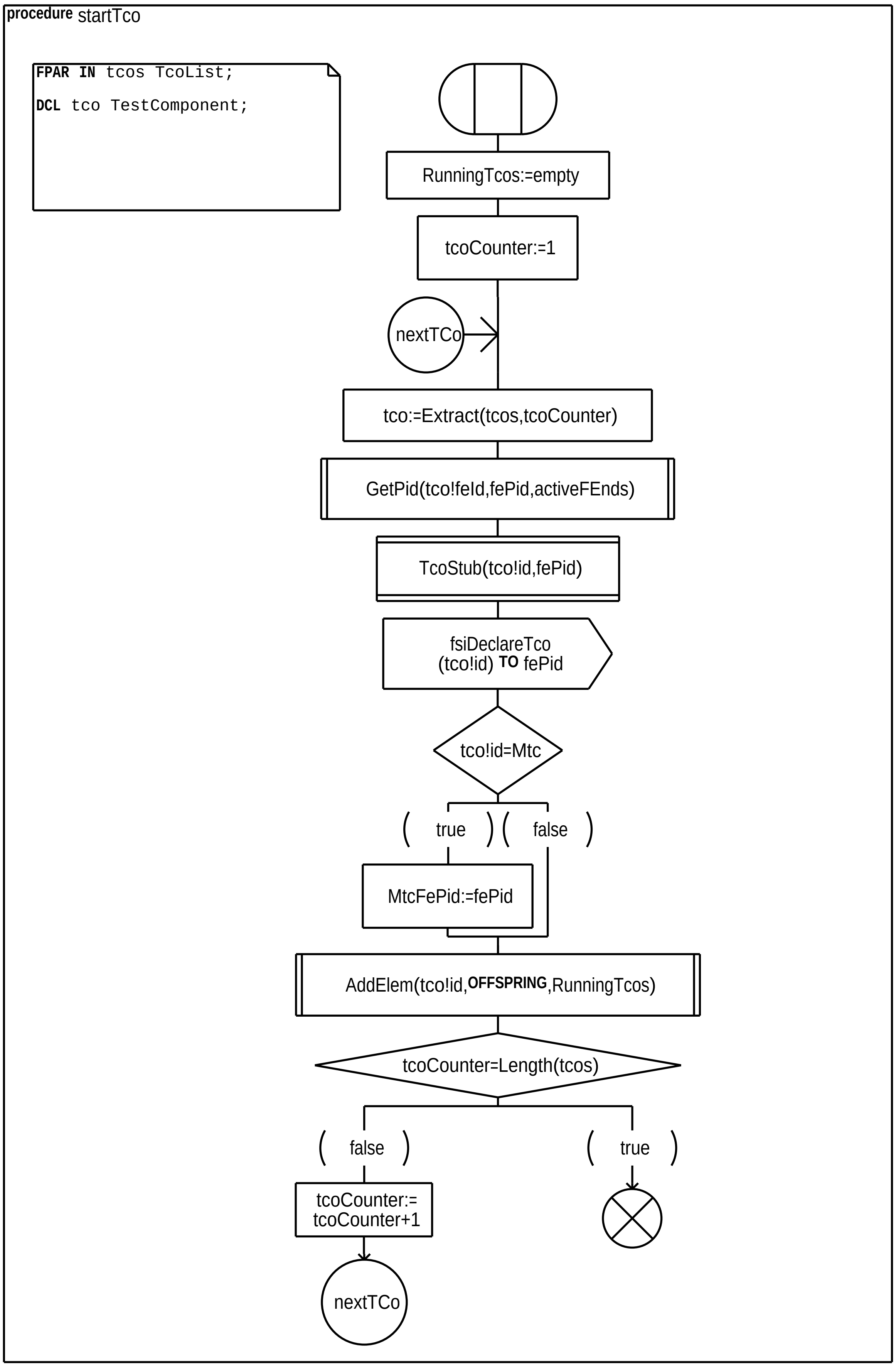

procedure getFrontEnd
fpar
IN id FeId,
IN/OUT fe FrontEnd;
DCL fCounter FeIndex;
fCounter:=1
nextFE
id=myRoutingTable (fCounter)!id
false
true
fCounter=nFE
fe:=myRoutingTable(fCounter)
false
true
fCounter:= fCounter+1
nextFE

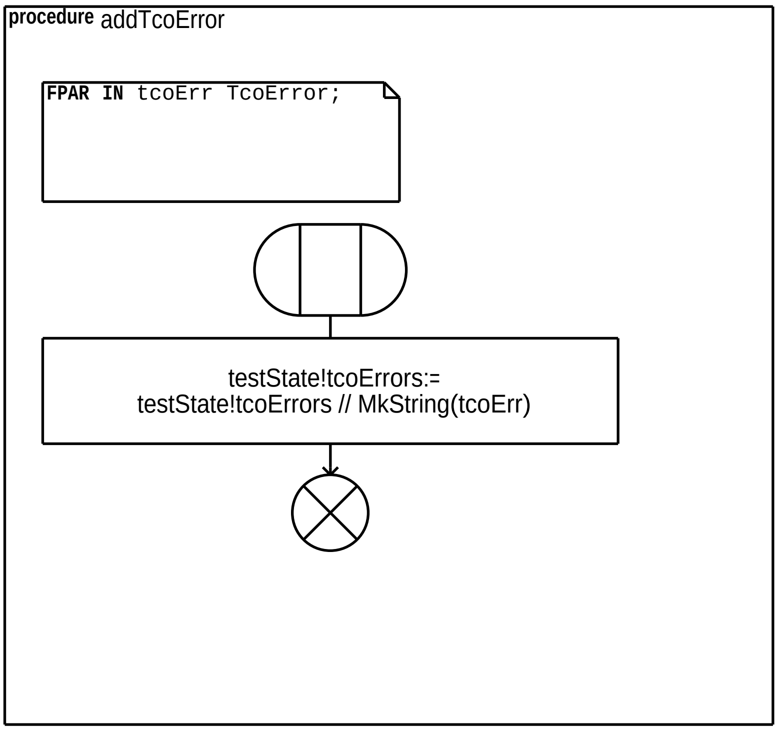

procedure getSessionFEnds
FPAR
IN tcos TcoList,
IN/OUT fends FeIdList;
DCL
i TcoIndex,
tco TestComponent,
alreadyMet FeIdSet;
fends:=empty
alreadyMet:=empty
tcos = noTco
true
false
i:=1
nextTco
tco:=Extract(tcos,i)
tco!feId in alreadyMet
false
true
alreadyMet:= Incl(tco!feId,alreadyMet)
fends:= fends // MkString(tco!feId)
i=Length(tcos)
true
false
i:=i+1
nextTco

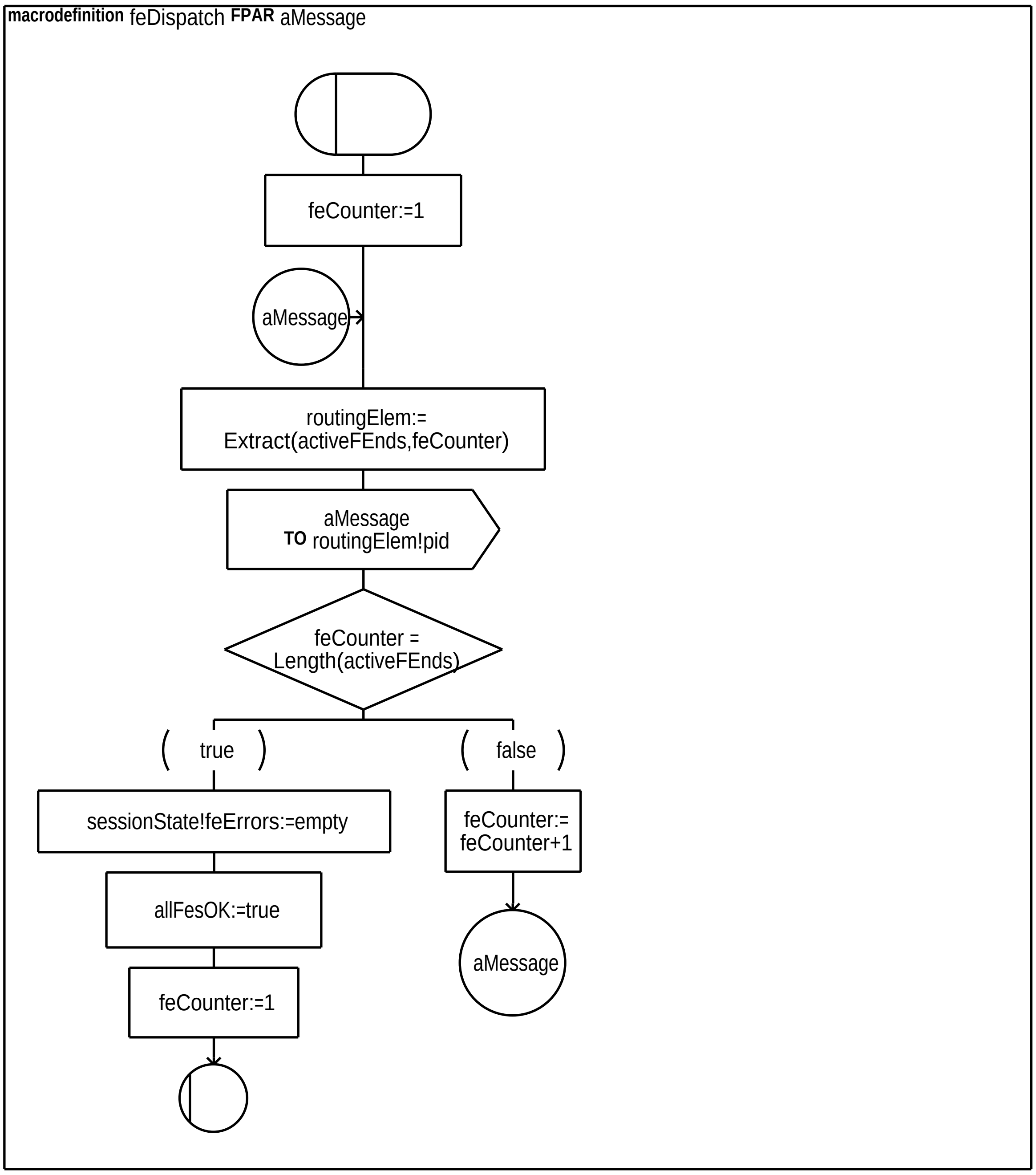

procedure updateStubPid
FPAR IN id Charstring,
IN pid PId;
DCL fCounter FeIndex;
fCounter:=1
nextFE
id=myRoutingTable (fCounter)!id
false
true
fCounter=nFE
myRoutingTable(fCounter)!stub :=pid
false
true
fCounter:= fCounter+1
nextFE

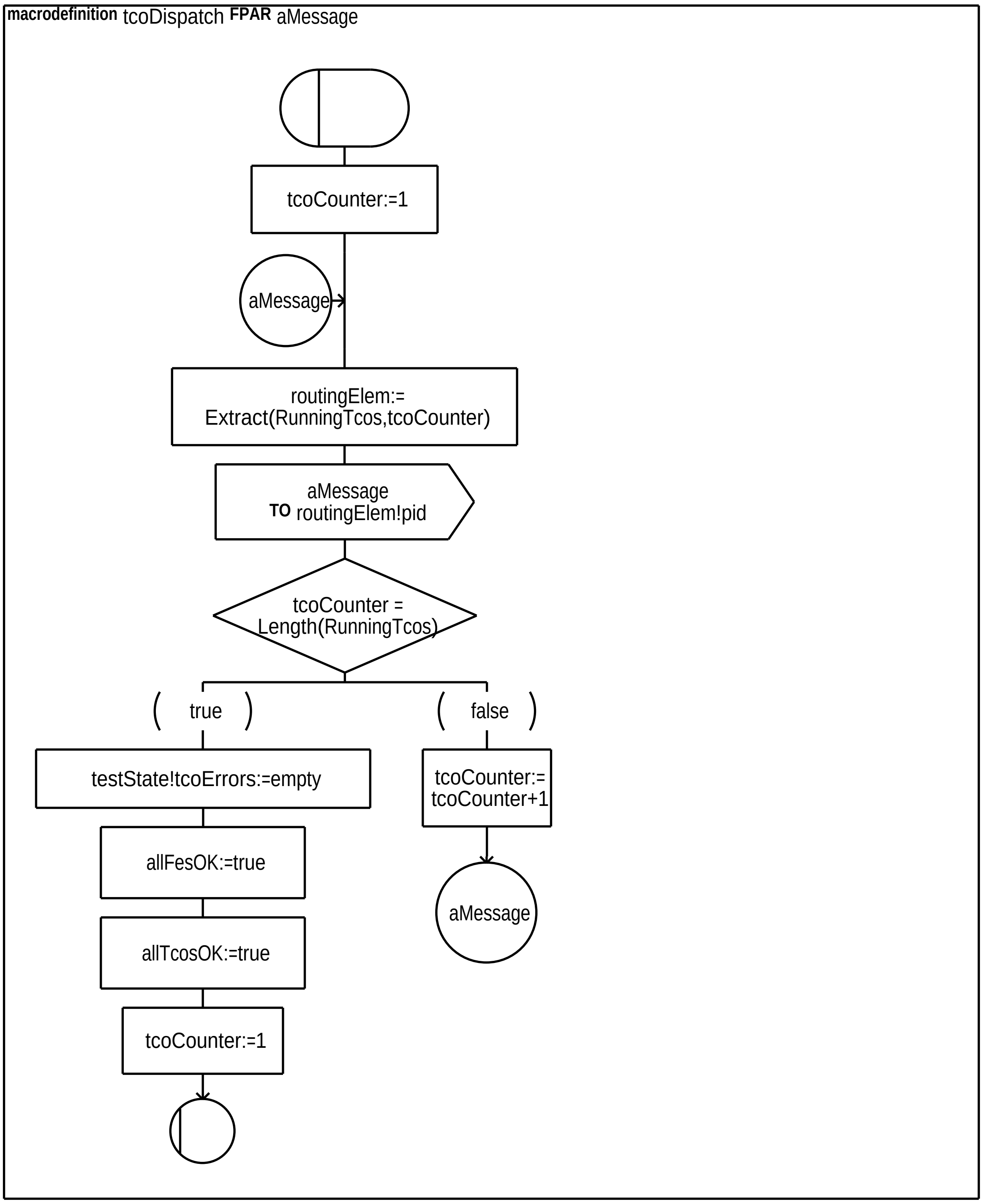


procedure addFeError
FPAR IN feErr FeError;
sessionState!feErrors:=
sessionState!feErrors // MkString(feErr)

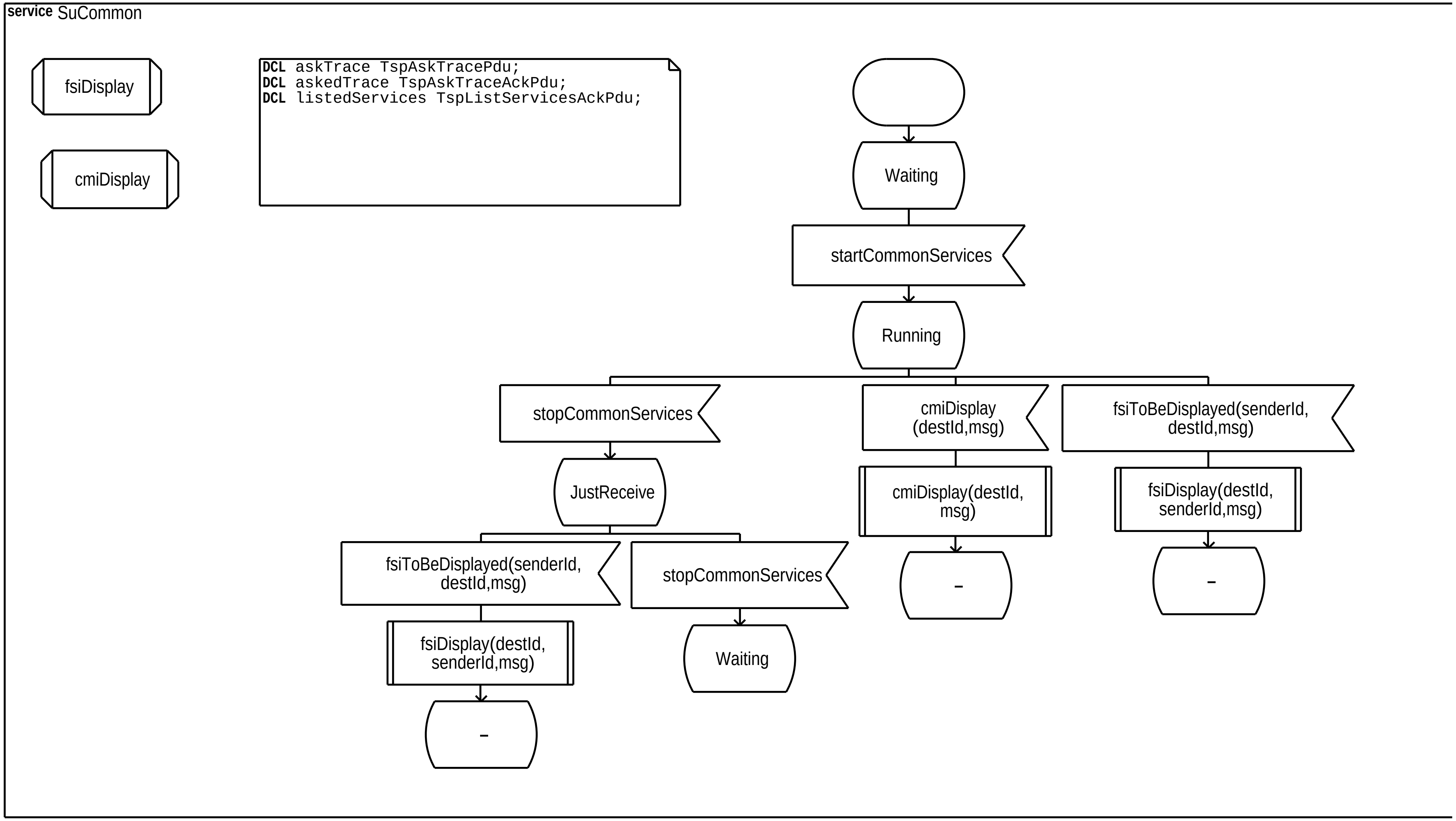

procedure startTco
FPAR IN tcos TcoList;
DCL tco TestComponent;
RunningTcos:=empty
tcoCounter:=1
nextTCo
tco:=Extract(tcos,tcoCounter)
GetPid(tco!feId,fePid,activeFEnds)
TcoStub(tco!id,fePid)
fsiDeclareTco (tco!id) TO fePid
tco!id=Mtc
true
false
MtcFePid:=fePid
AddElem(tco!id,OFFSPRING,RunningTcos)
tcoCounter=Length(tcos)
false
true
tcoCounter:= tcoCounter+1
nextTCo


procedure addTcoError
FPAR IN tcoErr TcoError;
testState!tcoErrors:=
testState!tcoErrors // MkString(tcoErr)

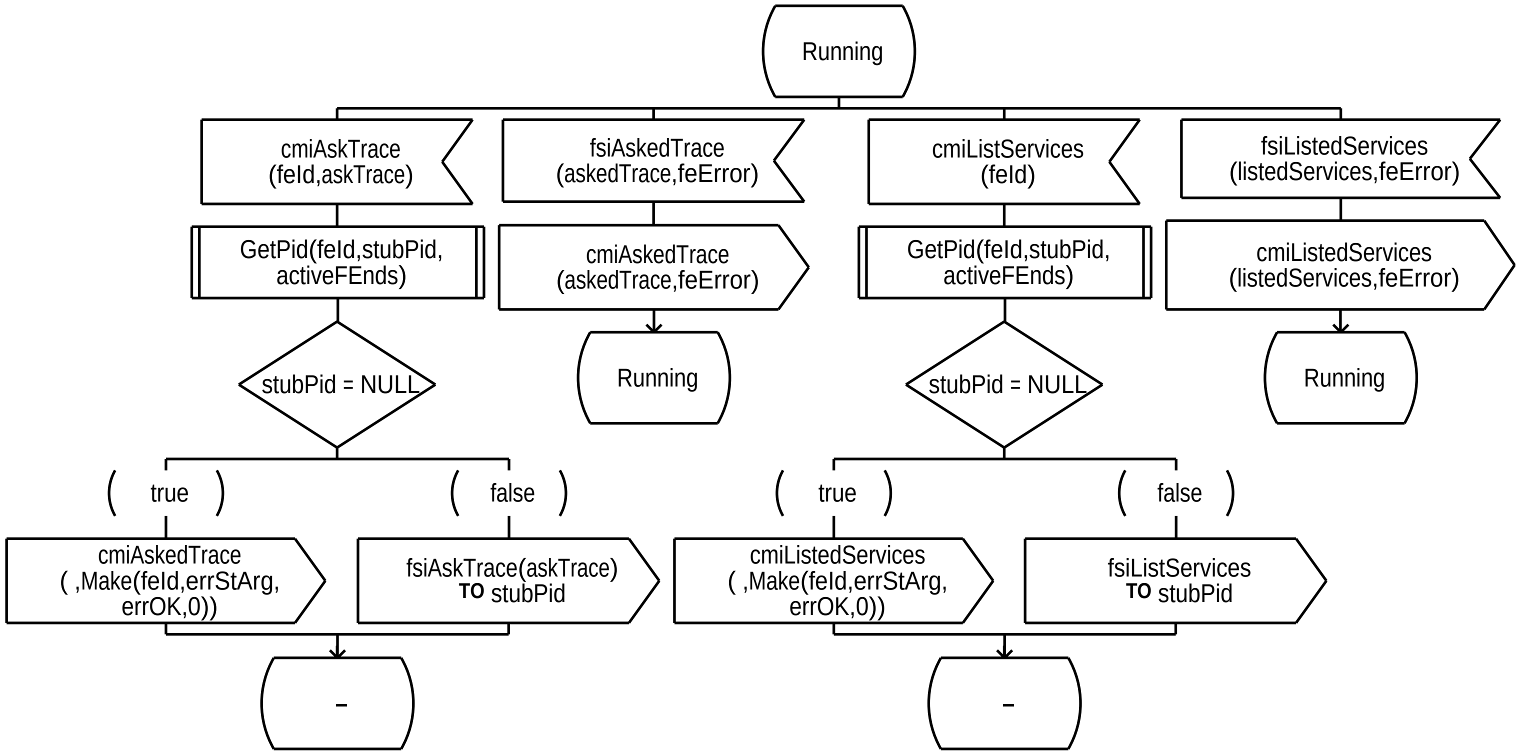

macrodefinition feDispatch FPAR aMessage
feCounter:=1
aMessage
routingElem:=
Extract(activeFEnds,feCounter)
aMessage
TO routingElem!pid
feCounter =
Length(activeFEnds)
true
false
sessionState!feErrors:=empty
allFesOK:=true
feCounter:=1
feCounter:=
feCounter+1
aMessage

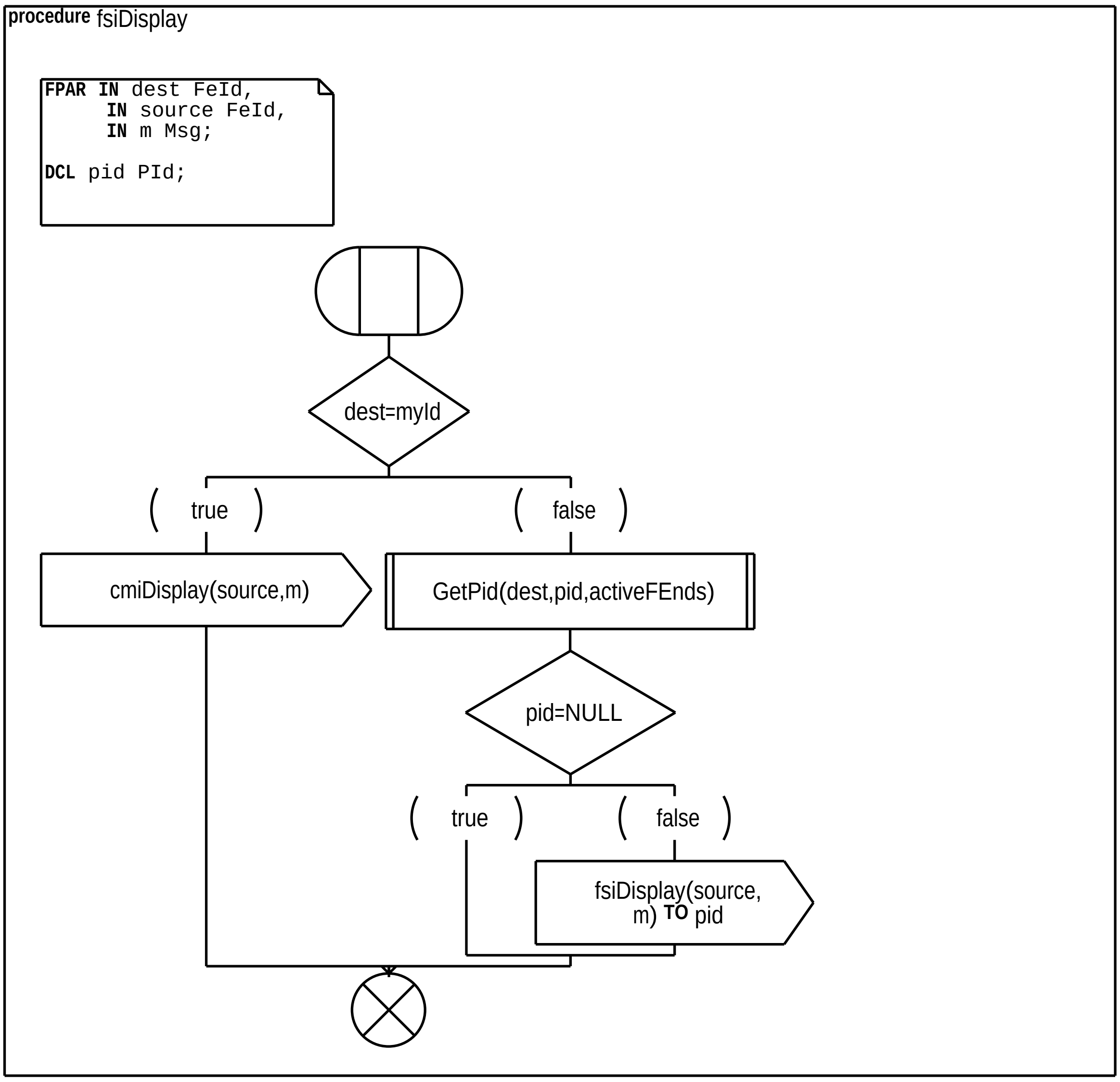

macrodefinition tcoDispatch FPAR aMessage
tcoCounter:=1
aMessage
routingElem:=
Extract(RunningTcos,tcoCounter)
aMessage
TO routingElem!pid
tcoCounter =
Length(RunningTcos)
true
false
testState!tcoErrors:=empty
allFesOK:=true
allTcosOK:=true
tcoCounter:=1
tcoCounter:=
tcoCounter+1
aMessage

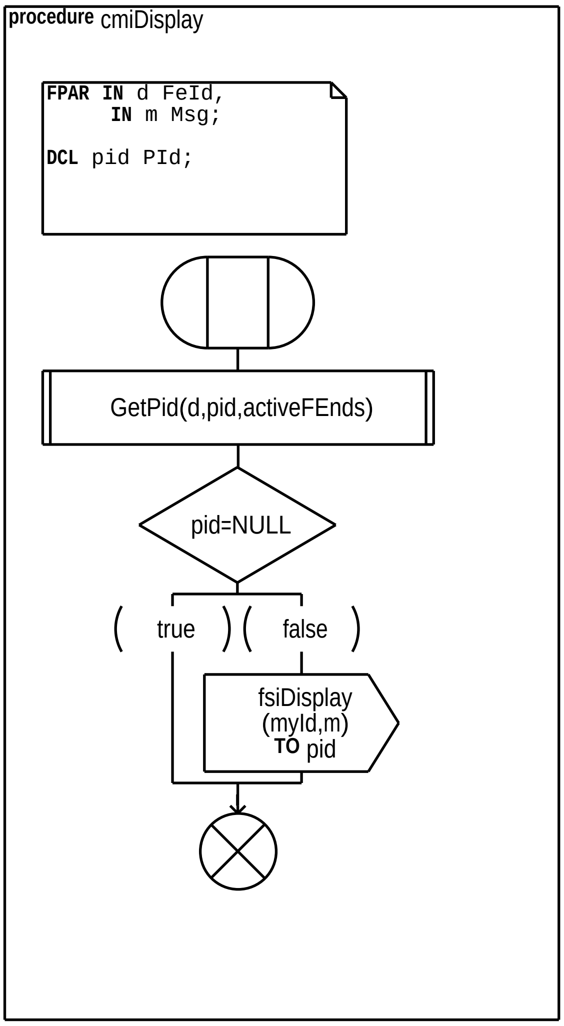
service SuCommon

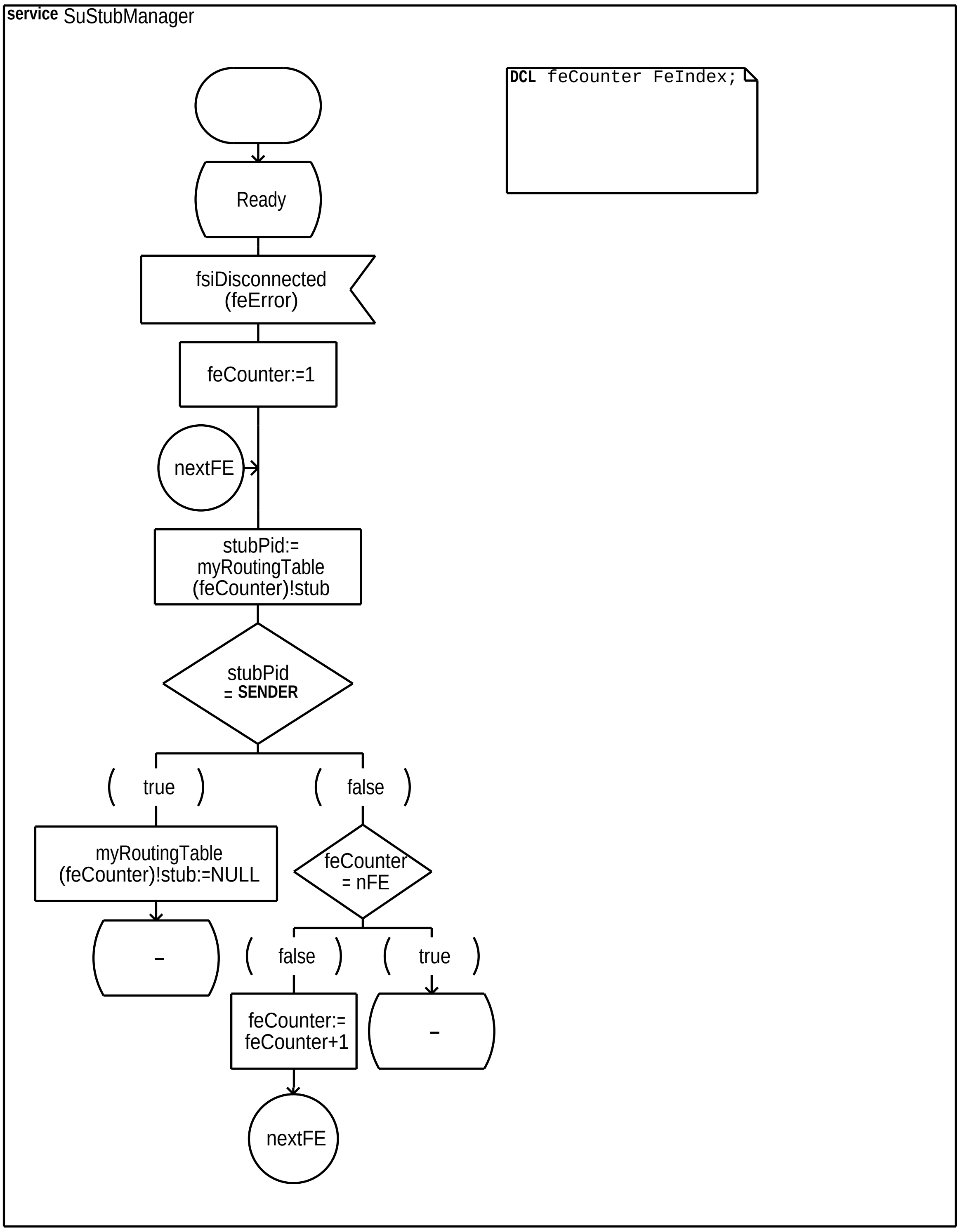


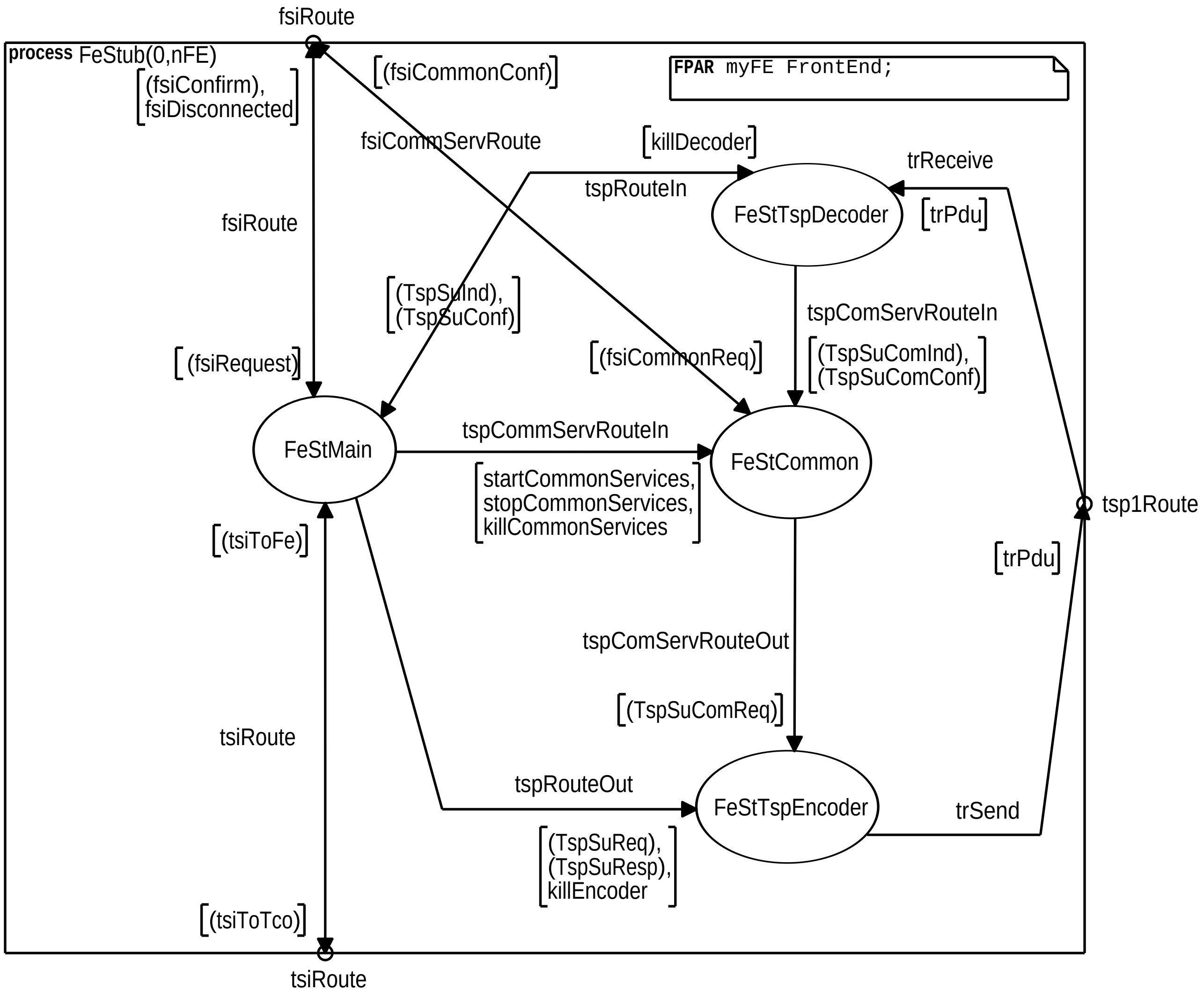
```
DCL askTrace TspAskTracePdu;
DCL askedTrace TspAskTraceAckPdu;
DCL listedServices TspListServicesAckPdu;
```


Running
cmiAskTrace (feId,askTrace)
fsiAskedTrace (askedTrace,feError)
cmiListServices (feId)
fsiListedServices (listedServices,feError)
GetPid(feId,stubPid, activeFEnds)
cmiAskedTrace (askedTrace,feError)
GetPid(feId,stubPid, activeFEnds)
cmiListedServices (listedServices,feError)
stubPid = NULL
Running
stubPid = NULL
Running
true
false
true
false
cmiAskedTrace (,Make(feId,errStArg, errOK,0))
fsiAskTrace(askTrace) TO stubPid
cmiListedServices (,Make(feId,errStArg, errOK,0))
fsiListServices TO stubPid
-
-

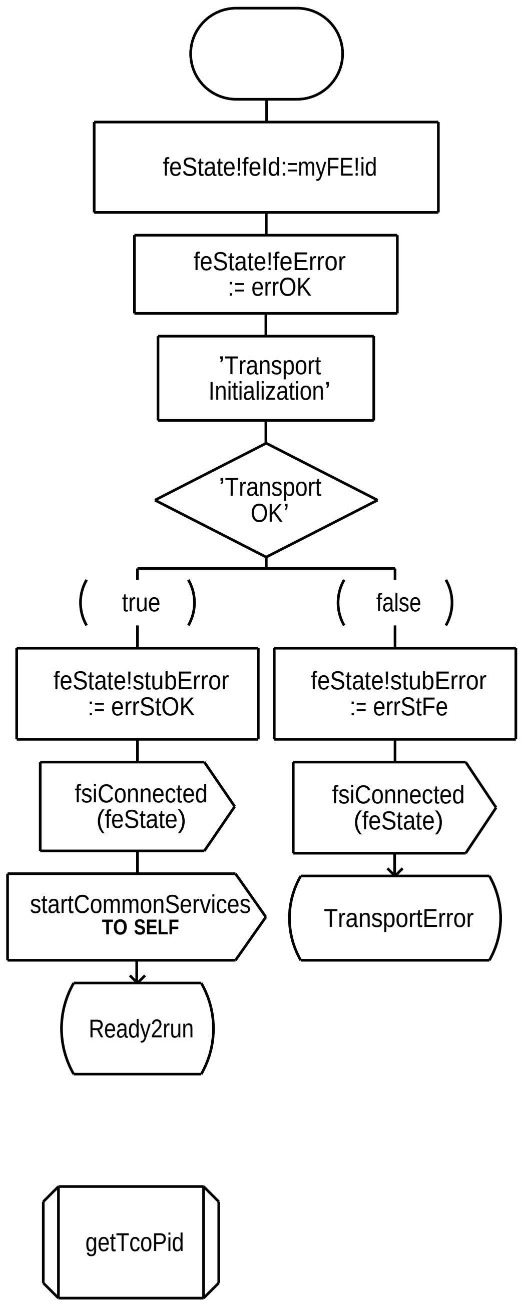

procedure fsiDisplay
FPAR IN dest FeId,
IN source FeId,
IN m Msg;
DCL pid PId;
dest=myId
true
false
cmiDisplay(source,m)
GetPid(dest,pid,activeFEnds)
pid=NULL
true
false
fsiDisplay(source, m) TO pid


procedure cmiDisplay
FPAR IN d FeId,
IN m Msg;
DCL pid PId;
GetPid(d,pid,activeFEnds)
pid=NULL
true
false
fsiDisplay (myId,m) **TO** pid

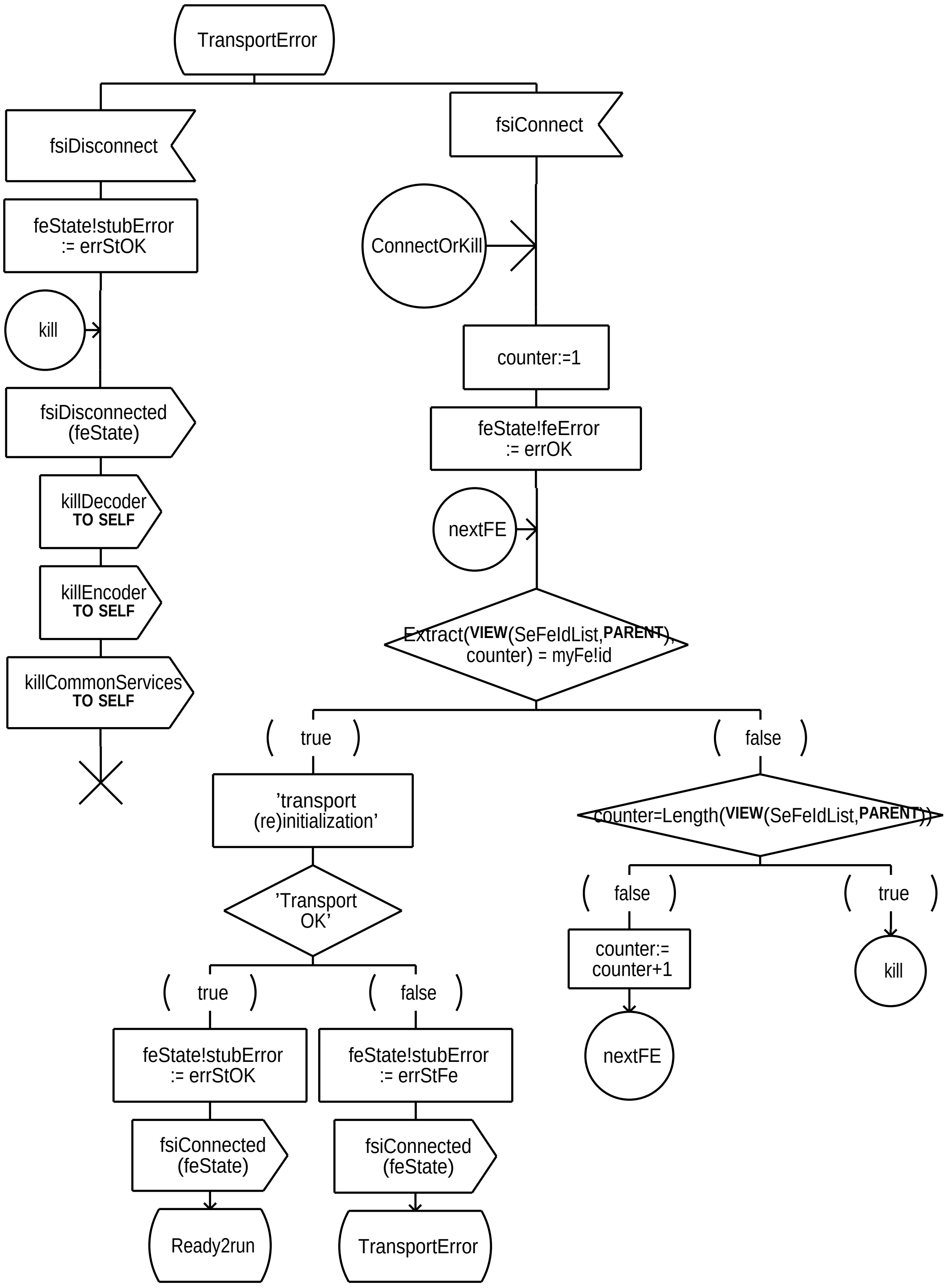

service SuStubManager
DCL feCounter FeIndex;
Ready
fsiDisconnected (feError)
feCounter:=1
nextFE
stubPid:= myRoutingTable (feCounter)!stub
stubPid = SENDER
true
false
myRoutingTable (feCounter)!stub:=NULL
-
feCounter = nFE
false
true
feCounter:= feCounter+1
-
nextFE

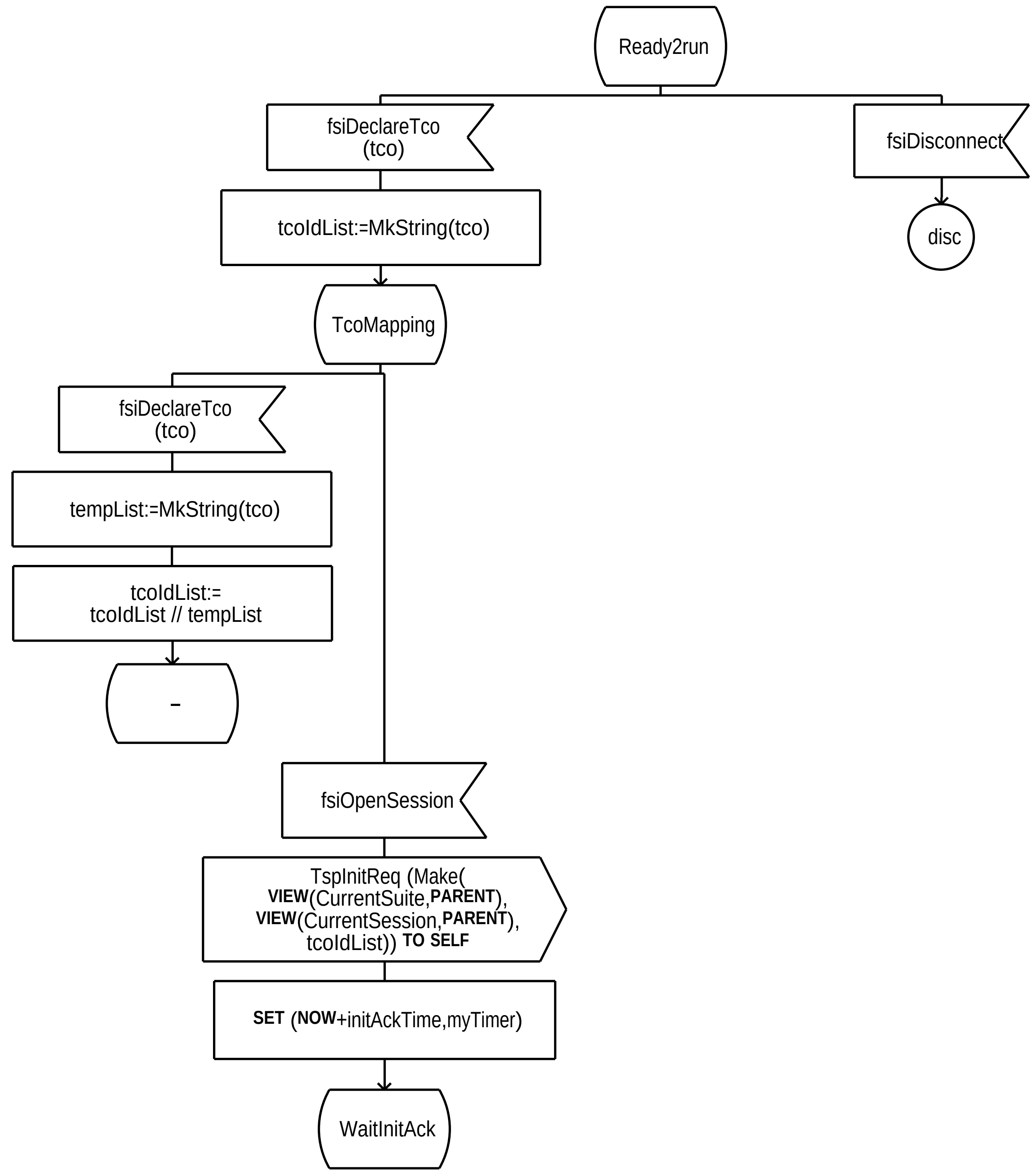

fsiRoute
process FeStub(0,nFE)
(fsiConfirm), fsiDisconnected
(fsiCommonConf)
FPAR myFE FrontEnd;
fsiCommServRoute
killDecoder
trReceive
tspRouteIn
FeStTspDecoder
trPdu
fsiRoute
(TspSuInd), (TspSuConf)
tspComServRouteIn
(fsiRequest)
(fsiCommonReq)
(TspSuComInd), (TspSuComConf)
tspCommServRouteIn
FeStMain
FeStCommon
tsp1Route
startCommonServices, stopCommonServices, killCommonServices
(tsiToFe)
trPdu
tspComServRouteOut
(TspSuComReq)
tsiRoute
tspRouteOut
FeStTspEncoder
trSend
(TspSuReq), (TspSuResp), killEncoder
(tsiToTco)
tsiRoute

service FeStMain

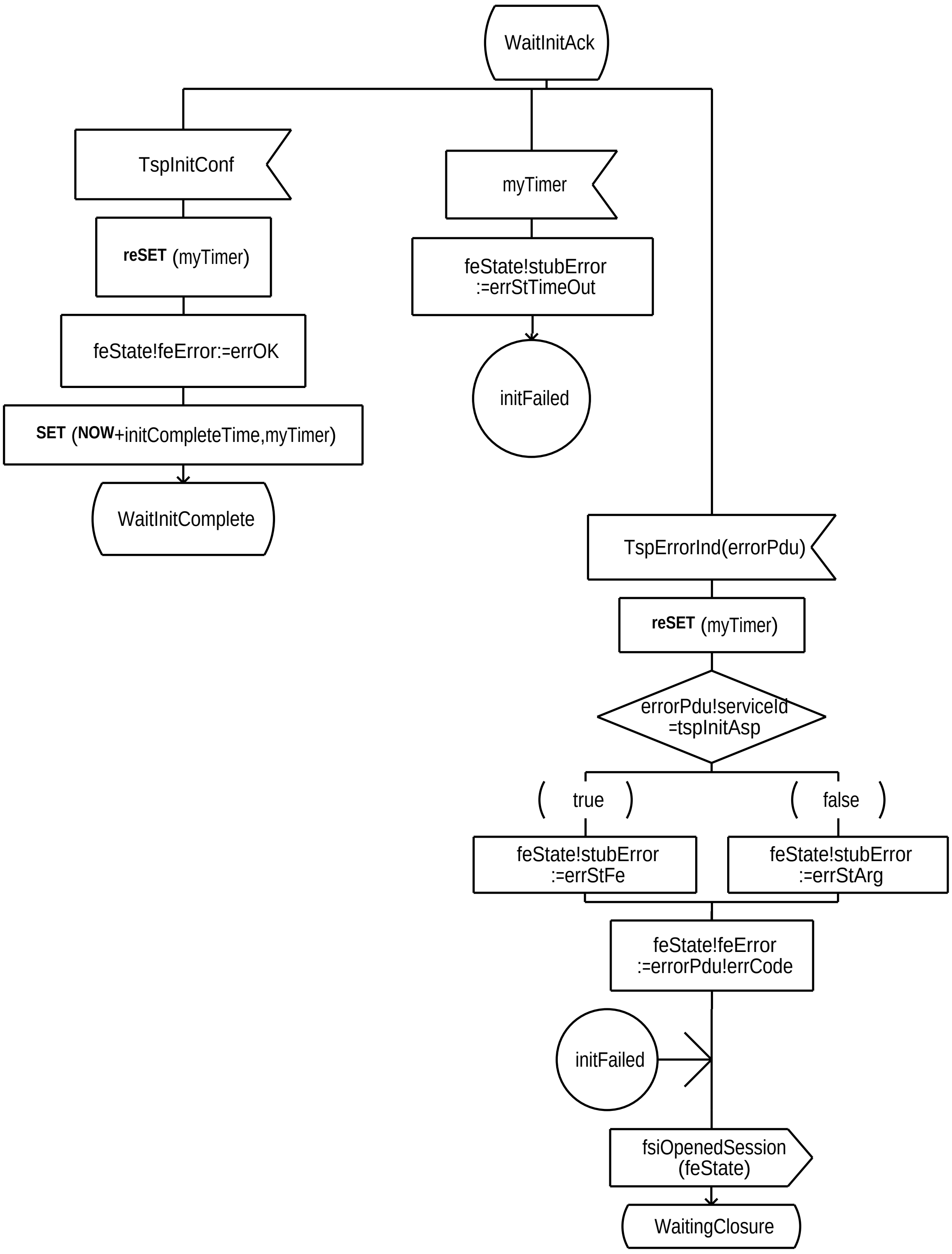


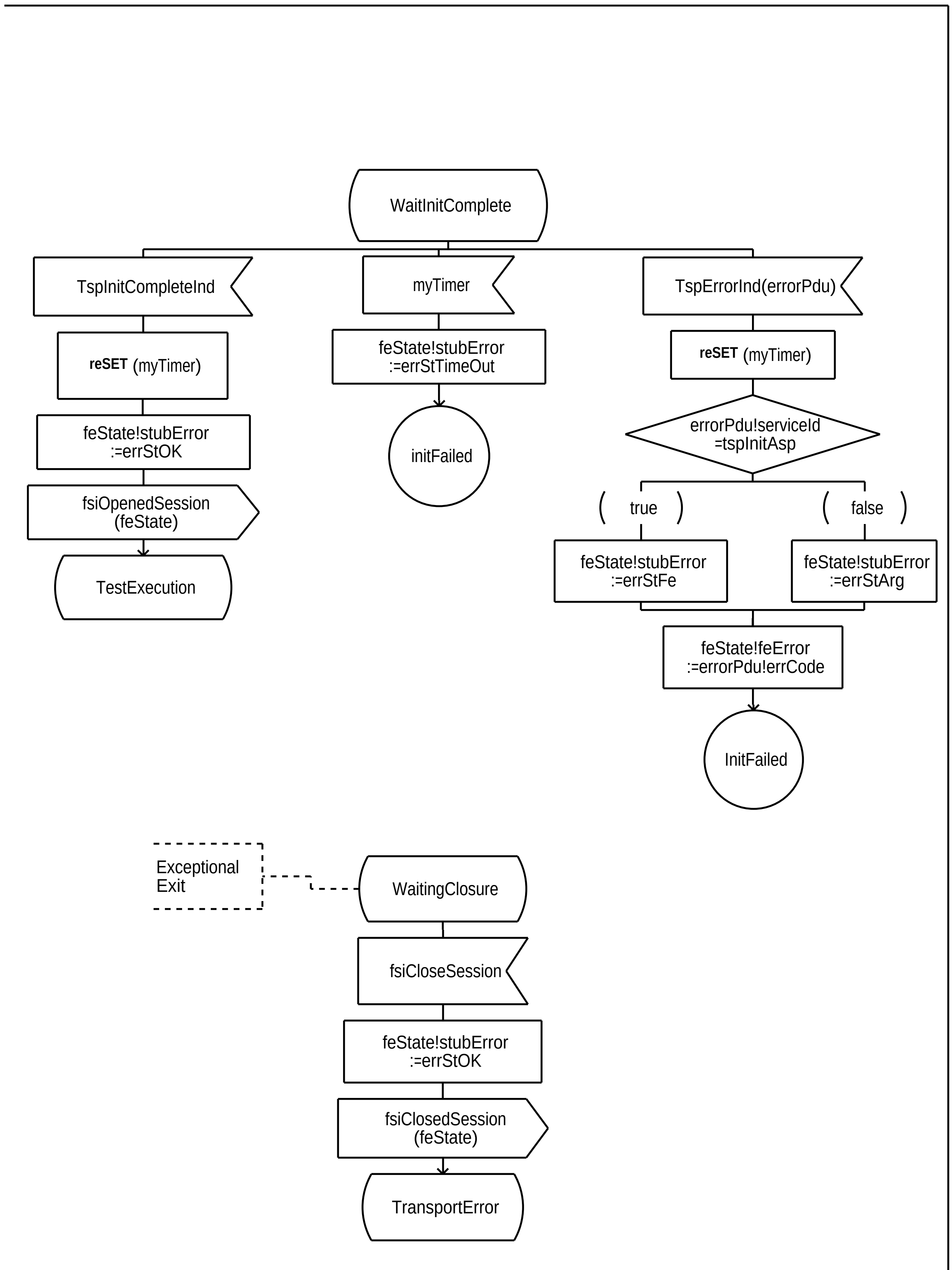
```
VIEWED CurrentSession SessionId;
VIEWED CurrentSuite EtsId;
VIEWED Parameters ParamList;
VIEWED Time TIME;
VIEWED RunningTcos RoutingList;
/* VIEWED Mtc MtcId; */
VIEWED SeFeIdList FeIdList;

DCL feState FeError;
TIMER myTimer;

DCL test TestCaseId;
DCL tco TcoId;
DCL cp CpId;
DCL msg Msg;
DCL tree Tree;
DCL param ParamList;
DCL tcoPid PId;

DCL tcoIdList, tempList TcoIdList;
DCL errorPdu TspErrorPdu;
DCL counter FeIndex;

DCL tspCreate TspCreatePdu;
DCL tspCreateAck TspCreateAckPdu;

DCL tspVerdict TspVerdictPdu;
DCL verdict VerdictValue;

DCL tspInfo TspInfoPdu;

DCL tspCancelOpAck TspCancelOpAckPdu;
```


TransportError
fsiDisconnect
feState!stubError := errStOK
kill
fsiDisconnected (feState)
killDecoder TO SELF
killEncoder TO SELF
killCommonServices TO SELF
fsiConnect
ConnectOrKill
counter:=1
feState!feError := errOK
nextFE
Extract(VIEW(SeFeIdList,PARENT), counter) = myFe!id
true
false
'transport (re)initialization'
'Transport OK'
true
false
feState!stubError := errStOK
feState!stubError := errStFe
fsiConnected (feState)
fsiConnected (feState)
Ready2run
TransportError
counter=Length(VIEW(SeFeIdList,PARENT))
false
true
counter:= counter+1
kill
nextFE

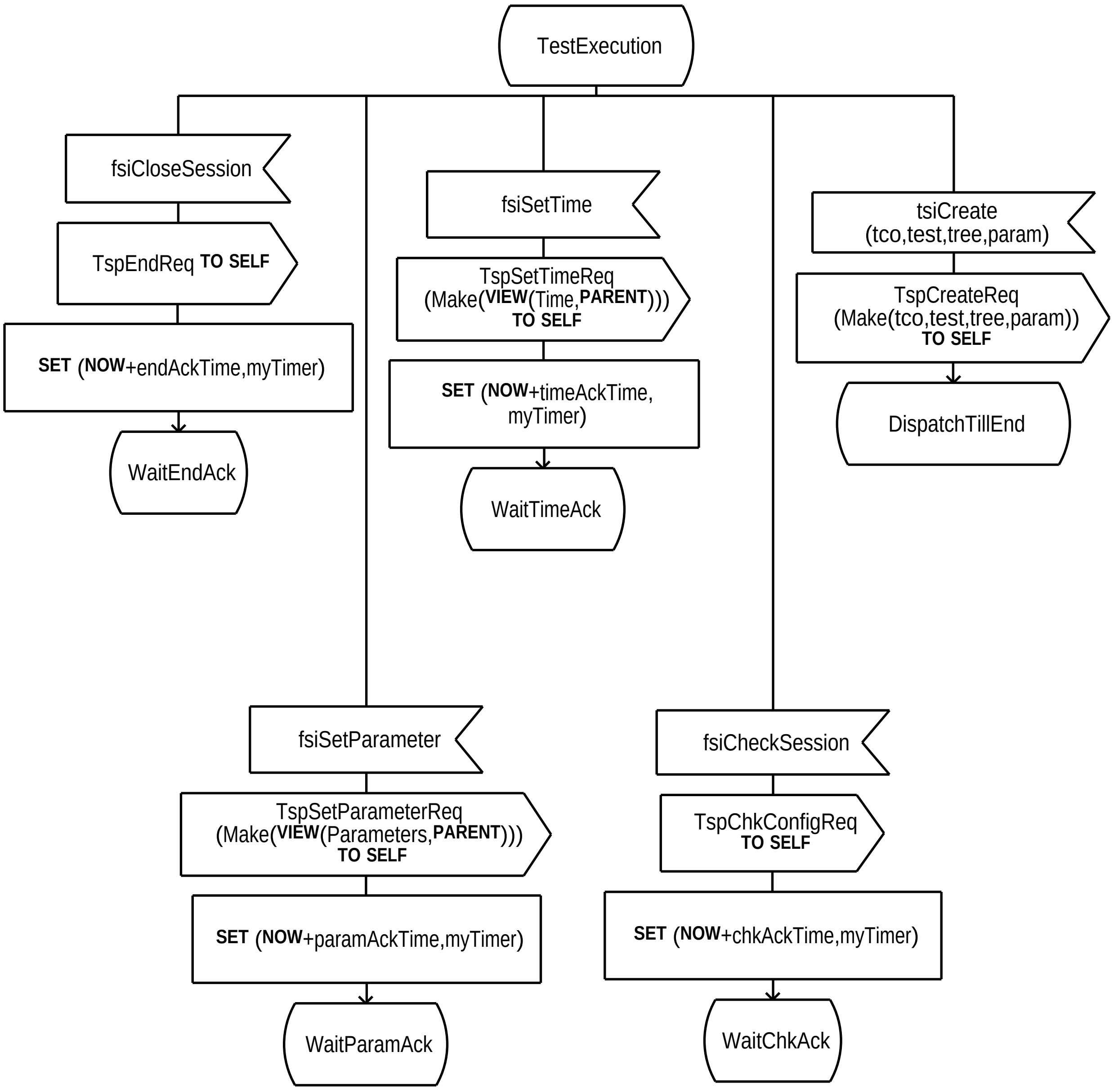


Ready2run
fsiDeclareTco (tco)
tcoIdList:=MkString(tco)
TcoMapping
fsiDisconnect
disc
fsiDeclareTco (tco)
tempList:=MkString(tco)
tcoIdList:= tcoIdList // tempList
–
fsiOpenSession
TspInitReq (Make(VIEW(CurrentSuite,PARENT), VIEW(CurrentSession,PARENT), tcoIdList)) TO SELF
SET (NOW+initAckTime,myTimer)
WaitInitAck

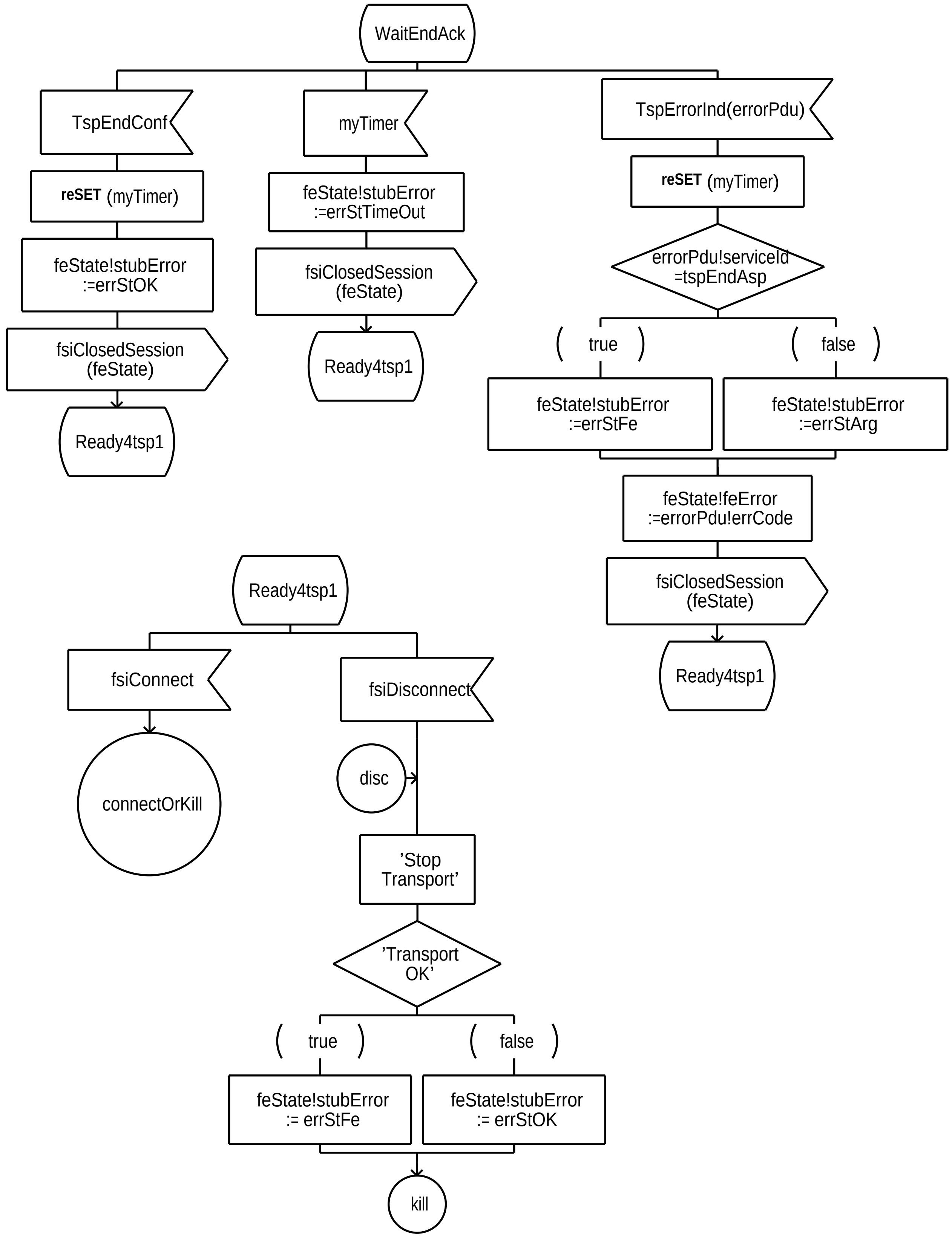

WaitInitAck
TspInitConf
reSET (myTimer)
feState!feError:=errOK
SET (NOW+initCompleteTime,myTimer)
WaitInitComplete
myTimer
feState!stubError
:=errStTimeOut
initFailed
TspErrorInd(errorPdu)
reSET (myTimer)
errorPdu!serviceId
=tspInitAsp
true
false
feState!stubError
:=errStFe
feState!stubError
:=errStArg
feState!feError
:=errorPdu!errCode
initFailed
fsiOpenedSession
(feState)
WaitingClosure

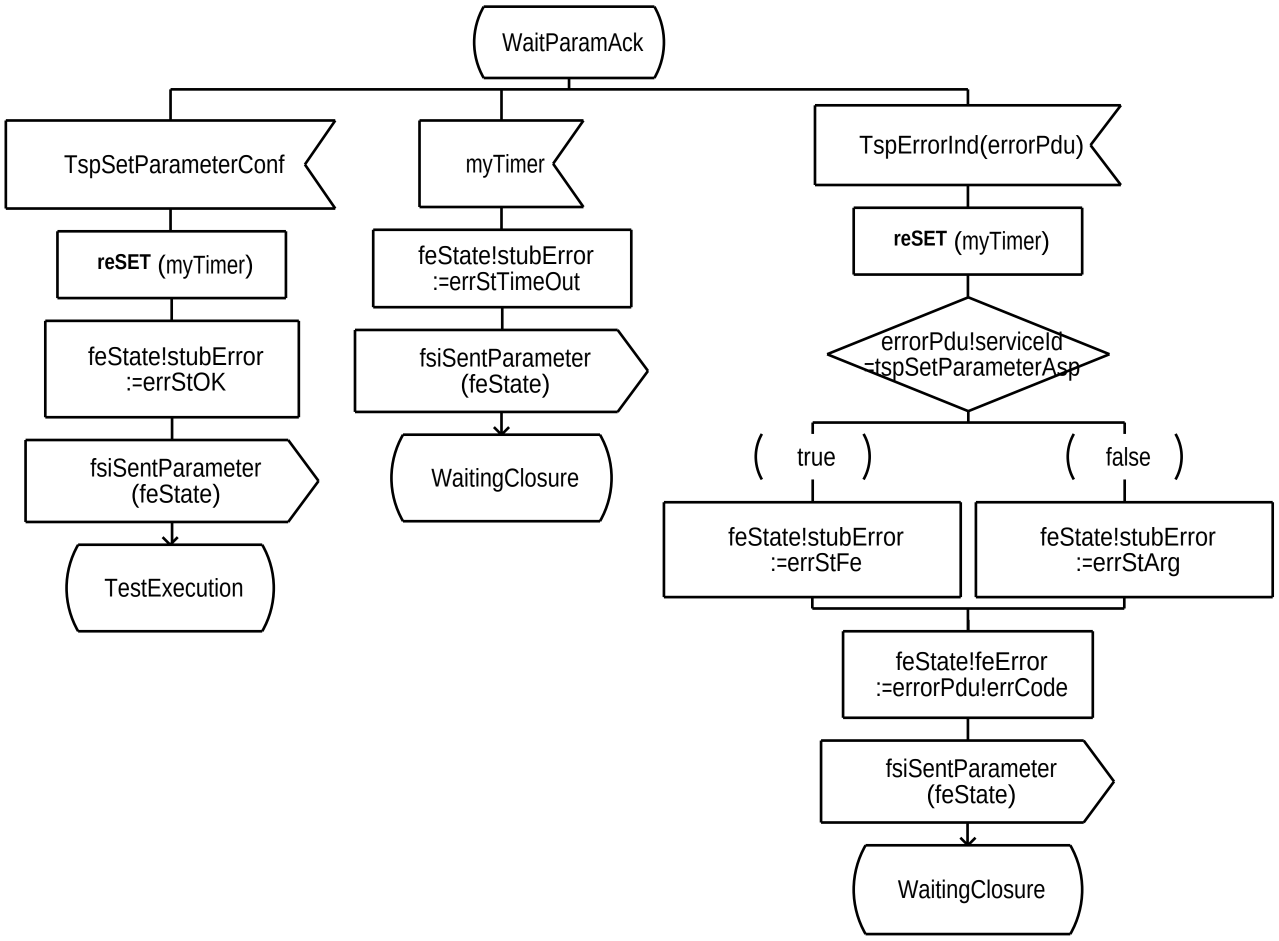


WaitInitComplete
TspInitCompleteInd
myTimer
TspErrorInd(errorPdu)
reSET (myTimer)
feState!stubError :=errStTimeOut
reSET (myTimer)
feState!stubError :=errStOK
initFailed
errorPdu!serviceId =tspInitAsp
fsiOpenedSession (feState)
true
false
TestExecution
feState!stubError :=errStFe
feState!stubError :=errStArg
feState!feError :=errorPdu!errCode
InitFailed
Exceptional Exit
WaitingClosure
fsiCloseSession
feState!stubError :=errStOK
fsiClosedSession (feState)
TransportError

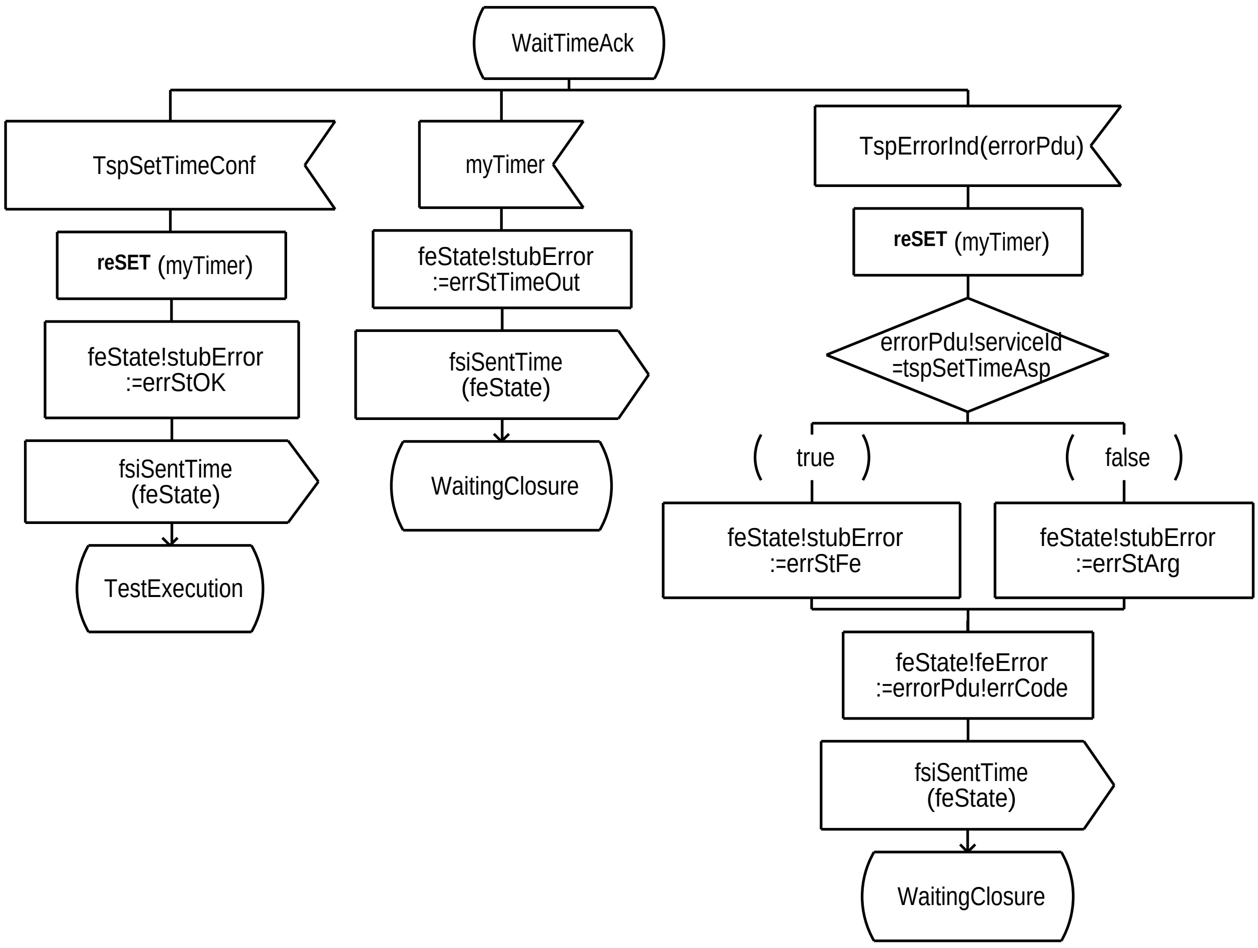


TestExecution
fsiCloseSession
TspEndReq TO SELF
SET (NOW+endAckTime,myTimer)
WaitEndAck
fsiSetTime
TspSetTimeReq (Make(VIEW(Time,PARENT))) TO SELF
SET (NOW+timeAckTime, myTimer)
WaitTimeAck
tsiCreate (tco,test,tree,param)
TspCreateReq (Make(tco,test,tree,param)) TO SELF
DispatchTillEnd
fsiSetParameter
TspSetParameterReq (Make(VIEW(Parameters,PARENT))) TO SELF
SET (NOW+paramAckTime,myTimer)
WaitParamAck
fsiCheckSession
TspChkConfigReq TO SELF
SET (NOW+chkAckTime,myTimer)
WaitChkAck

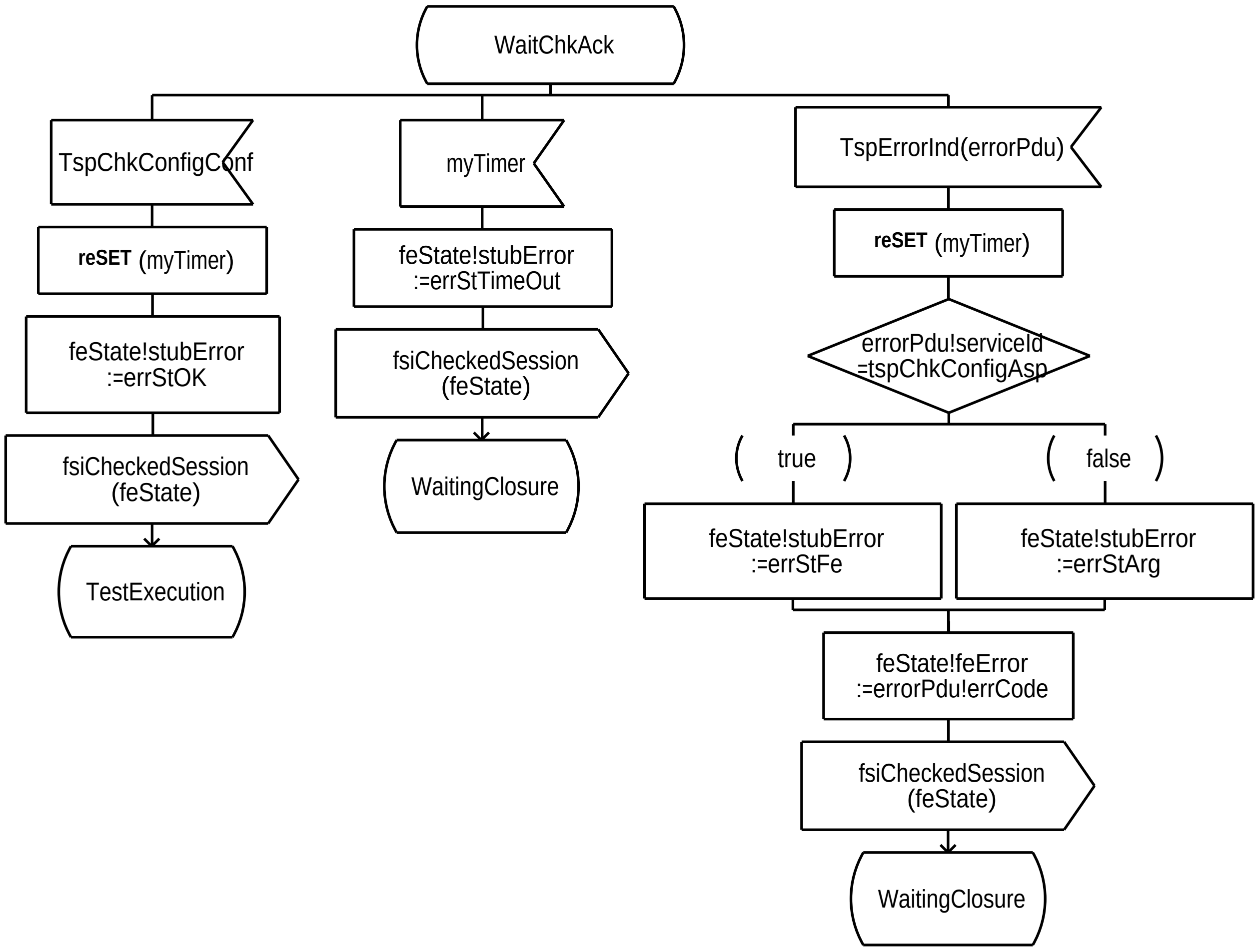

WaitEndAck
TspEndConf
reSET (myTimer)
feState!stubError :=errStOK
fsiClosedSession (feState)
Ready4tsp1
myTimer
feState!stubError :=errStTimeOut
fsiClosedSession (feState)
Ready4tsp1
TspErrorInd(errorPdu)
reSET (myTimer)
errorPdu!serviceId =tspEndAsp
true
false
feState!stubError :=errStFe
feState!stubError :=errStArg
feState!feError :=errorPdu!errCode
fsiClosedSession (feState)
Ready4tsp1
Ready4tsp1
fsiConnect
connectOrKill
fsiDisconnect
disc
'Stop Transport'
'Transport OK'
true
false
feState!stubError := errStFe
feState!stubError := errStOK
kill



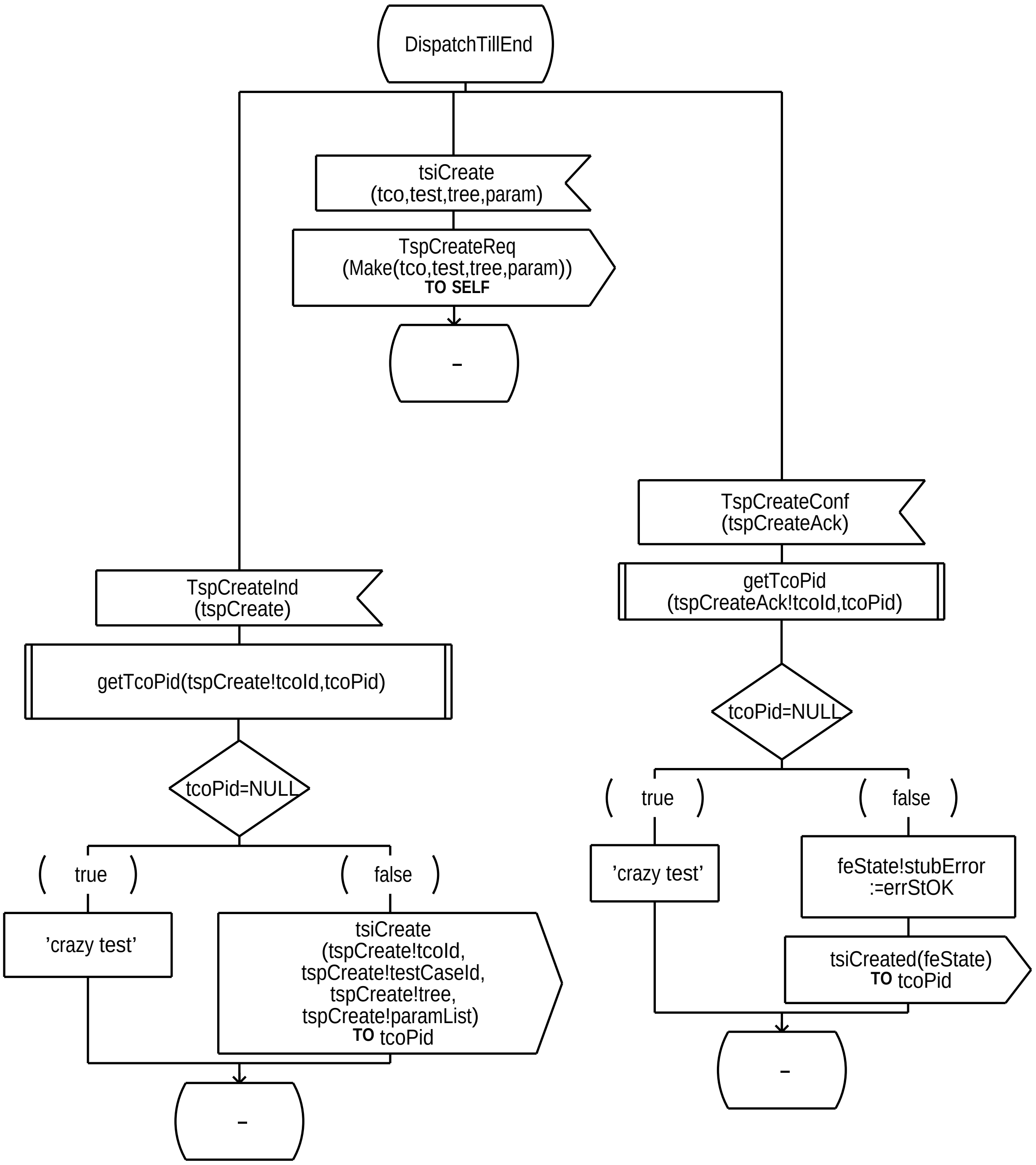
WaitParamAck
TspSetParameterConf
myTimer
TspErrorInd(errorPdu)
reSET (myTimer)
feState!stubError :=errStTimeOut
reSET (myTimer)
feState!stubError :=errStOK
fsiSentParameter (feState)
errorPdu!serviceId =tspSetParameterAsp
fsiSentParameter (feState)
WaitingClosure
true
false
TestExecution
feState!stubError :=errStFe
feState!stubError :=errStArg
feState!feError :=errorPdu!errCode
fsiSentParameter (feState)
WaitingClosure

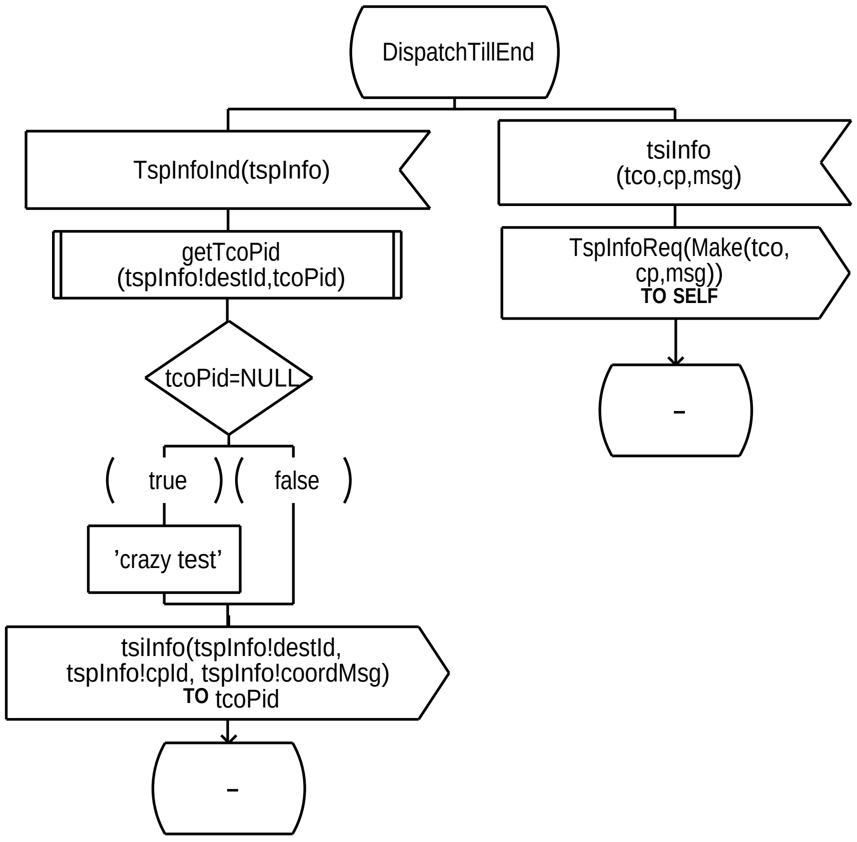


WaitTimeAck
TspSetTimeConf
myTimer
TspErrorInd(errorPdu)
reSET (myTimer)
feState!stubError :=errStOK
fsiSentTime (feState)
TestExecution
feState!stubError :=errStTimeOut
fsiSentTime (feState)
WaitingClosure
reSET (myTimer)
errorPdu!serviceId =tspSetTimeAsp
true
false
feState!stubError :=errStFe
feState!stubError :=errStArg
feState!feError :=errorPdu!errCode
fsiSentTime (feState)
WaitingClosure

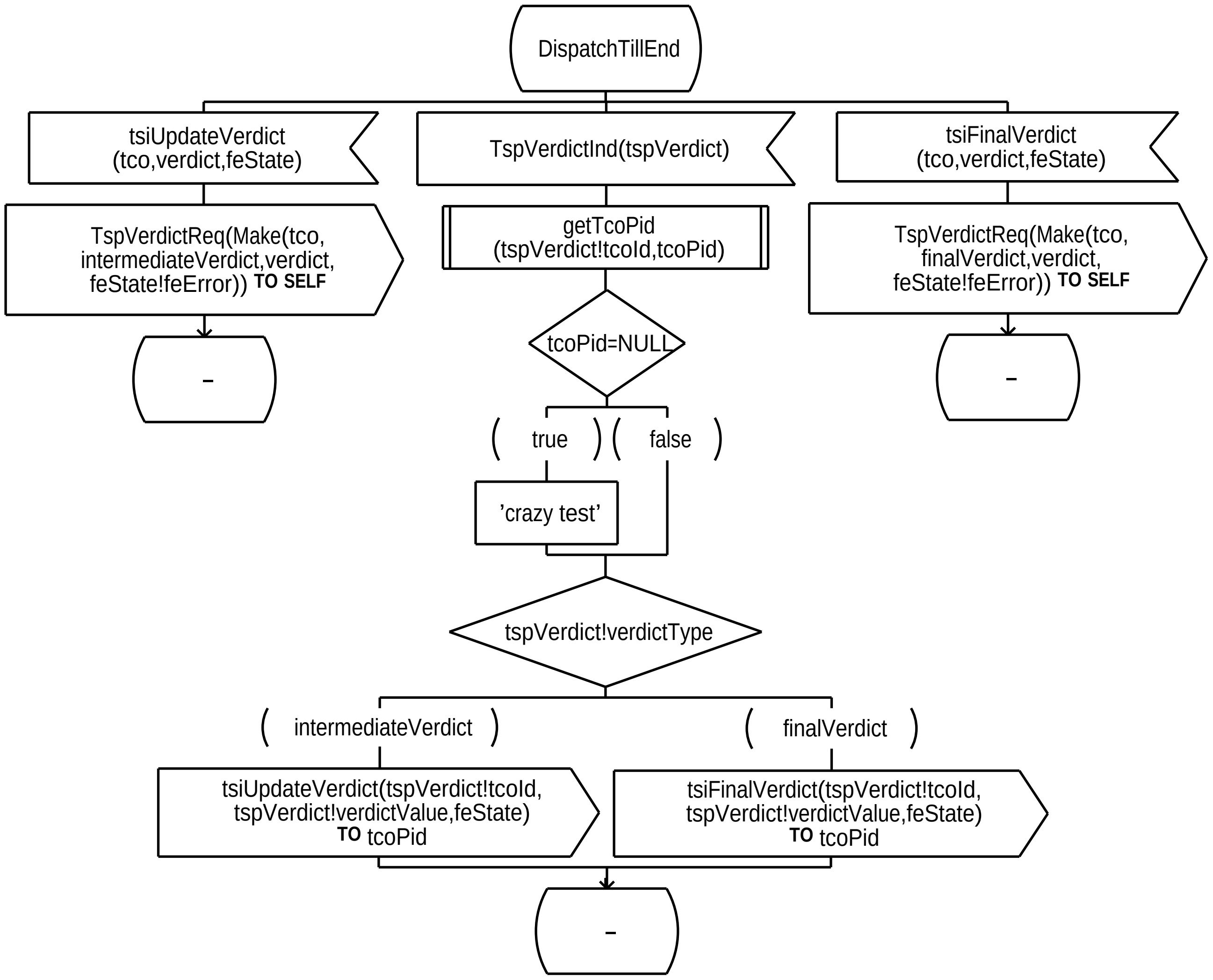


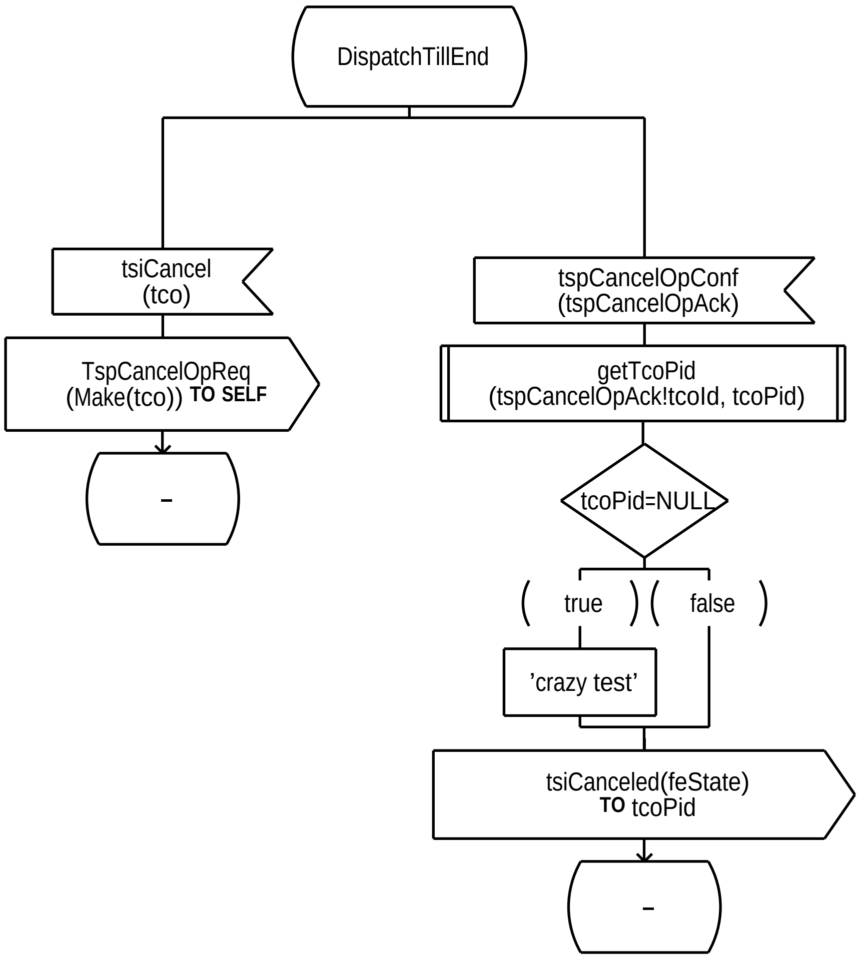
WaitChkAck
TspChkConfigConf
reSET (myTimer)
feState!stubError :=errStOK
fsiCheckedSession (feState)
TestExecution
myTimer
feState!stubError :=errStTimeOut
fsiCheckedSession (feState)
WaitingClosure
TspErrorInd(errorPdu)
reSET (myTimer)
errorPdu!serviceId =tspChkConfigAsp
true
false
feState!stubError :=errStFe
feState!stubError :=errStArg
feState!feError :=errorPdu!errCode
fsiCheckedSession (feState)
WaitingClosure

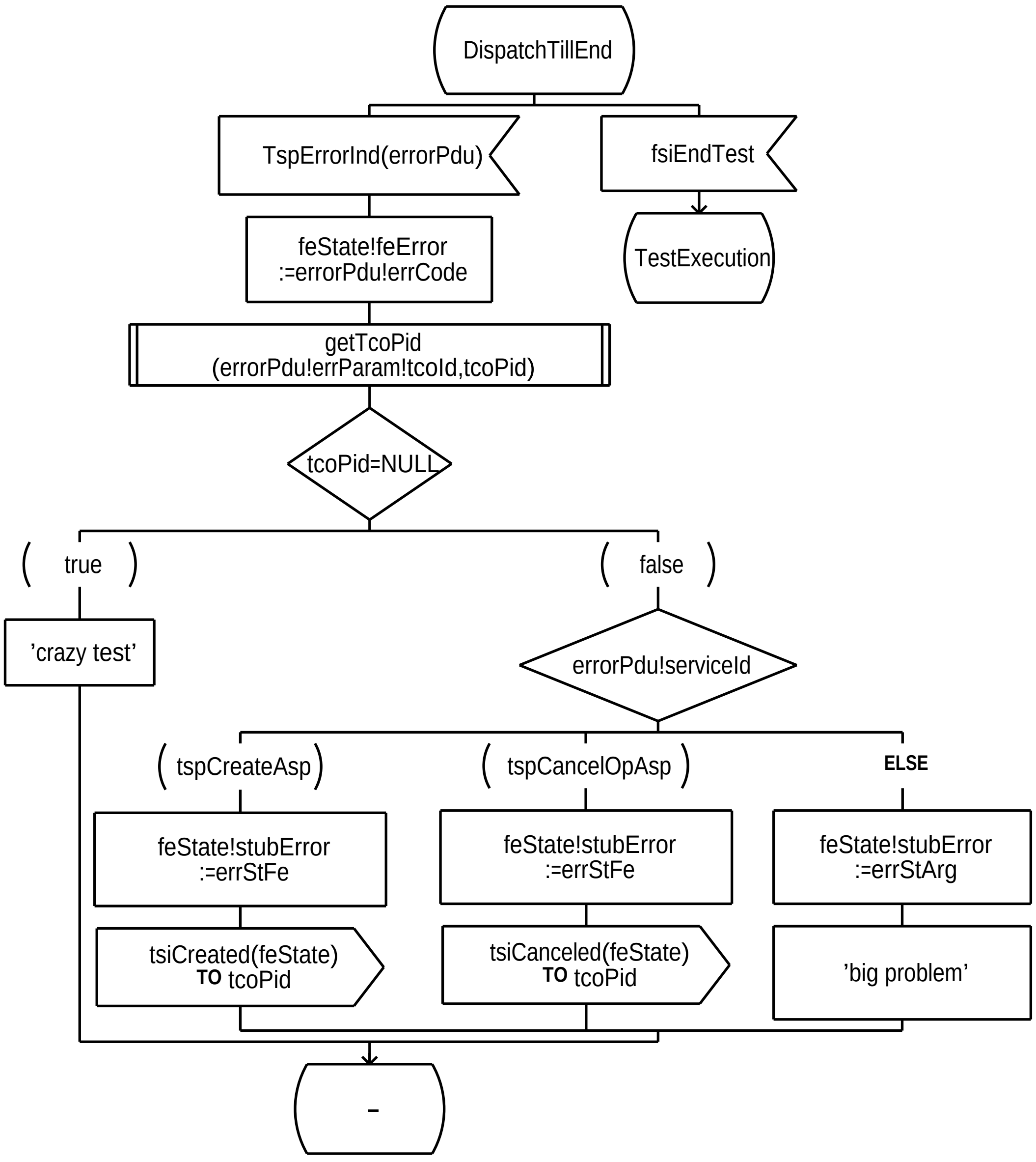

DispatchTillEnd
tsiCreate
(tco,test,tree,param)
TspCreateReq
(Make(tco,test,tree,param))
TO SELF
-
TspCreateConf
(tspCreateAck)
getTcoPid
(tspCreateAck!tcoId,tcoPid)
tcoPid=NULL
true
false
'crazy test'
feState!stubError
:=errStOK
tsiCreated(feState)
TO tcoPid
-
TspCreateInd
(tspCreate)
getTcoPid(tspCreate!tcoId,tcoPid)
tcoPid=NULL
true
false
'crazy test'
tsiCreate
(tspCreate!tcoId,
tspCreate!testCaseId,
tspCreate!tree,
tspCreate!paramList)
TO tcoPid
-

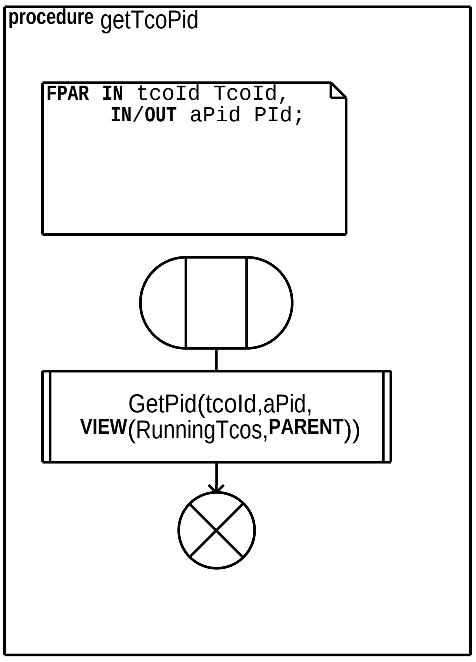


DispatchTillEnd
TspInfoInd(tspInfo)
getTcoPid
(tspInfo!destId,tcoPid)
tcoPid=NULL
true
false
'crazy test'
tsiInfo(tspInfo!destId,
tspInfo!cpId, tspInfo!coordMsg)
TO tcoPid
-
tsiInfo
(tco,cp,msg)
TspInfoReq(Make(tco,
cp,msg))
TO SELF
-

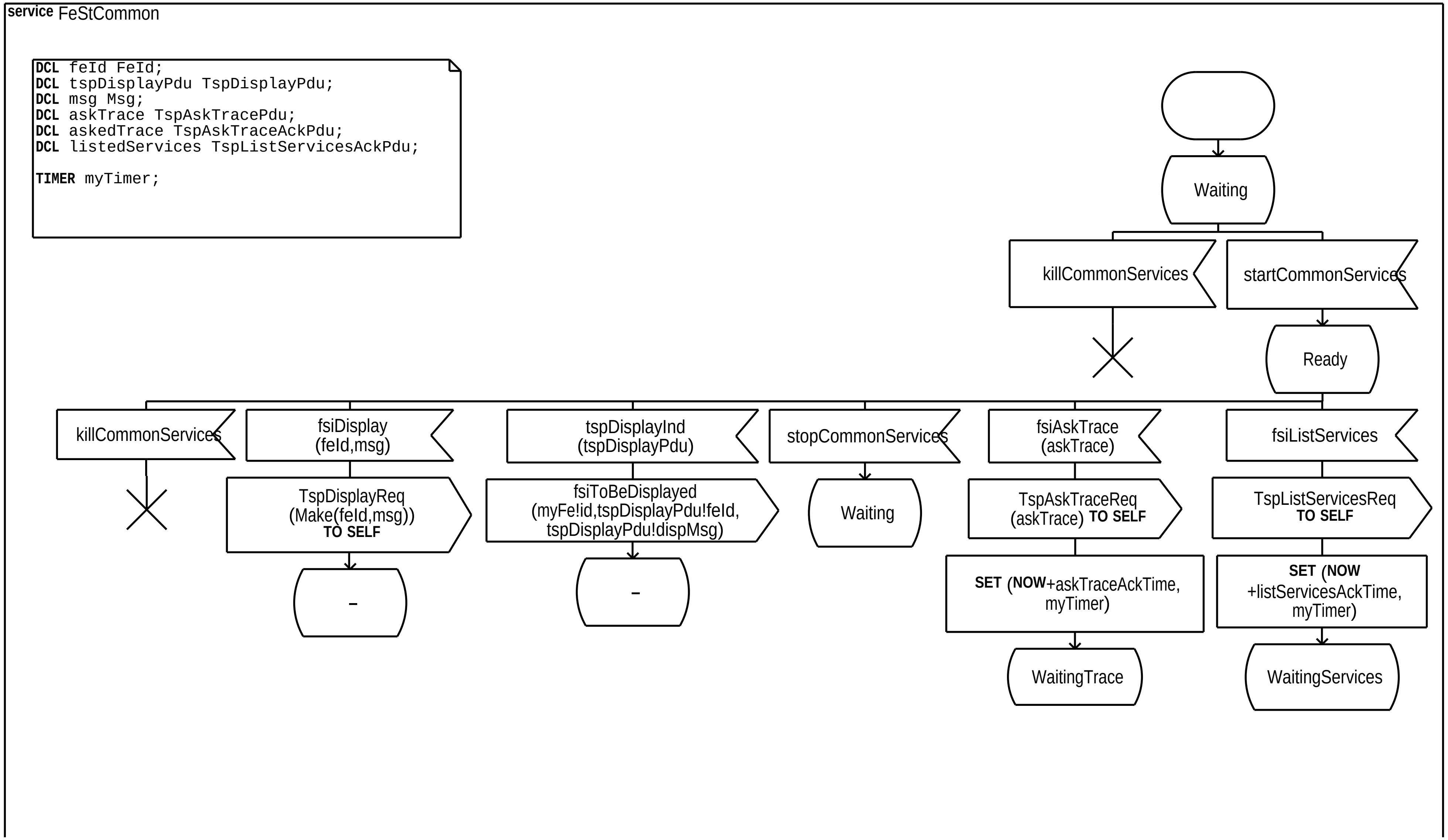

DispatchTillEnd
tsiUpdateVerdict (tco,verdict,feState)
TspVerdictInd(tspVerdict)
tsiFinalVerdict (tco,verdict,feState)
TspVerdictReq(Make(tco, intermediateVerdict,verdict, feState!feError)) TO SELF
getTcoPid (tspVerdict!tcoId,tcoPid)
TspVerdictReq(Make(tco, finalVerdict,verdict, feState!feError)) TO SELF
-
tcoPid=NULL
-
true
false
'crazy test'
tspVerdict!verdictType
intermediateVerdict
finalVerdict
tsiUpdateVerdict(tspVerdict!tcoId, tspVerdict!verdictValue,feState) TO tcoPid
tsiFinalVerdict(tspVerdict!tcoId, tspVerdict!verdictValue,feState) TO tcoPid
-


DispatchTillEnd
tsiCancel (tco)
TspCancelOpReq (Make(tco)) TO SELF
–
tspCancelOpConf (tspCancelOpAck)
getTcoPid (tspCancelOpAck!tcoId, tcoPid)
tcoPid=NULL
true
false
'crazy test'
tsiCanceled(feState) TO tcoPid
–

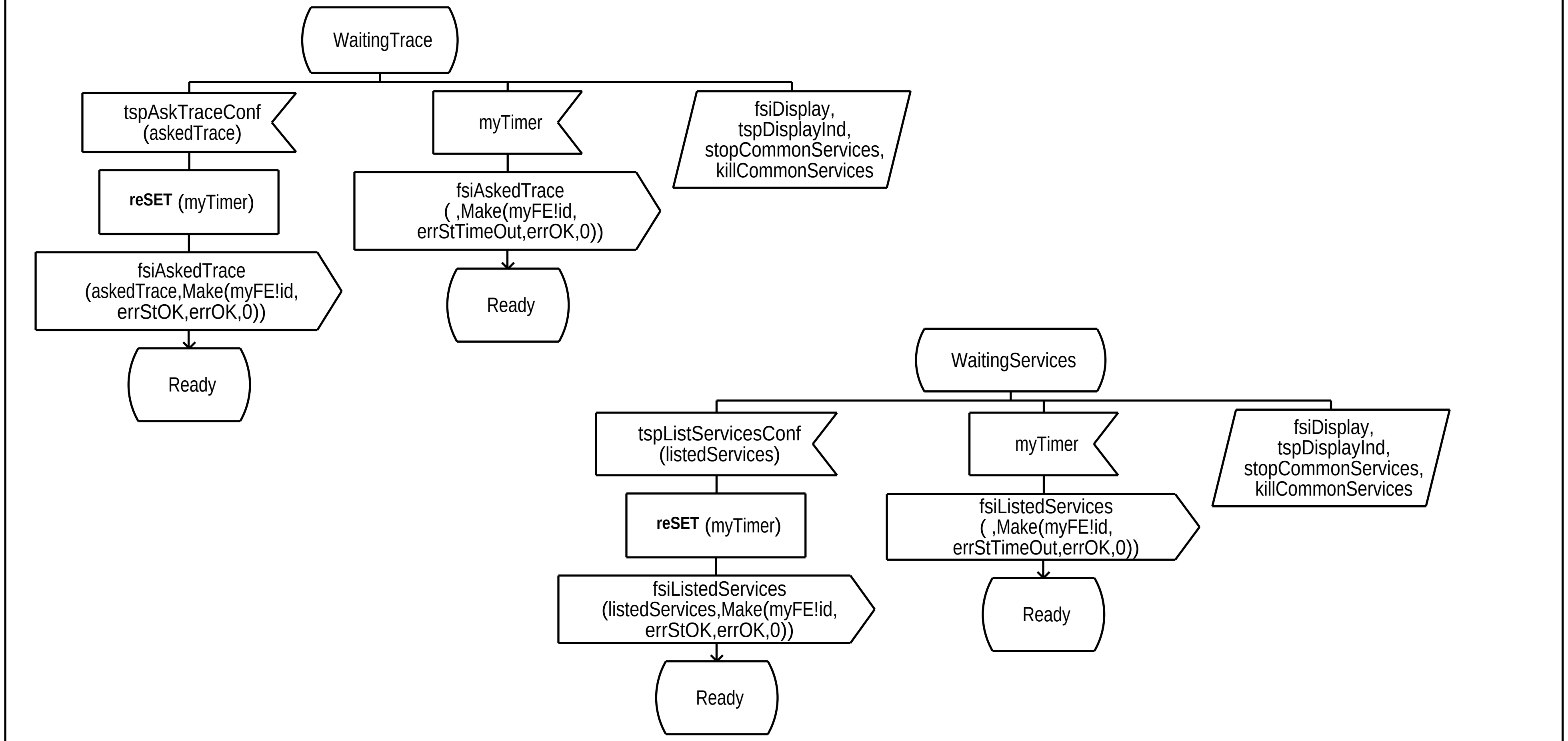

DispatchTillEnd
TspErrorInd(errorPdu)
fsiEndTest
TestExecution
feState!feError :=errorPdu!errCode
getTcoPid (errorPdu!errParam!tcoId,tcoPid)
tcoPid=NULL
true
false
'crazy test'
errorPdu!serviceId
tspCreateAsp
tspCancelOpAsp
ELSE
feState!stubError :=errStFe
feState!stubError :=errStFe
feState!stubError :=errStArg
tsiCreated(feState) TO tcoPid
tsiCanceled(feState) TO tcoPid
'big problem'
-

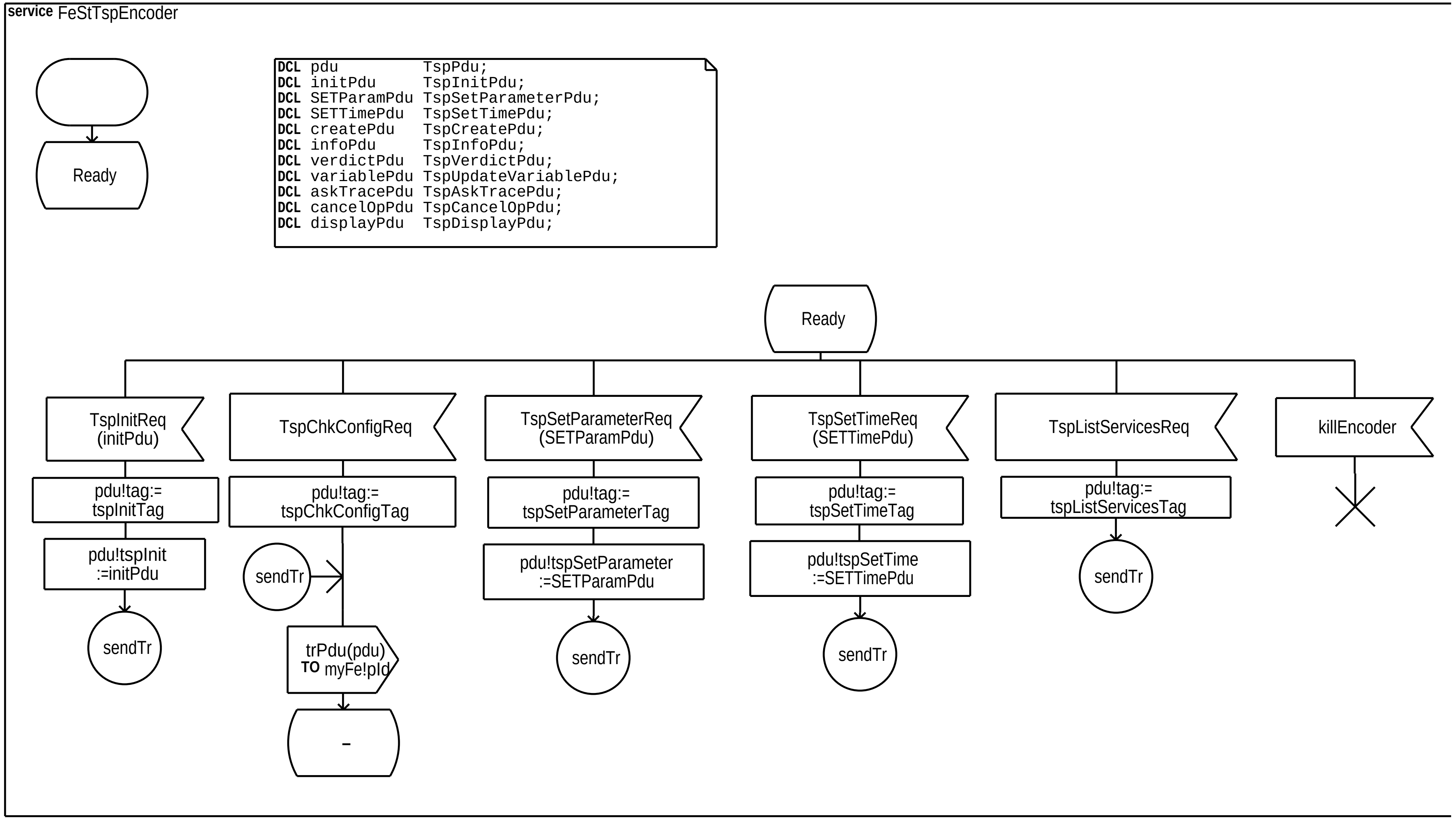


procedure getTcoPid
FPAR IN tcoId TcoId,
IN/OUT aPid PId;
GetPid(tcoId,aPid,
VIEW(RunningTcos,PARENT))

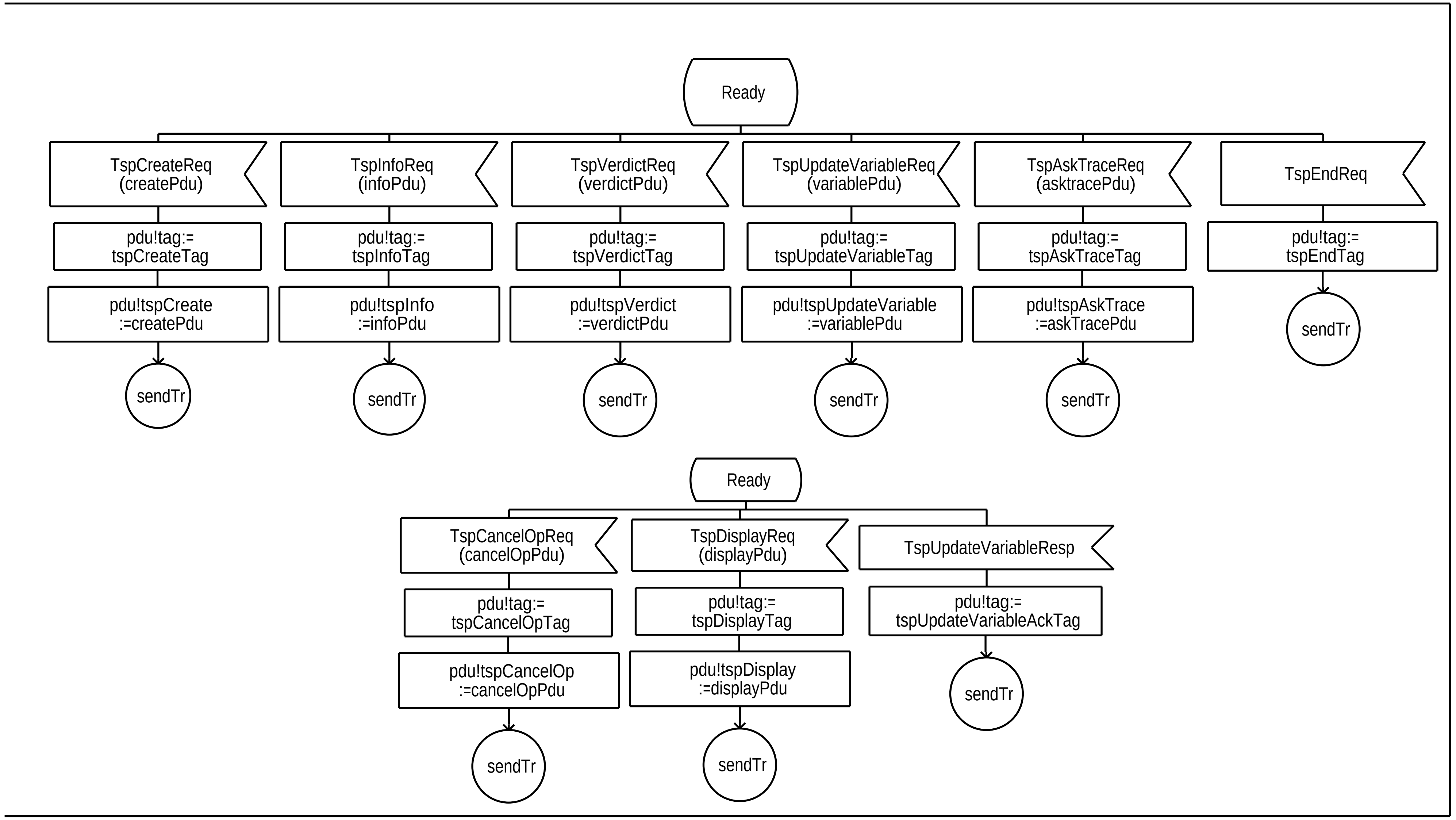

service FeStCommon
Waiting
killCommonServices
startCommonServices
Ready
killCommonServices
fsiDisplay (feId,msg)
TspDisplayReq (Make(feId,msg)) TO SELF
-
tspDisplayInd (tspDisplayPdu)
fsiToBeDisplayed (myFe!id,tspDisplayPdu!feId, tspDisplayPdu!dispMsg)
-
stopCommonServices
Waiting
fsiAskTrace (askTrace)
TspAskTraceReq (askTrace) TO SELF
SET (NOW+askTraceAckTime, myTimer)
WaitingTrace
fsiListServices
TspListServicesReq TO SELF
SET (NOW +listServicesAckTime, myTimer)
WaitingServices

```
DCL feId FeId;
DCL tspDisplayPdu TspDisplayPdu;
DCL msg Msg;
DCL askTrace TspAskTracePdu;
DCL askedTrace TspAskTraceAckPdu;
DCL listedServices TspListServicesAckPdu;

TIMER myTimer;
```

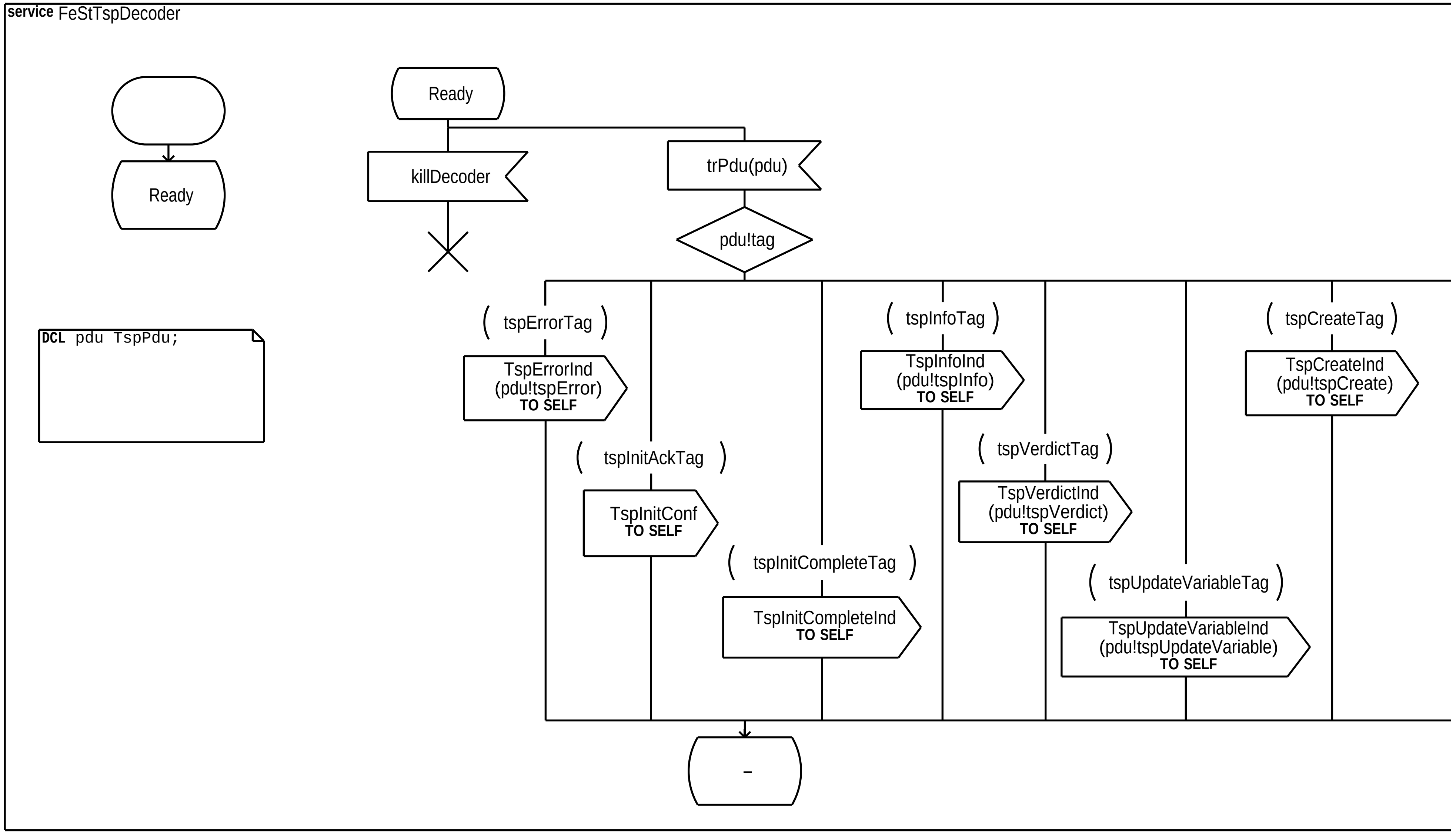

WaitingTrace
tspAskTraceConf (askedTrace)
reSET (myTimer)
fsiAskedTrace (askedTrace,Make(myFE!id, errStOK,errOK,0))
Ready
myTimer
fsiAskedTrace (,Make(myFE!id, errStTimeOut,errOK,0))
Ready
fsiDisplay, tspDisplayInd, stopCommonServices, killCommonServices
WaitingServices
tspListServicesConf (listedServices)
reSET (myTimer)
fsiListedServices (listedServices,Make(myFE!id, errStOK,errOK,0))
Ready
myTimer
fsiListedServices (,Make(myFE!id, errStTimeOut,errOK,0))
Ready
fsiDisplay, tspDisplayInd, stopCommonServices, killCommonServices

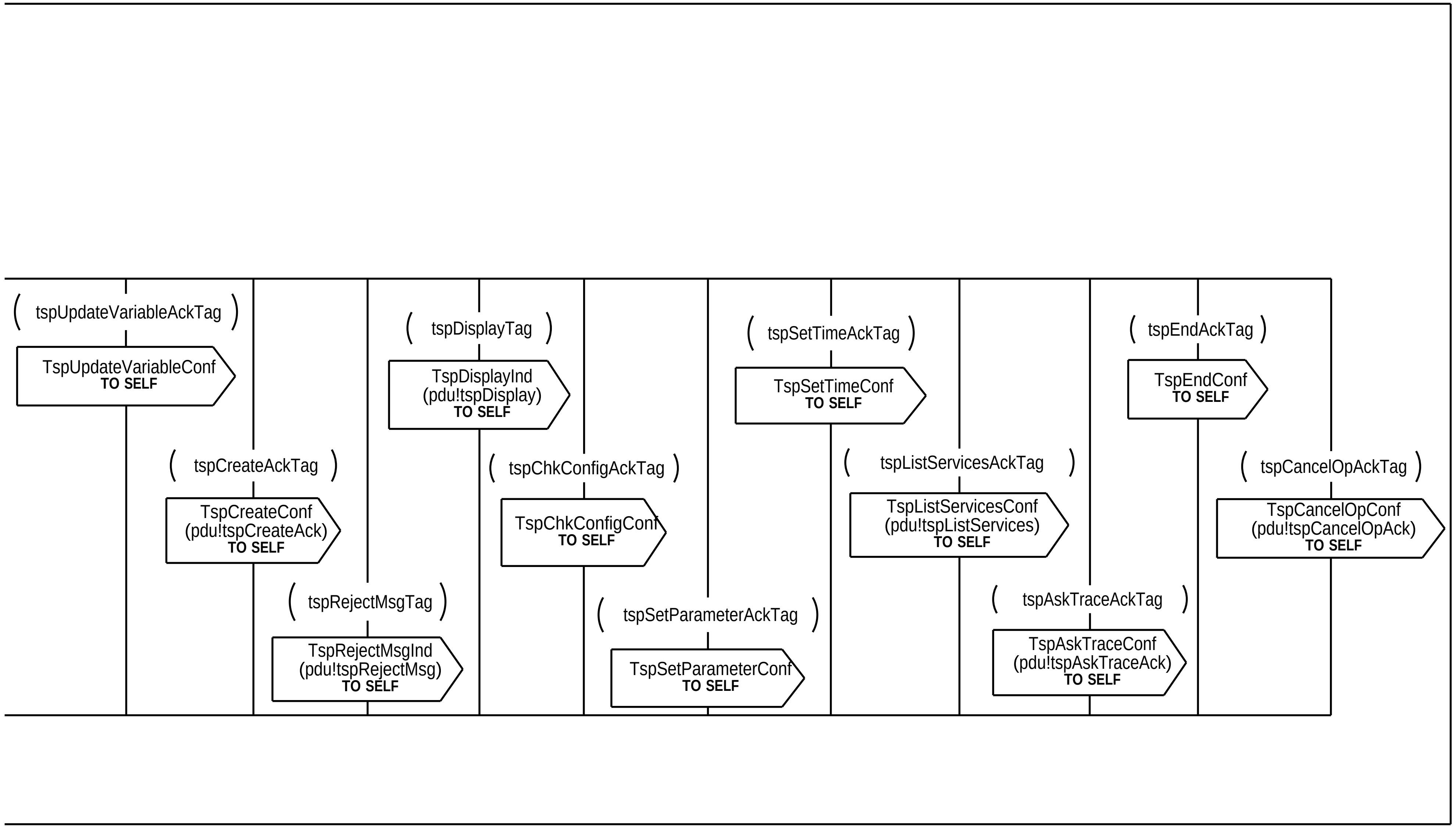


```
DCL pdu          TspPdu;
DCL initPdu      TspInitPdu;
DCL SETParamPdu  TspSetParameterPdu;
DCL SETTimePdu   TspSetTimePdu;
DCL createPdu    TspCreatePdu;
DCL infoPdu      TspInfoPdu;
DCL verdictPdu   TspVerdictPdu;
DCL variablePdu  TspUpdateVariablePdu;
DCL askTracePdu  TspAskTracePdu;
DCL cancelOpPdu  TspCancelOpPdu;
DCL displayPdu   TspDisplayPdu;
```

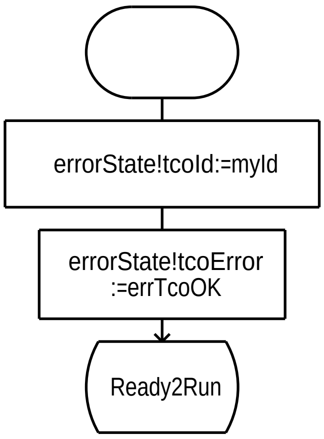


Ready
TspCreateReq (createPdu)
TspInfoReq (infoPdu)
TspVerdictReq (verdictPdu)
TspUpdateVariableReq (variablePdu)
TspAskTraceReq (asktracePdu)
TspEndReq
pdu!tag:= tspCreateTag
pdu!tag:= tspInfoTag
pdu!tag:= tspVerdictTag
pdu!tag:= tspUpdateVariableTag
pdu!tag:= tspAskTraceTag
pdu!tag:= tspEndTag
pdu!tspCreate :=createPdu
pdu!tspInfo :=infoPdu
pdu!tspVerdict :=verdictPdu
pdu!tspUpdateVariable :=variablePdu
pdu!tspAskTrace :=askTracePdu
sendTr
sendTr
sendTr
sendTr
sendTr
sendTr
Ready
TspCancelOpReq (cancelOpPdu)
TspDisplayReq (displayPdu)
TspUpdateVariableResp
pdu!tag:= tspCancelOpTag
pdu!tag:= tspDisplayTag
pdu!tag:= tspUpdateVariableAckTag
pdu!tspCancelOp :=cancelOpPdu
pdu!tspDisplay :=displayPdu
sendTr
sendTr
sendTr


service FeStTspDecoder
Ready
Ready
killDecoder
trPdu(pdu)
pdu!tag
DCL pdu TspPdu;
(tspErrorTag)
TspErrorInd (pdu!tspError) TO SELF
(tspInitAckTag)
TspInitConf TO SELF
(tspInitCompleteTag)
TspInitCompleteInd TO SELF
(tspInfoTag)
TspInfoInd (pdu!tspInfo) TO SELF
(tspVerdictTag)
TspVerdictInd (pdu!tspVerdict) TO SELF
(tspUpdateVariableTag)
TspUpdateVariableInd (pdu!tspUpdateVariable) TO SELF
(tspCreateTag)
TspCreateInd (pdu!tspCreate) TO SELF
-

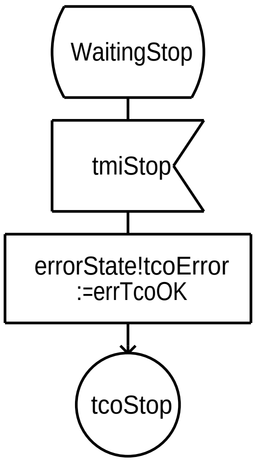

tspUpdateVariableAckTag
TspUpdateVariableConf TO SELF
tspCreateAckTag
TspCreateConf (pdu!tspCreateAck) TO SELF
tspRejectMsgTag
TspRejectMsgInd (pdu!tspRejectMsg) TO SELF
tspDisplayTag
TspDisplayInd (pdu!tspDisplay) TO SELF
tspChkConfigAckTag
TspChkConfigConf TO SELF
tspSetParameterAckTag
TspSetParameterConf TO SELF
tspSetTimeAckTag
TspSetTimeConf TO SELF
tspListServicesAckTag
TspListServicesConf (pdu!tspListServices) TO SELF
tspAskTraceAckTag
TspAskTraceConf (pdu!tspAskTraceAck) TO SELF
tspEndAckTag
TspEndConf TO SELF
tspCancelOpAckTag
TspCancelOpConf (pdu!tspCancelOpAck) TO SELF

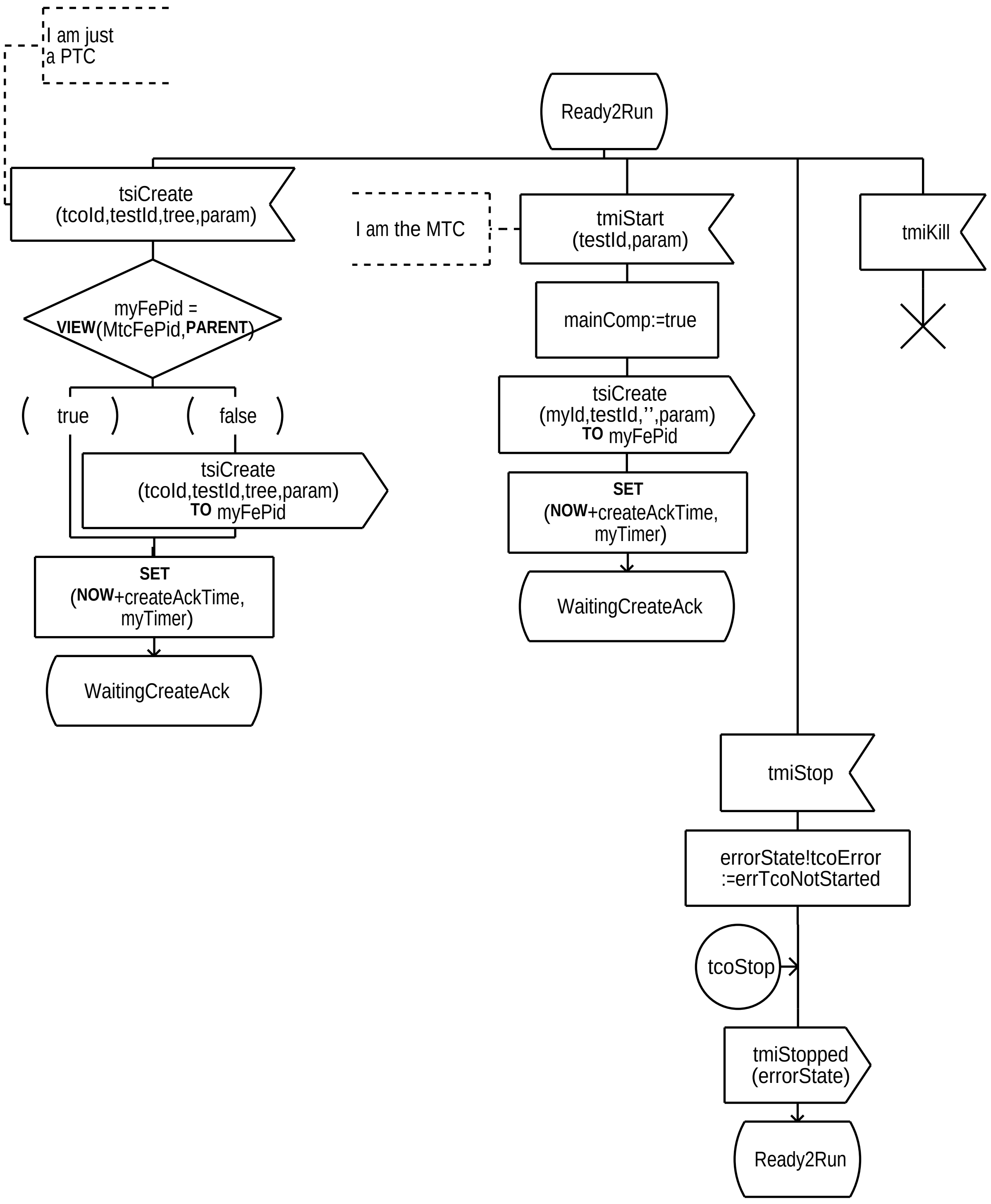
process TcoStub(0,nTCo)

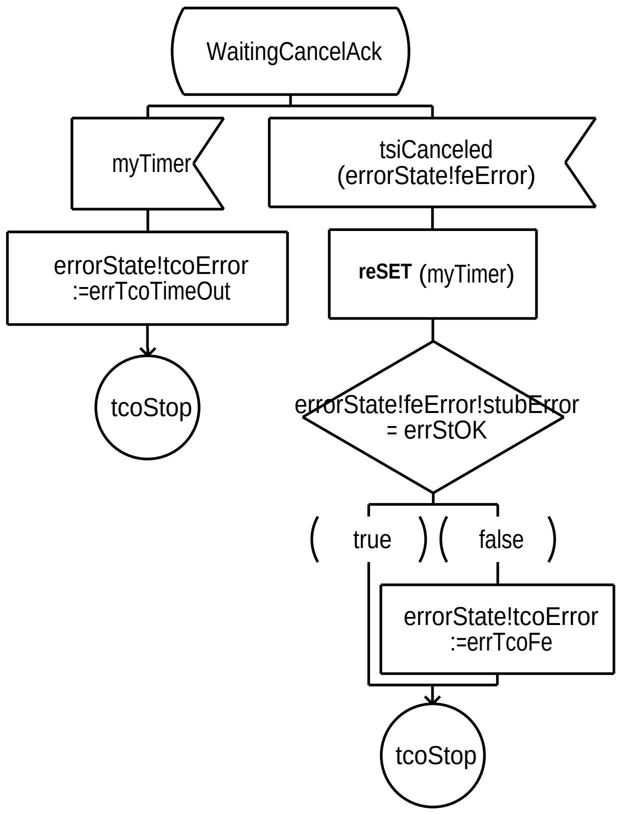


```
FPAR myId TcoId, myFePid PId;

DCL errorState TcoError;
TIMER myTimer;
DCL tree Tree;
DCL param ParamList;
VIEWED RunningTcos RoutingList;
VIEWED Mtc MtcId;
VIEWED MtcFePid PId;
DCL mtcPid PId;
DCL tcoId TcoId;
DCL cpId CpId;
DCL msg Msg;
DCL verdict VerdictValue;
DCL mainComp Boolean;
DCL testId TestCaseId;
```

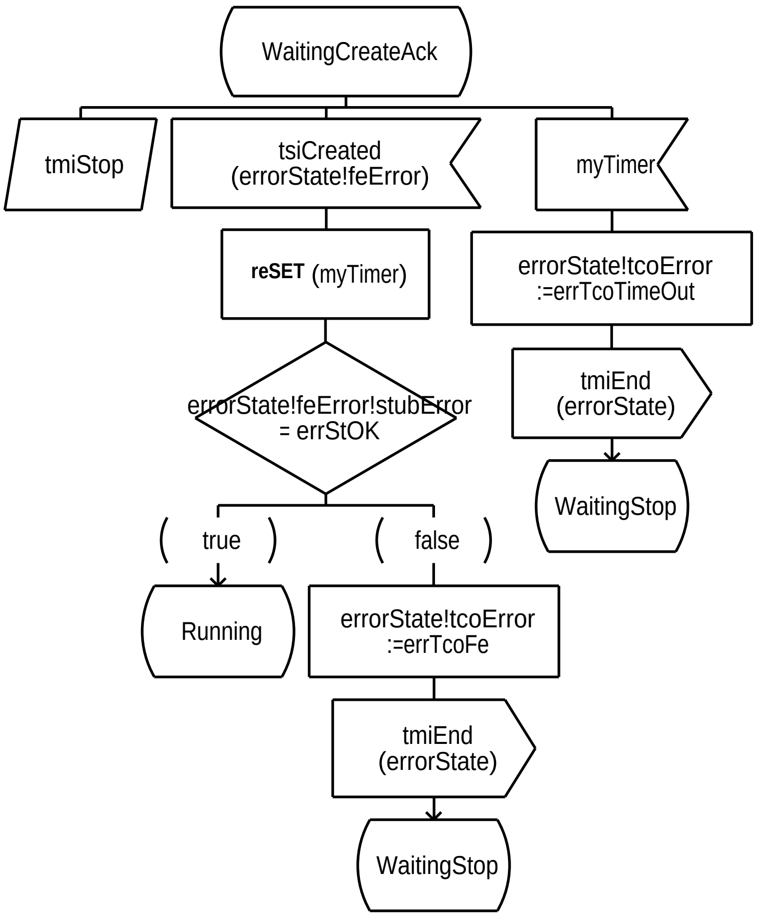
```
SIGNALSET tsiFinalVerdict,tsiUpdateVerdict;
```

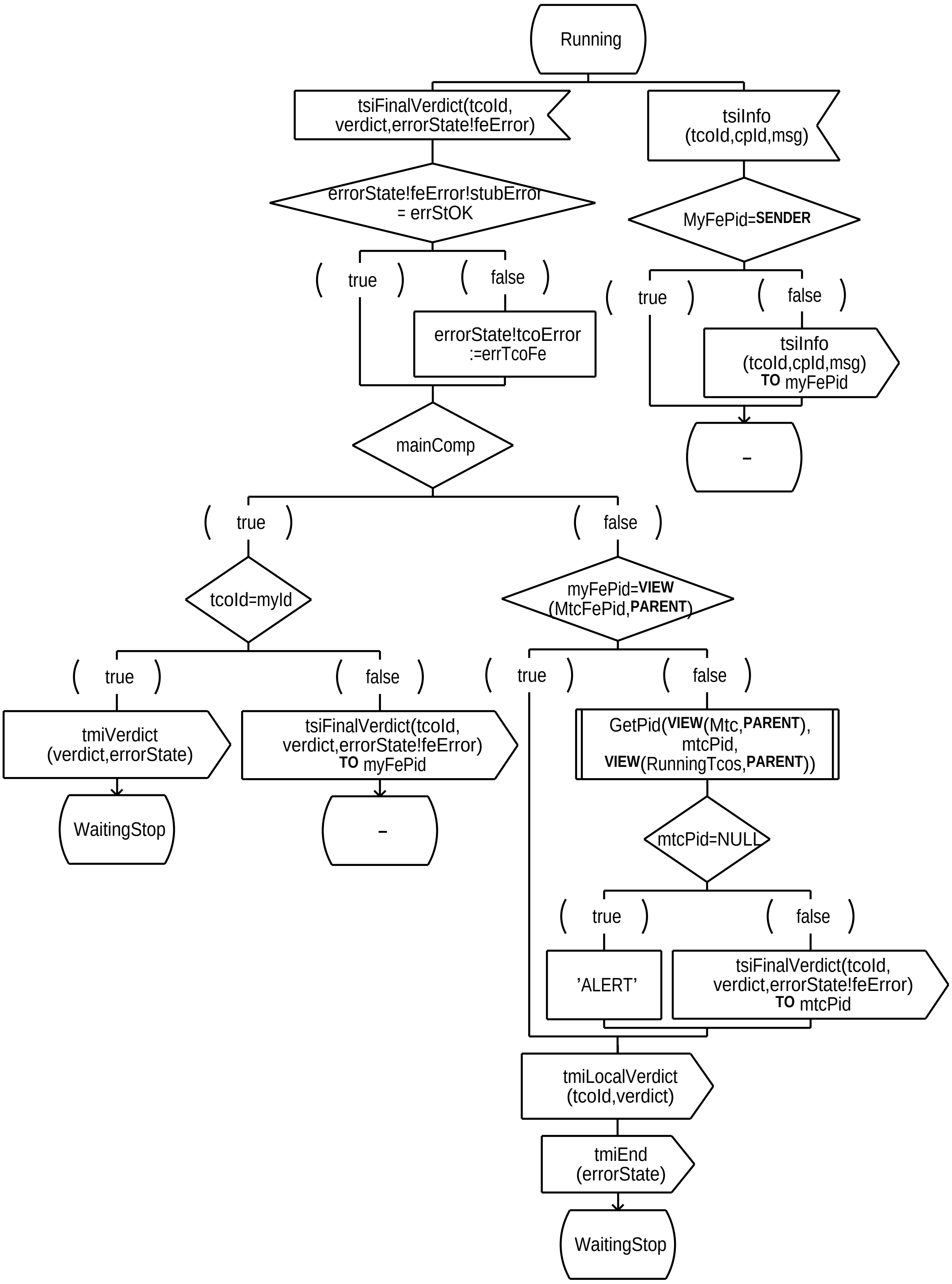



I am just
a PTC
Ready2Run
tsiCreate
(tcoId,testId,tree,param)
myFePid =
VIEW(MtcFePid,PARENT)
true
false
tsiCreate
(tcoId,testId,tree,param)
TO myFePid
SET
(NOW+createAckTime,
myTimer)
WaitingCreateAck
I am the MTC
tmiStart
(testId,param)
mainComp:=true
tsiCreate
(myId,testId,'',param)
TO myFePid
SET
(NOW+createAckTime,
myTimer)
WaitingCreateAck
tmiKill
tmiStop
errorState!tcoError
:=errTcoNotStarted
tcoStop
tmiStopped
(errorState)
Ready2Run

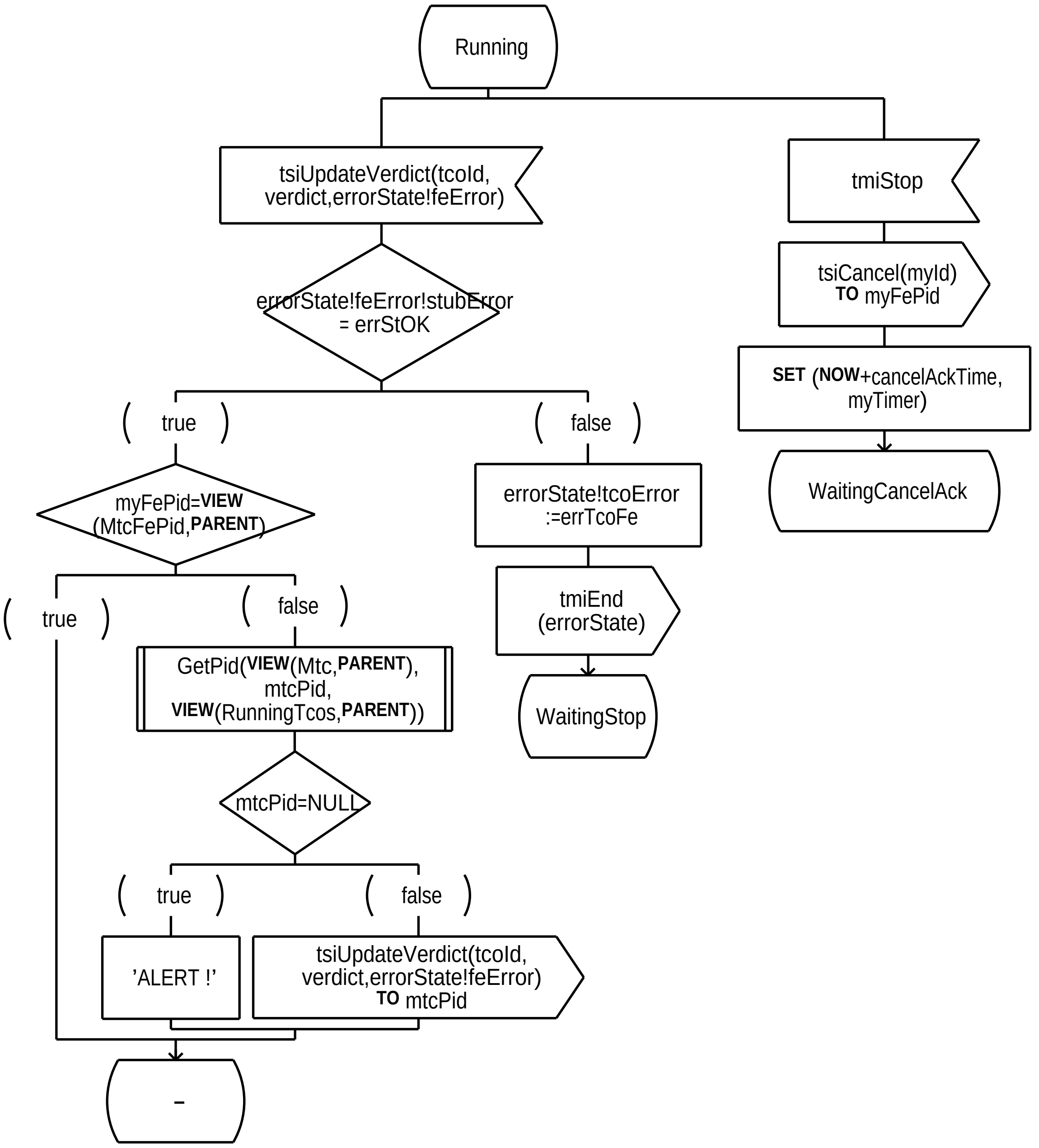


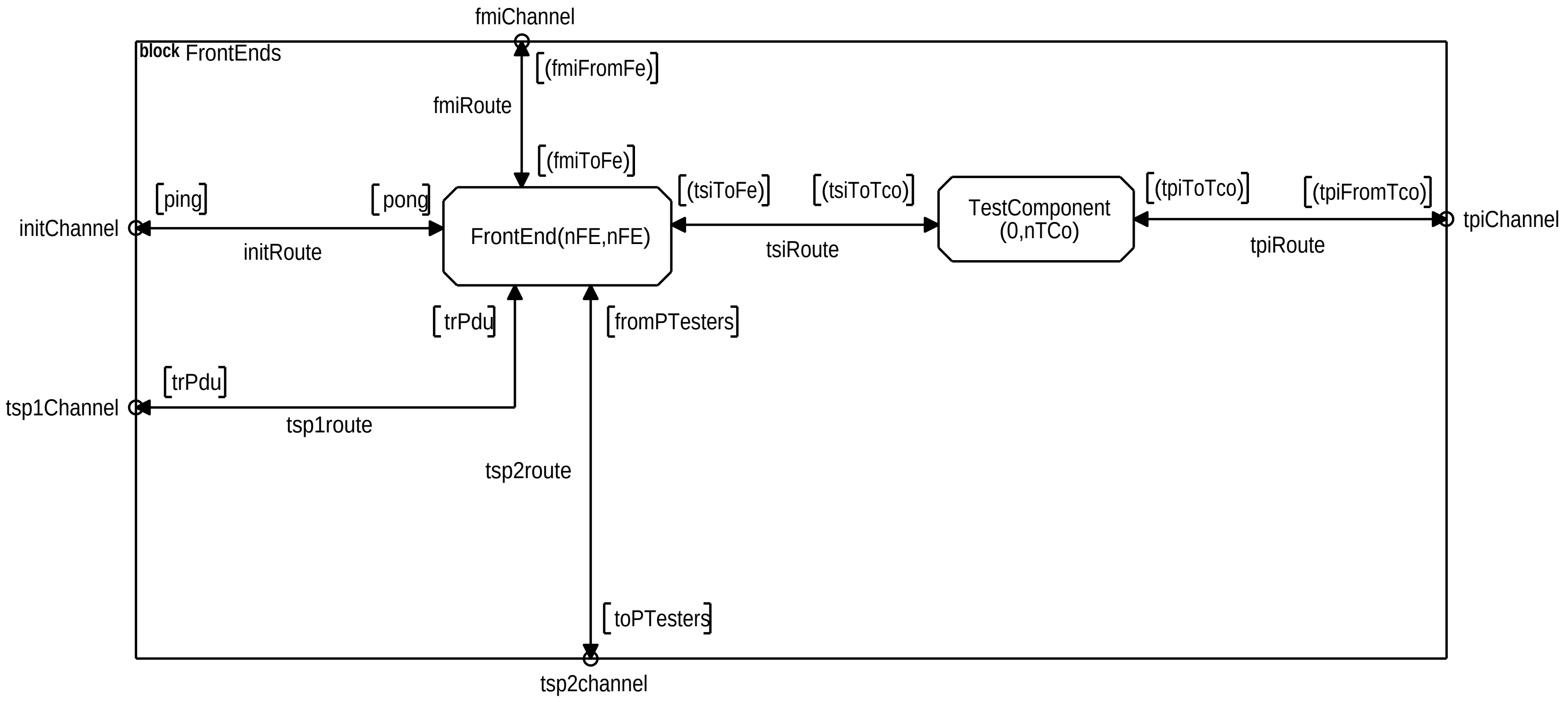
WaitingCancelAck
myTimer
tsiCanceled (errorState!feError)
errorState!tcoError :=errTcoTimeOut
reSET (myTimer)
tcoStop
errorState!feError!stubError = errStOK
true
false
errorState!tcoError :=errTcoFe
tcoStop

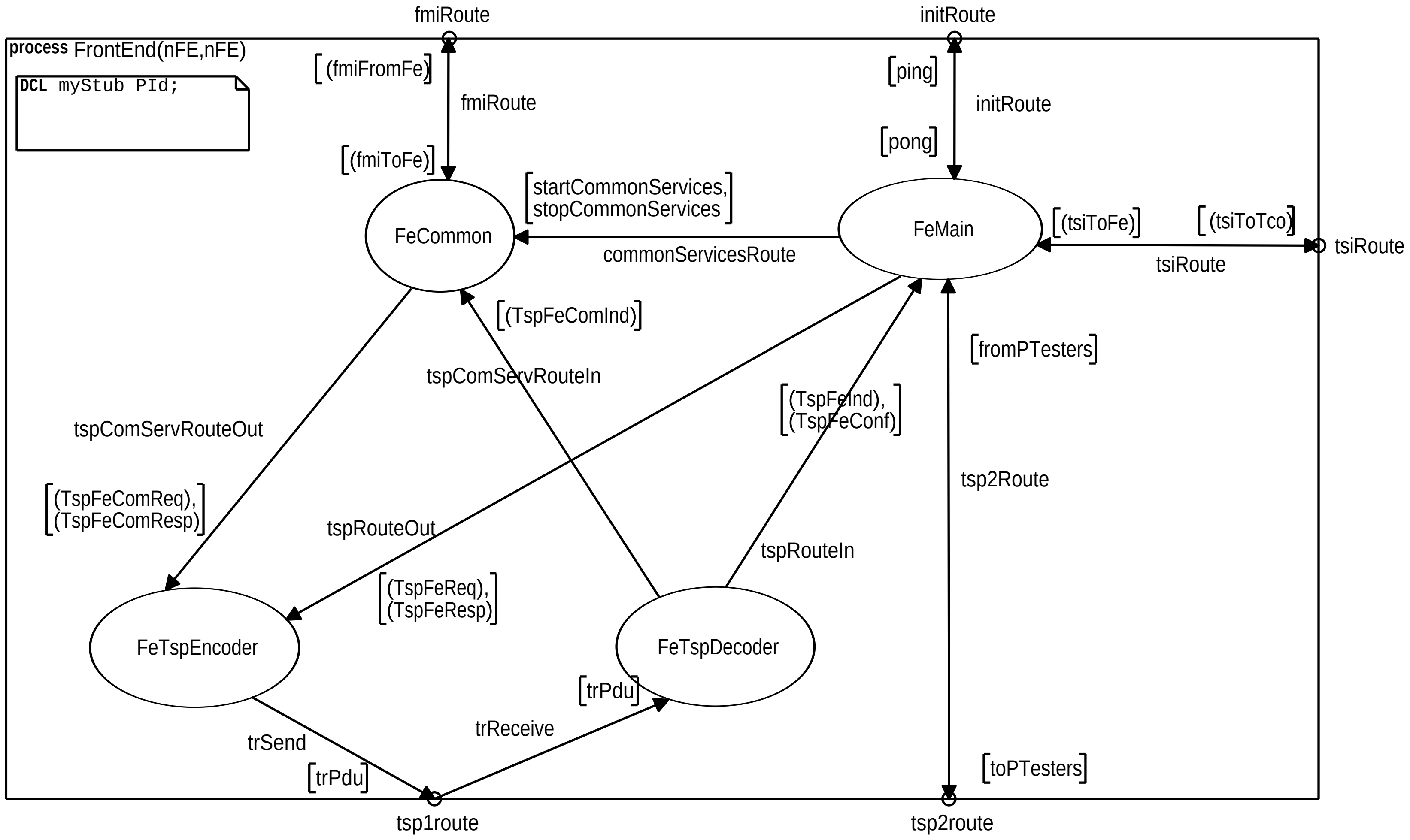

WaitingCreateAck
tmiStop
tsiCreated
(errorState!feError)
myTimer
reSET (myTimer)
errorState!tcoError
:=errTcoTimeOut
errorState!feError!stubError
= errStOK
tmiEnd
(errorState)
WaitingStop
true
false
Running
errorState!tcoError
:=errTcoFe
tmiEnd
(errorState)
WaitingStop



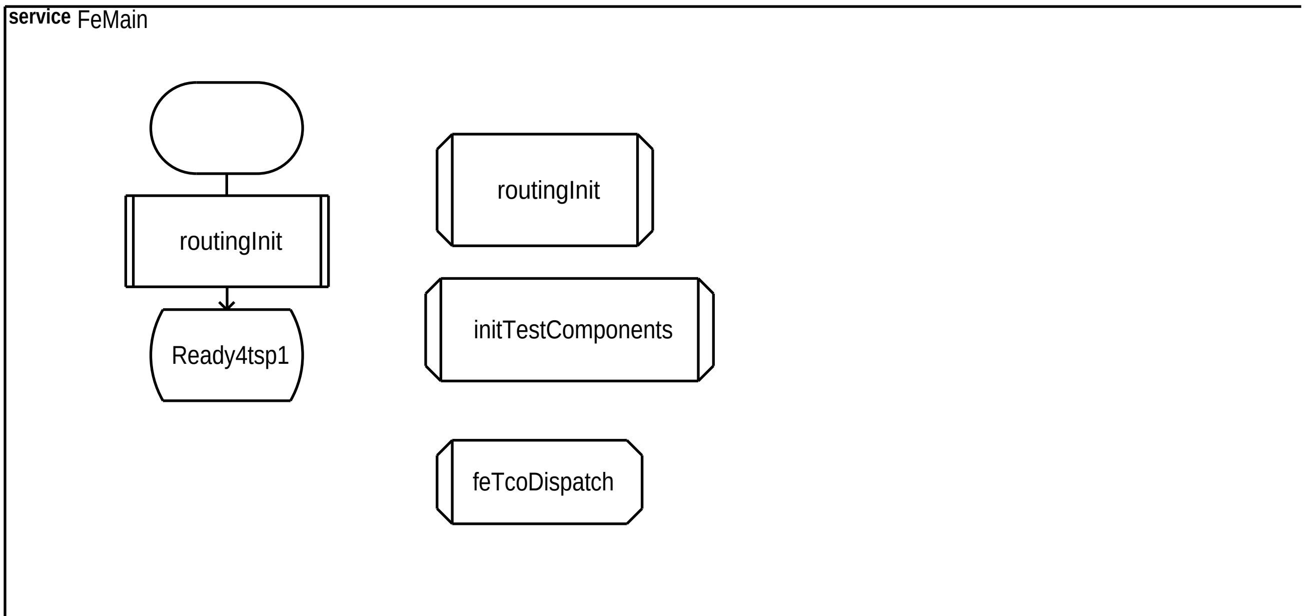
Running
tsiFinalVerdict(tcoId, verdict,errorState!feError)
tsiInfo (tcoId,cpId,msg)
errorState!feError!stubError = errStOK
true
false
errorState!tcoError :=errTcoFe
MyFePid=SENDER
true
false
tsiInfo (tcoId,cpId,msg) TO myFePid
-
mainComp
true
false
tcoId=myId
myFePid=VIEW (MtcFePid,PARENT)
true
false
tmiVerdict (verdict,errorState)
WaitingStop
tsiFinalVerdict(tcoId, verdict,errorState!feError) TO myFePid
-
true
false
GetPid(VIEW(Mtc,PARENT), mtcPid, VIEW(RunningTcos,PARENT))
mtcPid=NULL
true
false
'ALERT'
tsiFinalVerdict(tcoId, verdict,errorState!feError) TO mtcPid
tmiLocalVerdict (tcoId,verdict)
tmiEnd (errorState)
WaitingStop


Running
tsiUpdateVerdict(tcoId, verdict,errorState!feError)
tmiStop
tsiCancel(myId) TO myFePid
SET (NOW+cancelAckTime, myTimer)
WaitingCancelAck
errorState!feError!stubError = errStOK
true
false
myFePid=VIEW (MtcFePid,PARENT)
errorState!tcoError :=errTcoFe
tmiEnd (errorState)
WaitingStop
true
false
GetPid(VIEW(Mtc,PARENT), mtcPid, VIEW(RunningTcos,PARENT))
mtcPid=NULL
true
false
'ALERT !'
tsiUpdateVerdict(tcoId, verdict,errorState!feError) TO mtcPid
-

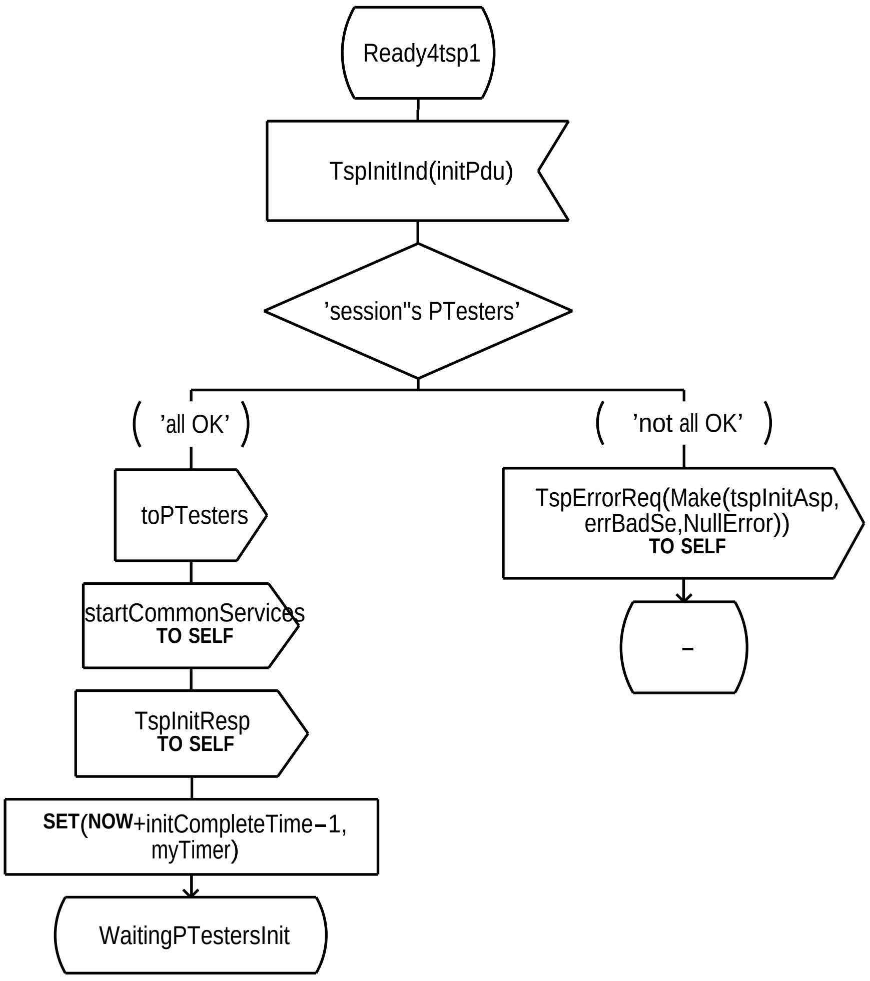

fmiChannel
block FrontEnds
[(fmiFromFe)]
fmiRoute
[(fmiToFe)]
[ping]
[pong]
initChannel
initRoute
FrontEnd(nFE,nFE)
[(tsiToFe)]
[(tsiToTco)]
tsiRoute
TestComponent
(0,nTCo)
[(tpiToTco)]
[(tpiFromTco)]
tpiChannel
tpiRoute
[trPdu]
[fromPTesters]
[trPdu]
tsp1Channel
tsp1route
tsp2route
[toPTesters]
tsp2channel

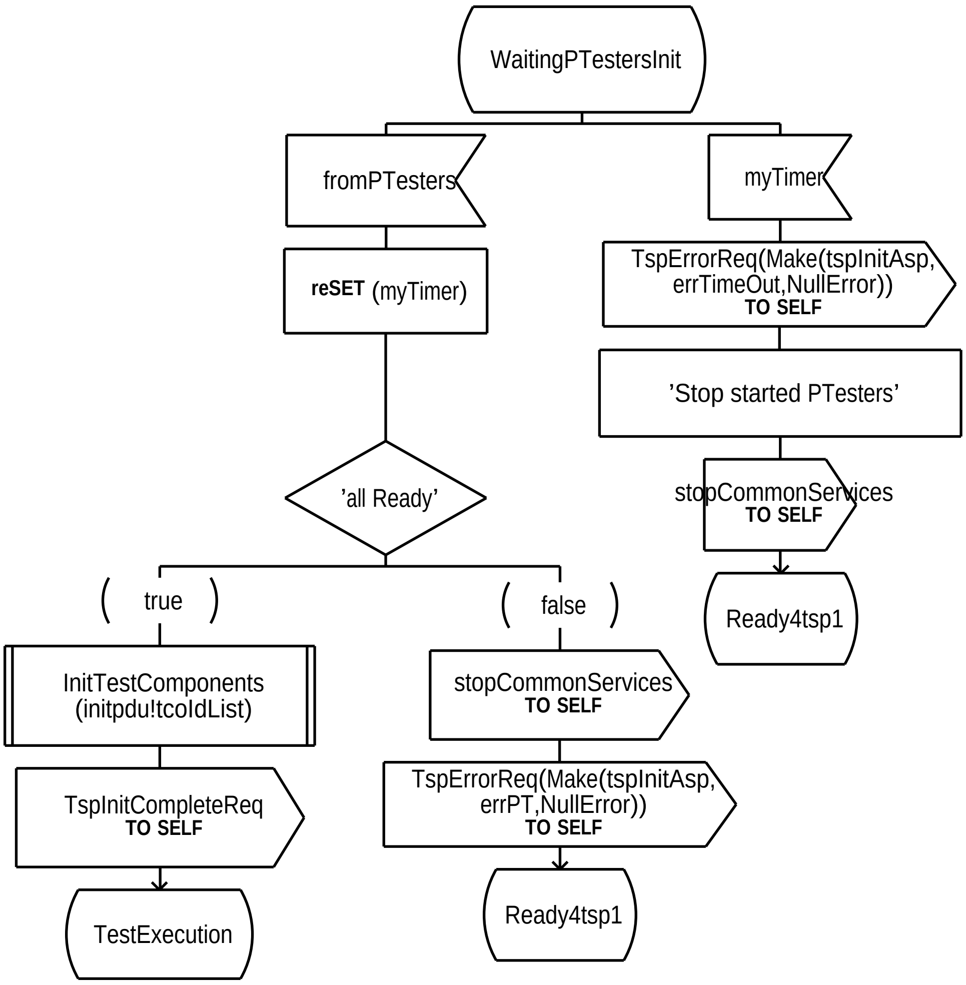


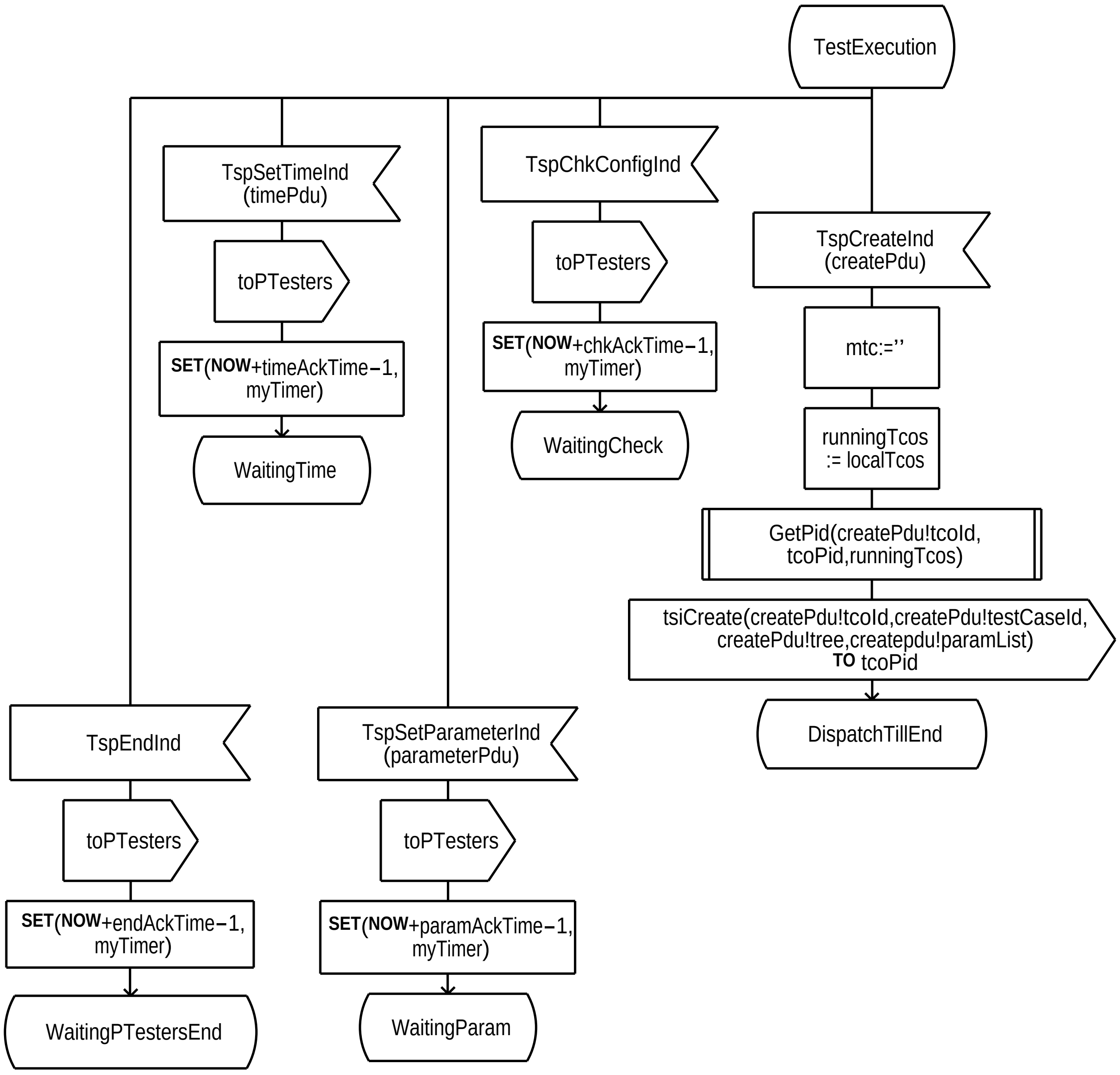
fmiRoute
initRoute
process FrontEnd(nFE,nFE)
DCL myStub PId;
[(fmiFromFe)]
fmiRoute
[(fmiToFe)]
[ping]
initRoute
[pong]
[startCommonServices, stopCommonServices]
FeCommon
commonServicesRoute
FeMain
[(tsiToFe)]
[(tsiToTco)]
tsiRoute
tsiRoute
[(TspFeComInd)]
tspComServRouteIn
[fromPTesters]
[(TspFeInd), (TspFeConf)]
tspComServRouteOut
tsp2Route
[(TspFeComReq), (TspFeComResp)]
tspRouteOut
tspRouteIn
[(TspFeReq), (TspFeResp)]
FeTspEncoder
FeTspDecoder
[trPdu]
trReceive
trSend
[trPdu]
[toPTesters]
tsp1route
tsp2route

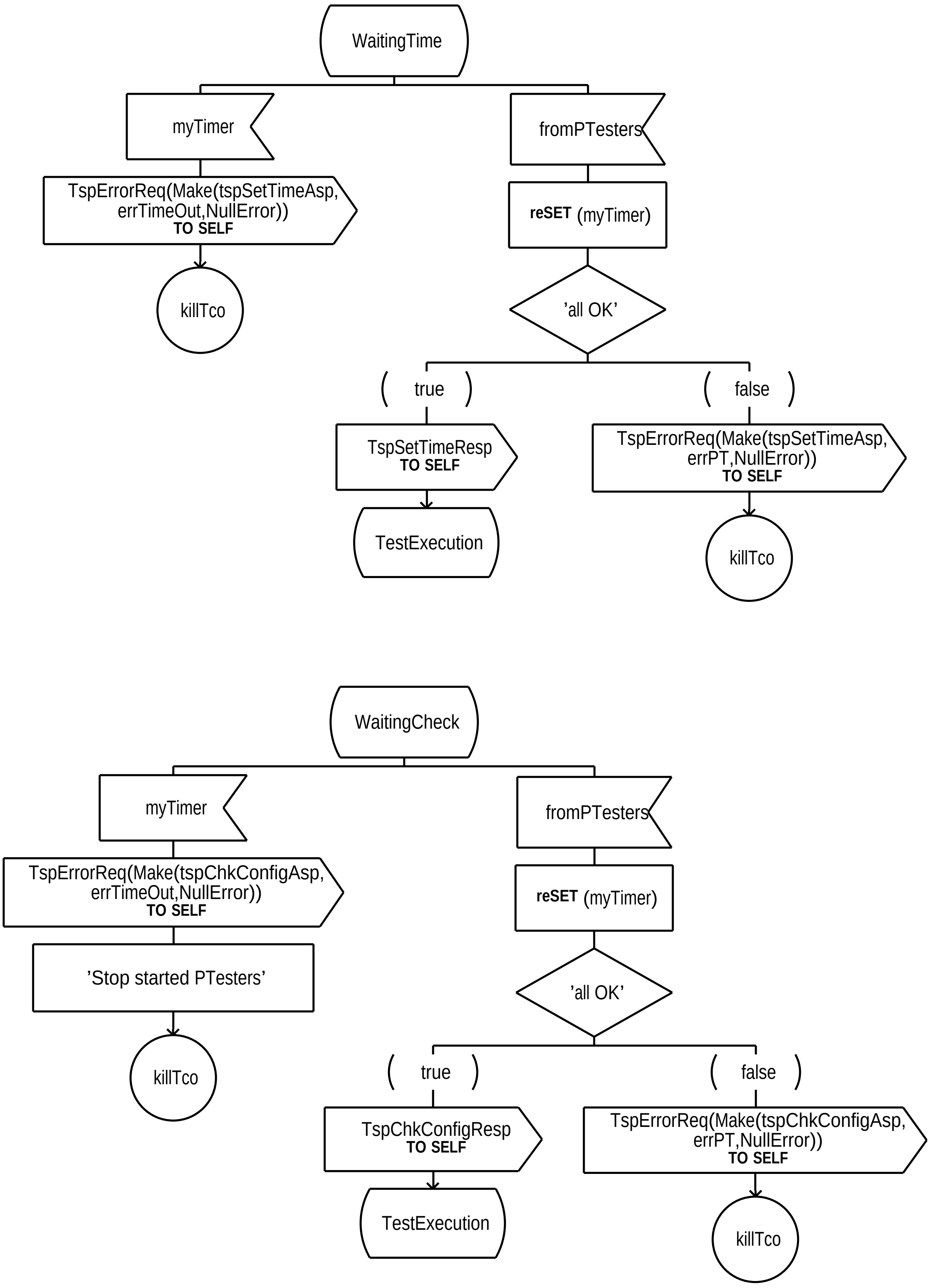


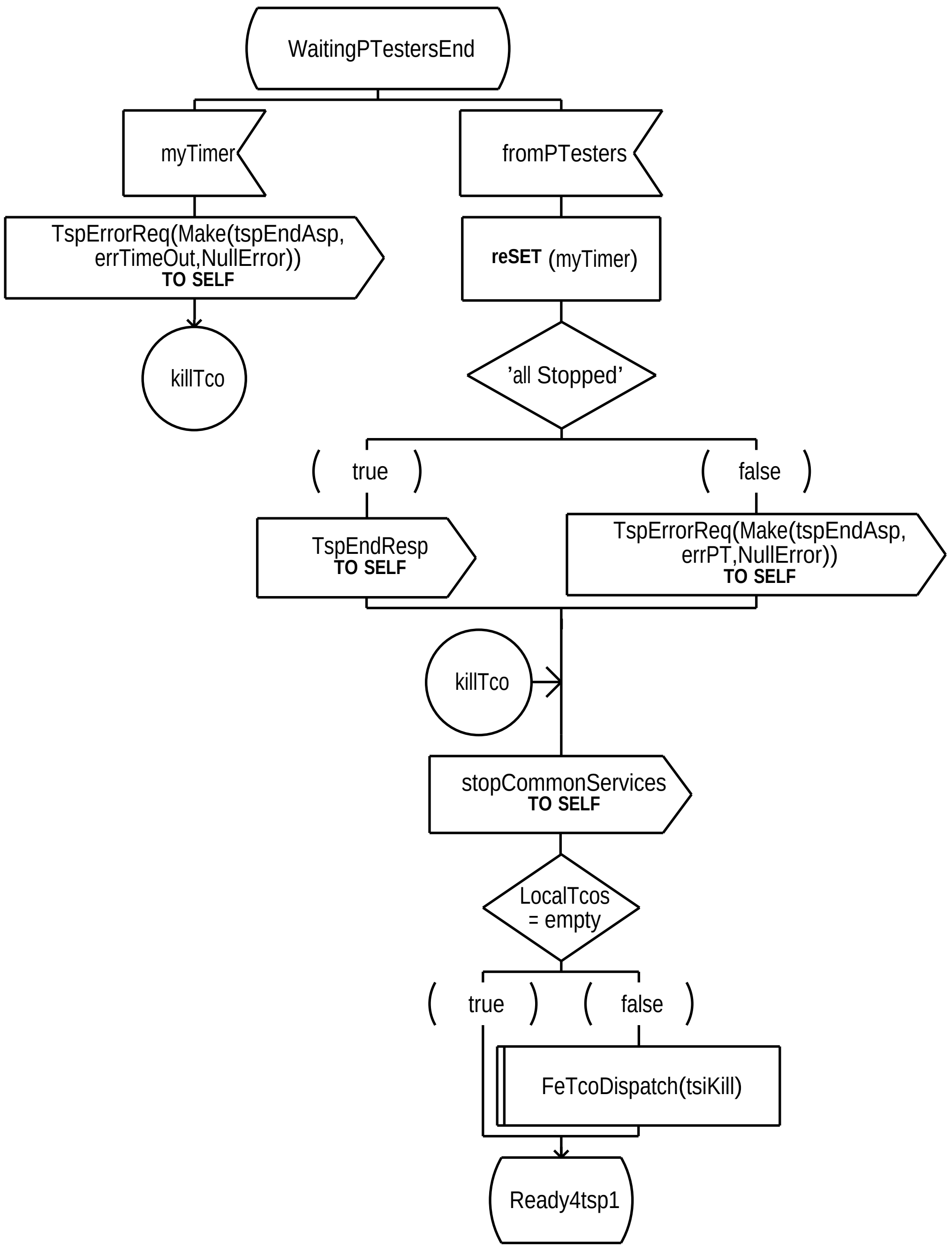
```
DCL myId FeId;
DCL initPdu TspInitPdu;
DCL timePdu TspSetTimePdu;
DCL parameterPdu TspSetParameterPdu;
DCL createPdu TspCreatePdu;
DCL verdictPdu TspVerdictPdu;
DCL cancelOpPdu TspCancelOpPdu;
DCL infoPdu TspInfoPdu;

DCL tcoPid PId;
DCL mtc Charstring;
TIMER myTimer;
DCL err FeError;
DCL testCaseId TestCaseId;
DCL tcoId CharString;
DCL feError FeError;
DCL tree Tree;
DCL param ParamList;
DCL verdict VerdictValue;
DCL tcoCounter TcoIndex;
DCL localTcos, runningTcos RoutingList;
DCL routingElem RoutingElement;
DCL cpId CpId;
DCL coordMsg Msg;
```

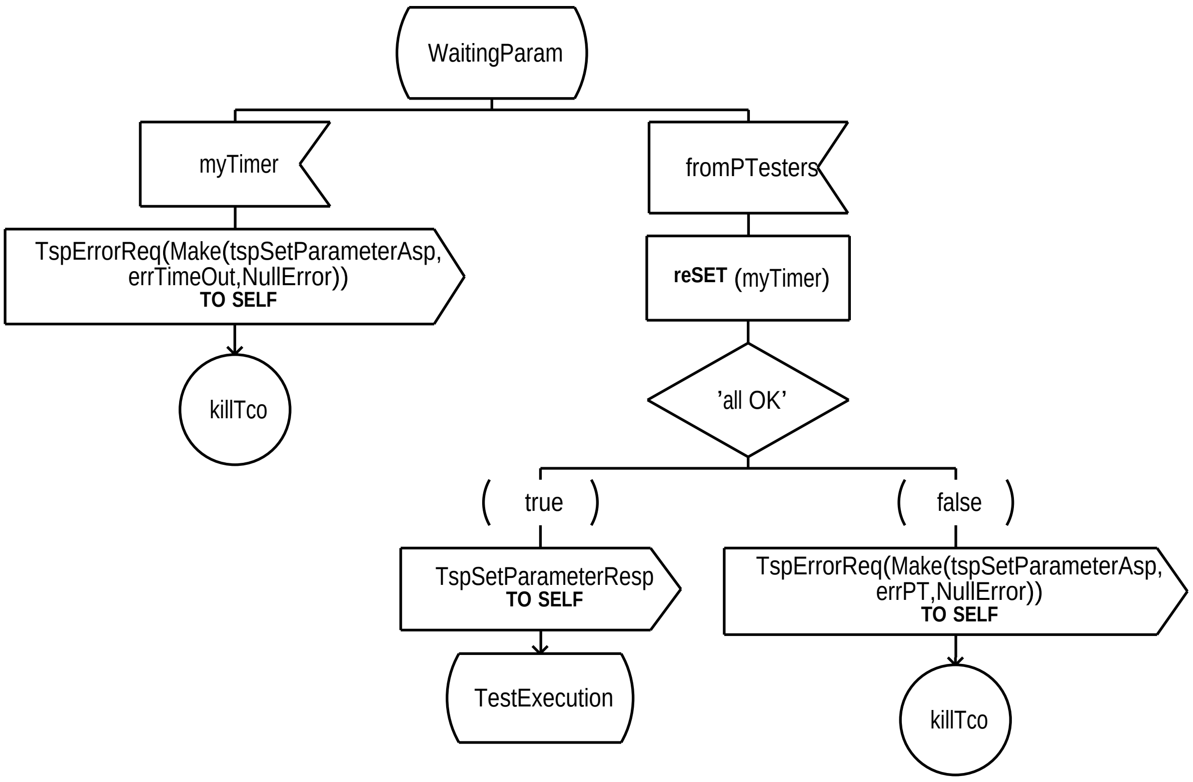

Ready4tsp1
TspInitInd(initPdu)
'session''s PTesters'
'all OK'
'not all OK'
toPTesters
startCommonServices TO SELF
TspInitResp TO SELF
SET(NOW+initCompleteTime−1, myTimer)
WaitingPTestersInit
TspErrorReq(Make(tspInitAsp, errBadSe,NullError)) TO SELF
–

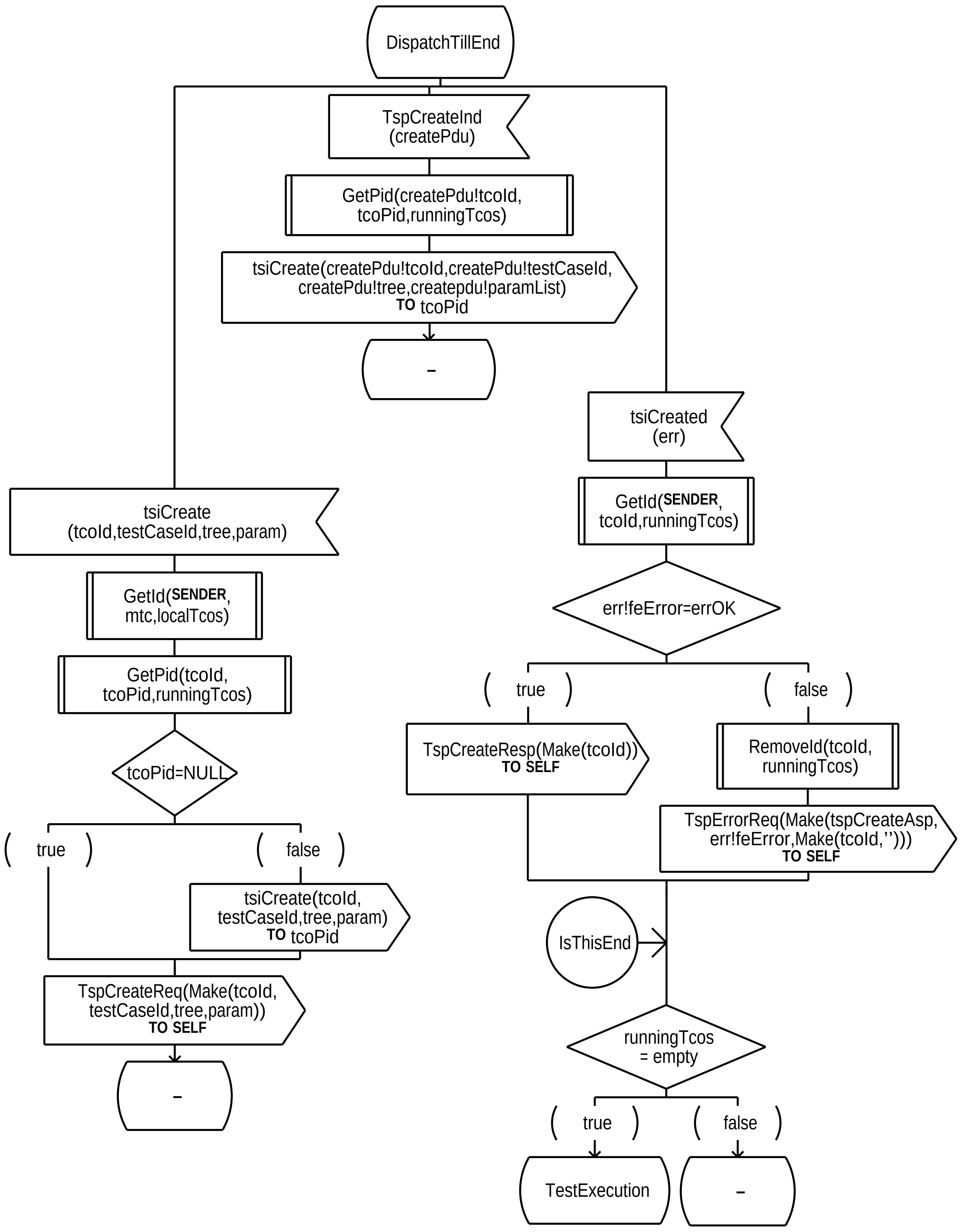

WaitingPTestersInit
fromPTesters
myTimer
reSET (myTimer)
TspErrorReq(Make(tspInitAsp, errTimeOut,NullError)) TO SELF
'Stop started PTesters'
stopCommonServices TO SELF
Ready4tsp1
'all Ready'
true
false
InitTestComponents (initpdu!tcoIdList)
TspInitCompleteReq TO SELF
TestExecution
stopCommonServices TO SELF
TspErrorReq(Make(tspInitAsp, errPT,NullError)) TO SELF
Ready4tsp1

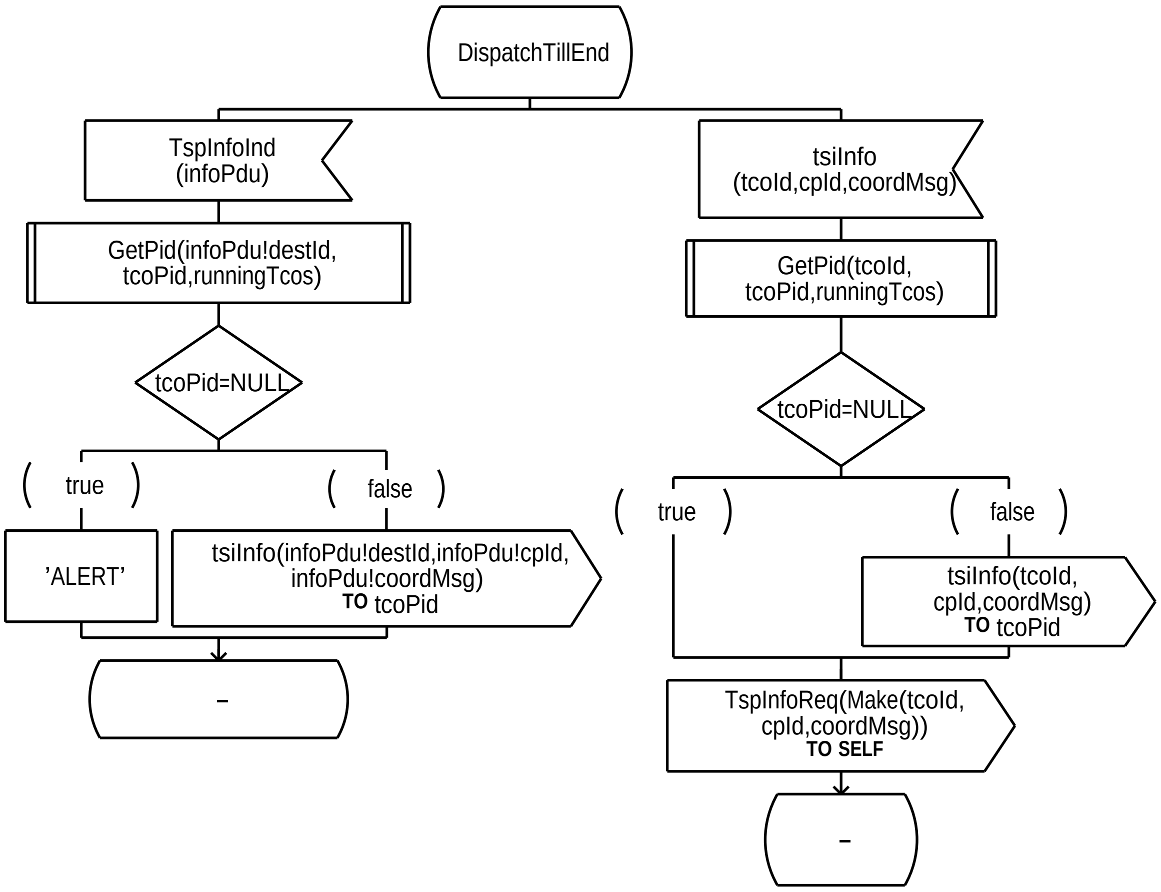

TestExecution
TspSetTimeInd (timePdu)
toPTesters
SET(NOW+timeAckTime−1, myTimer)
WaitingTime
TspChkConfigInd
toPTesters
SET(NOW+chkAckTime−1, myTimer)
WaitingCheck
TspCreateInd (createPdu)
mtc:=''
runningTcos := localTcos
GetPid(createPdu!tcoId, tcoPid,runningTcos)
tsiCreate(createPdu!tcoId,createPdu!testCaseId, createPdu!tree,createpdu!paramList) TO tcoPid
DispatchTillEnd
TspEndInd
toPTesters
SET(NOW+endAckTime−1, myTimer)
WaitingPTestersEnd
TspSetParameterInd (parameterPdu)
toPTesters
SET(NOW+paramAckTime−1, myTimer)
WaitingParam

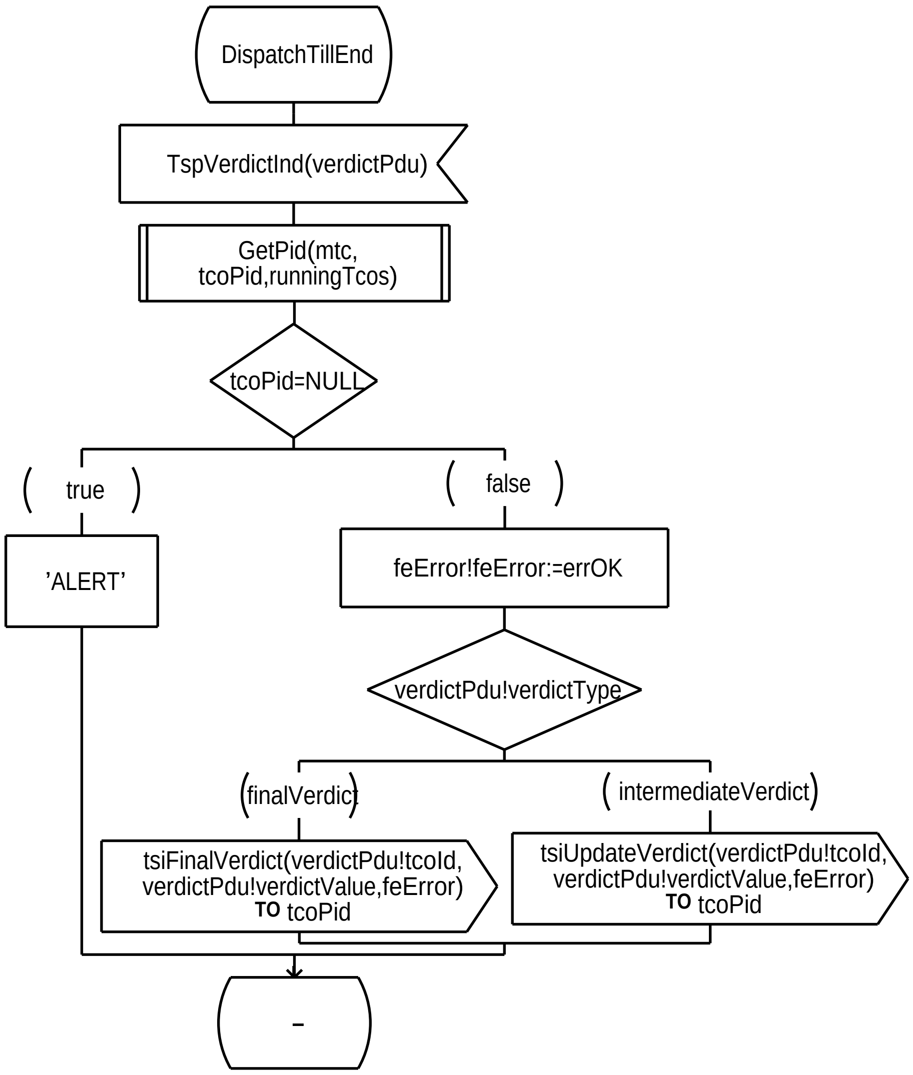

WaitingTime
myTimer
TspErrorReq(Make(tspSetTimeAsp,
errTimeOut,NullError))
TO SELF
killTco
fromPTesters
reSET (myTimer)
'all OK'
true
false
TspSetTimeResp
TO SELF
TestExecution
TspErrorReq(Make(tspSetTimeAsp,
errPT,NullError))
TO SELF
killTco
WaitingCheck
myTimer
TspErrorReq(Make(tspChkConfigAsp,
errTimeOut,NullError))
TO SELF
'Stop started PTesters'
killTco
fromPTesters
reSET (myTimer)
'all OK'
true
false
TspChkConfigResp
TO SELF
TestExecution
TspErrorReq(Make(tspChkConfigAsp,
errPT,NullError))
TO SELF
killTco

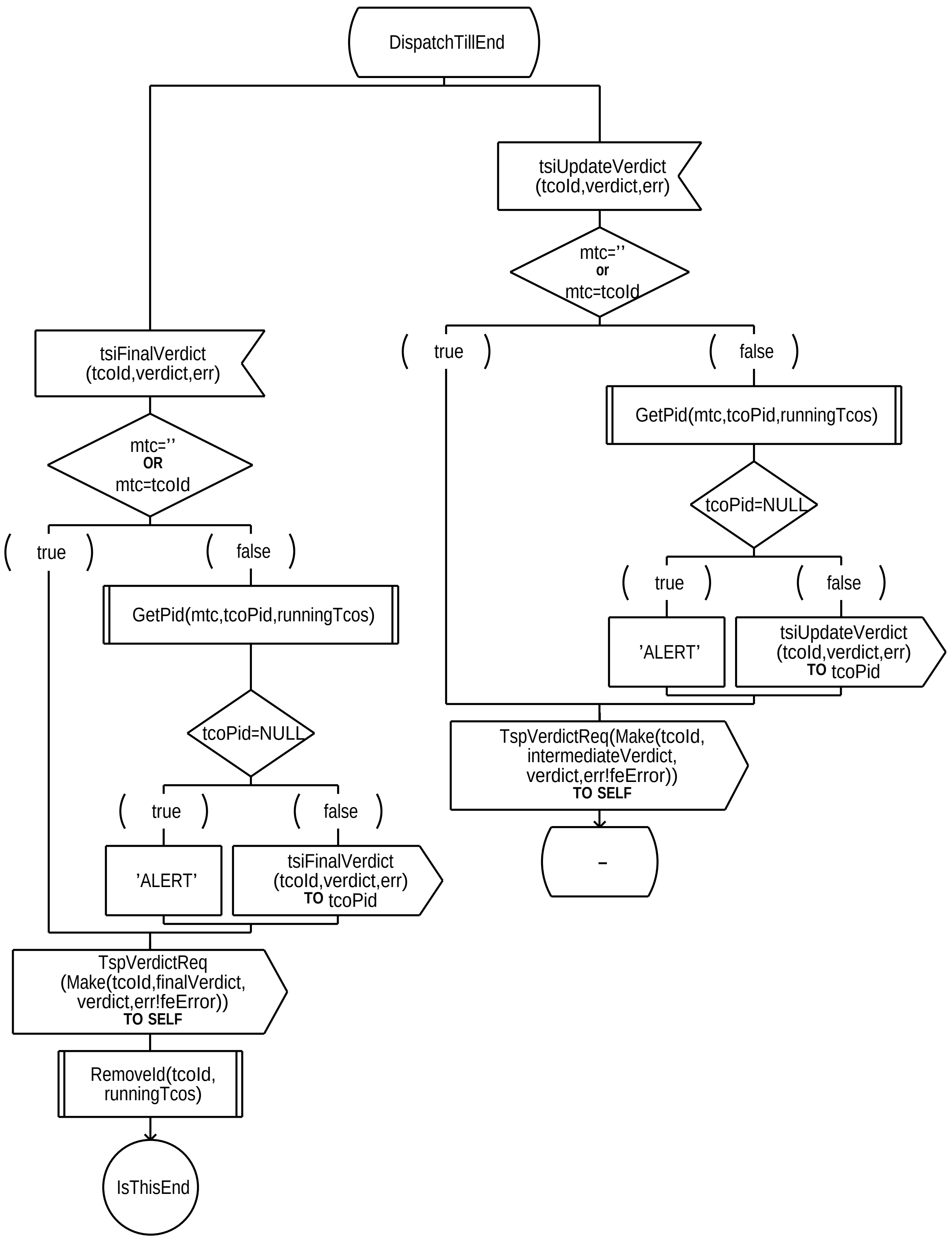

WaitingPTestersEnd
myTimer
fromPTesters
TspErrorReq(Make(tspEndAsp, errTimeOut,NullError)) TO SELF
reSET (myTimer)
killTco
'all Stopped'
true
false
TspEndResp TO SELF
TspErrorReq(Make(tspEndAsp, errPT,NullError)) TO SELF
killTco
stopCommonServices TO SELF
LocalTcos = empty
true
false
FeTcoDispatch(tsiKill)
Ready4tsp1

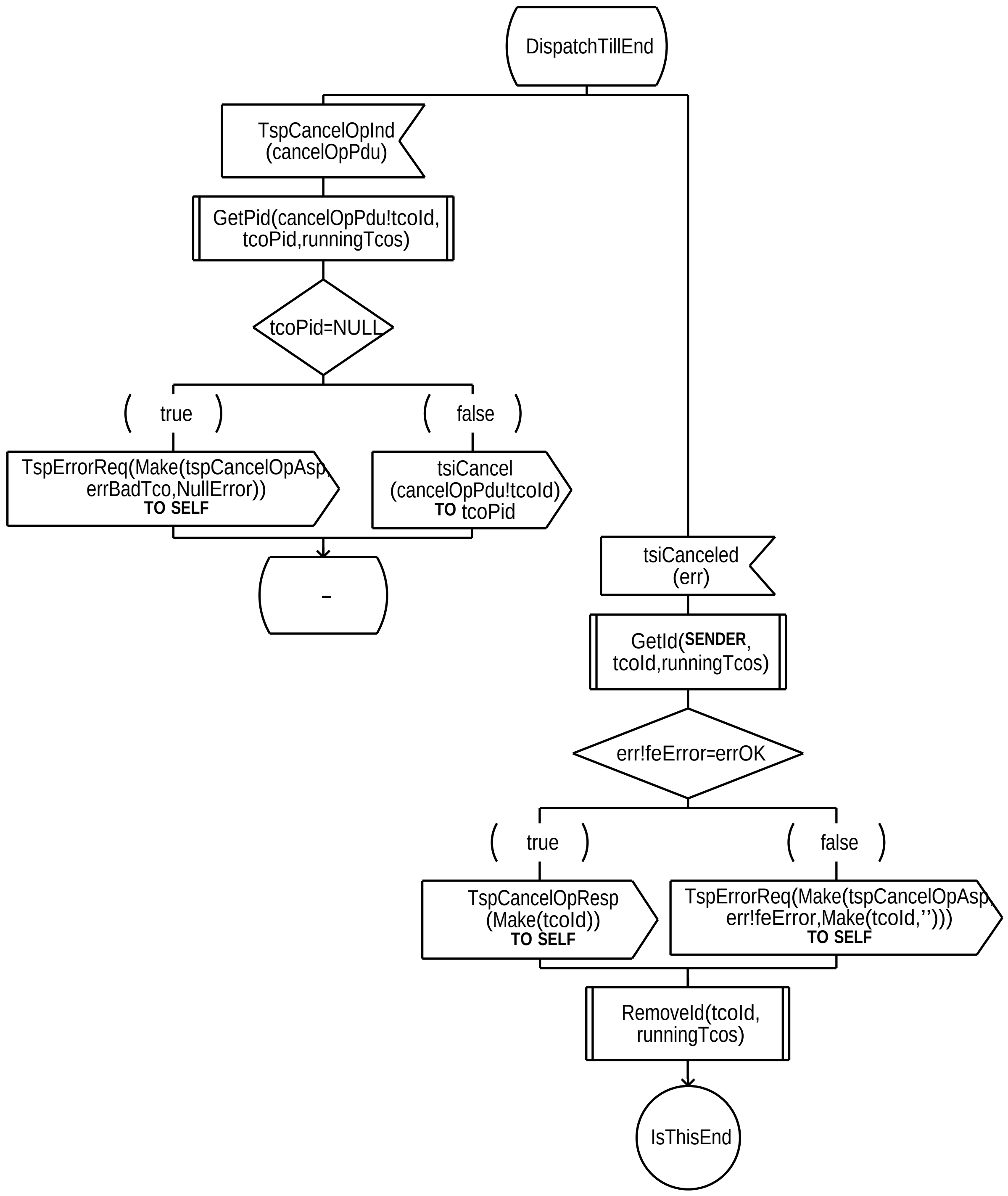

WaitingParam
myTimer
TspErrorReq(Make(tspSetParameterAsp, errTimeOut,NullError))
TO SELF
killTco
fromPTesters
reSET (myTimer)
'all OK'
true
false
TspSetParameterResp
TO SELF
TestExecution
TspErrorReq(Make(tspSetParameterAsp, errPT,NullError))
TO SELF
killTco

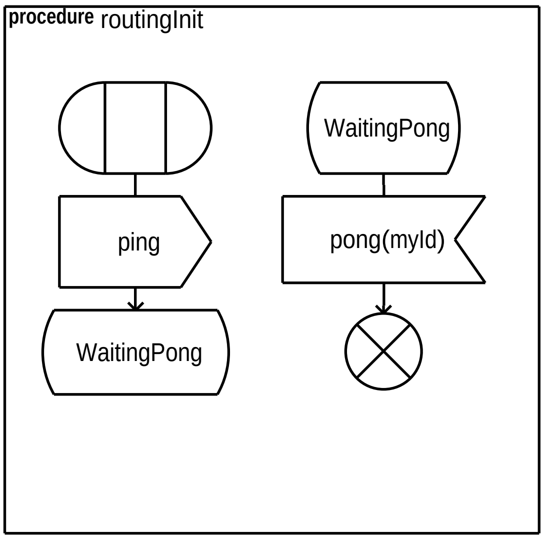

DispatchTillEnd
TspCreateInd
(createPdu)
GetPid(createPdu!tcoId,
tcoPid,runningTcos)
tsiCreate(createPdu!tcoId,createPdu!testCaseId,
createPdu!tree,createpdu!paramList)
TO tcoPid
-
tsiCreated
(err)
GetId(SENDER,
tcoId,runningTcos)
err!feError=errOK
true
false
TspCreateResp(Make(tcoId))
TO SELF
RemoveId(tcoId,
runningTcos)
TspErrorReq(Make(tspCreateAsp,
err!feError,Make(tcoId,’’)))
TO SELF
IsThisEnd
runningTcos
= empty
true
false
TestExecution
-
tsiCreate
(tcoId,testCaseId,tree,param)
GetId(SENDER,
mtc,localTcos)
GetPid(tcoId,
tcoPid,runningTcos)
tcoPid=NULL
true
false
tsiCreate(tcoId,
testCaseId,tree,param)
TO tcoPid
TspCreateReq(Make(tcoId,
testCaseId,tree,param))
TO SELF
-


DispatchTillEnd
TspInfoInd (infoPdu)
GetPid(infoPdu!destId, tcoPid,runningTcos)
tcoPid=NULL
true
false
'ALERT'
tsiInfo(infoPdu!destId,infoPdu!cpId, infoPdu!coordMsg) TO tcoPid
-
tsiInfo (tcoId,cpId,coordMsg)
GetPid(tcoId, tcoPid,runningTcos)
tcoPid=NULL
true
false
tsiInfo(tcoId, cpId,coordMsg) TO tcoPid
TspInfoReq(Make(tcoId, cpId,coordMsg)) TO SELF
-

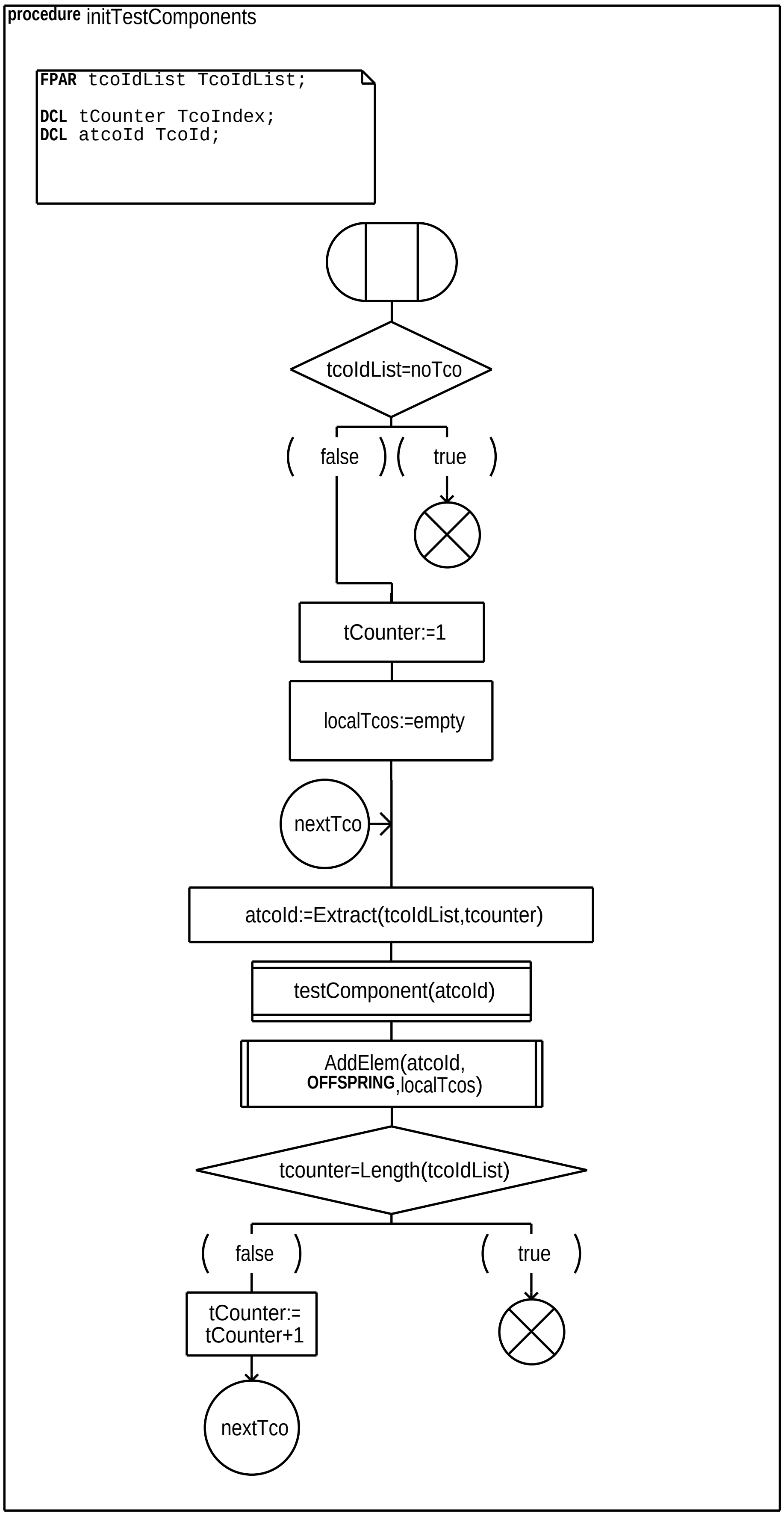

DispatchTillEnd
TspVerdictInd(verdictPdu)
GetPid(mtc, tcoPid,runningTcos)
tcoPid=NULL
true
false
'ALERT'
feError!feError:=errOK
verdictPdu!verdictType
finalVerdict
intermediateVerdict
tsiFinalVerdict(verdictPdu!tcoId, verdictPdu!verdictValue,feError) TO tcoPid
tsiUpdateVerdict(verdictPdu!tcoId, verdictPdu!verdictValue,feError) TO tcoPid
-


DispatchTillEnd
tsiUpdateVerdict (tcoId,verdict,err)
mtc='' or mtc=tcoId
true
false
GetPid(mtc,tcoPid,runningTcos)
tcoPid=NULL
true
false
'ALERT'
tsiUpdateVerdict (tcoId,verdict,err) TO tcoPid
TspVerdictReq(Make(tcoId, intermediateVerdict, verdict,err!feError)) TO SELF
–
tsiFinalVerdict (tcoId,verdict,err)
mtc='' OR mtc=tcoId
true
false
GetPid(mtc,tcoPid,runningTcos)
tcoPid=NULL
true
false
'ALERT'
tsiFinalVerdict (tcoId,verdict,err) TO tcoPid
TspVerdictReq (Make(tcoId,finalVerdict, verdict,err!feError)) TO SELF
RemoveId(tcoId, runningTcos)
IsThisEnd

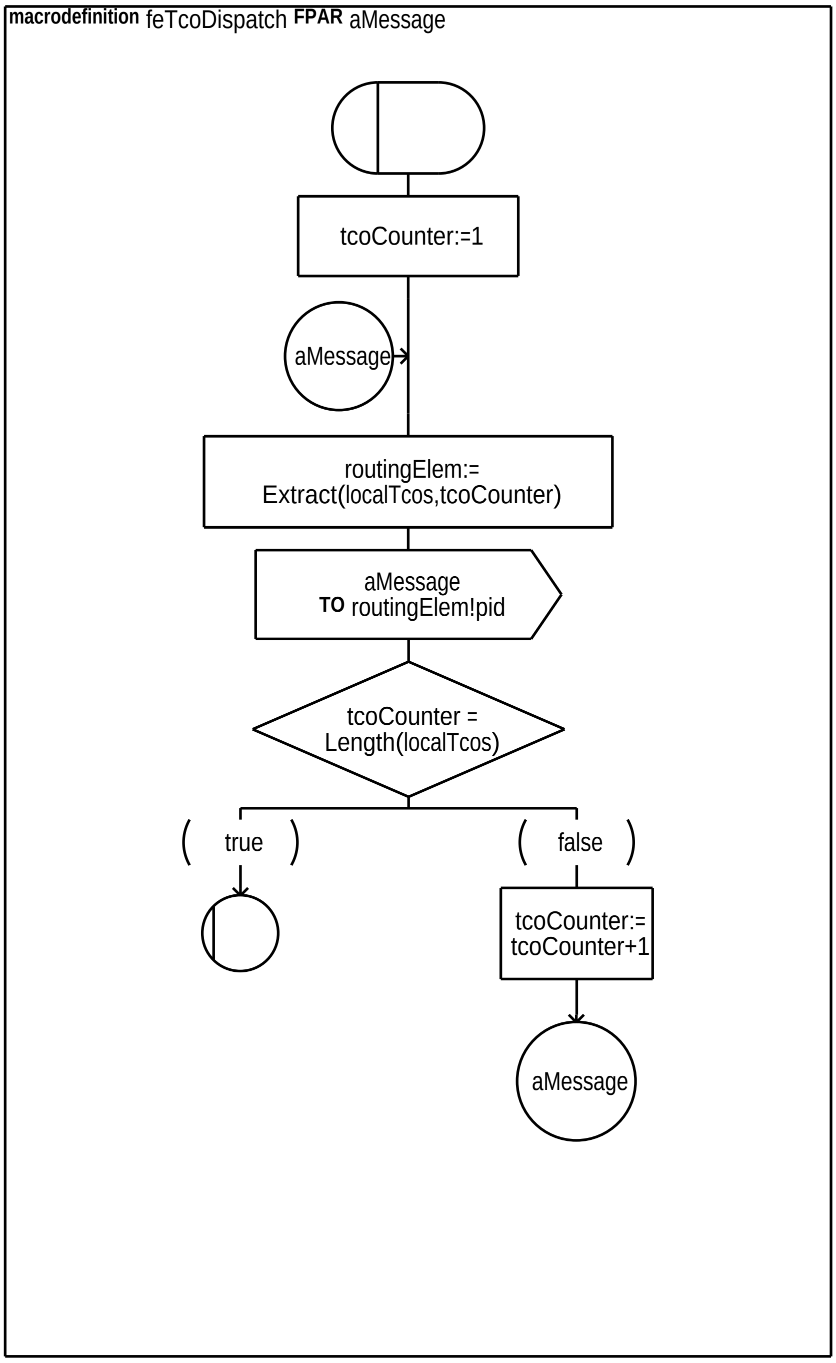

DispatchTillEnd
TspCancelOpInd (cancelOpPdu)
GetPid(cancelOpPdu!tcoId, tcoPid,runningTcos)
tcoPid=NULL
true
false
TspErrorReq(Make(tspCancelOpAsp, errBadTco,NullError)) TO SELF
tsiCancel (cancelOpPdu!tcoId) TO tcoPid
-
tsiCanceled (err)
GetId(SENDER, tcoId,runningTcos)
err!feError=errOK
true
false
TspCancelOpResp (Make(tcoId)) TO SELF
TspErrorReq(Make(tspCancelOpAsp, err!feError,Make(tcoId,''))) TO SELF
RemoveId(tcoId, runningTcos)
IsThisEnd

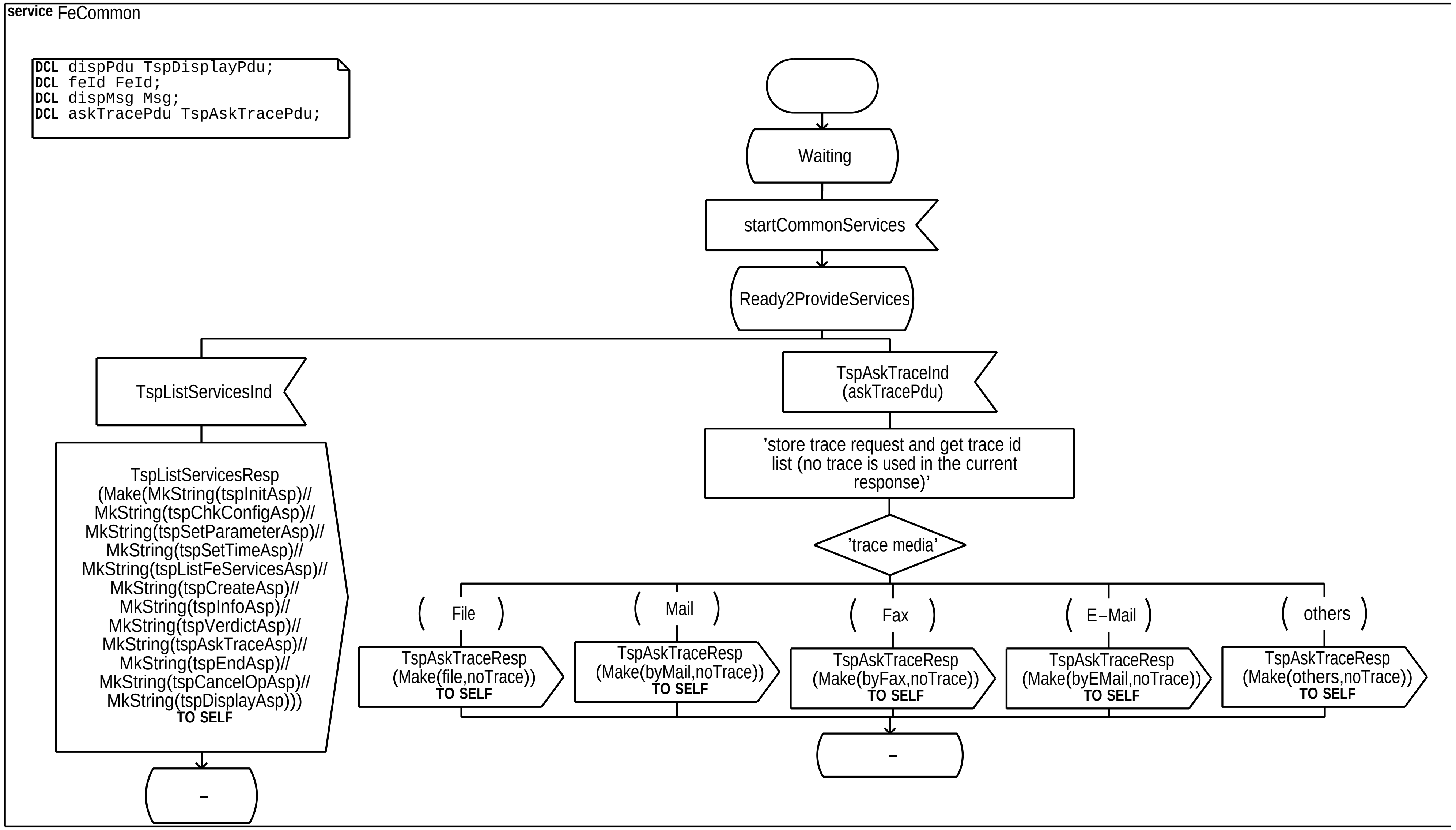


procedure routingInit
WaitingPong
ping
pong(myId)
WaitingPong

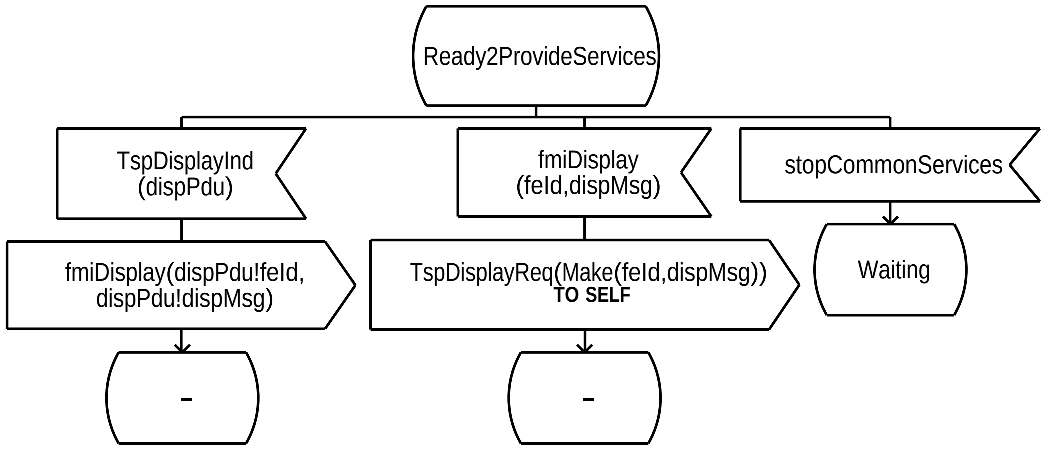


procedure initTestComponents
FPAR tcoIdList TcoIdList;
DCL tCounter TcoIndex;
DCL atcoId TcoId;
tcoIdList=noTco
false
true
tCounter:=1
localTcos:=empty
nextTco
atcoId:=Extract(tcoIdList,tcounter)
testComponent(atcoId)
AddElem(atcoId, **OFFSPRING**,localTcos)
tcounter=Length(tcoIdList)
false
true
tCounter:= tCounter+1
nextTco


macrodefinition feTcoDispatch FPAR aMessage
tcoCounter:=1
aMessage
routingElem:=
Extract(localTcos,tcoCounter)
aMessage
TO routingElem!pid
tcoCounter =
Length(localTcos)
true
false
tcoCounter:=
tcoCounter+1
aMessage

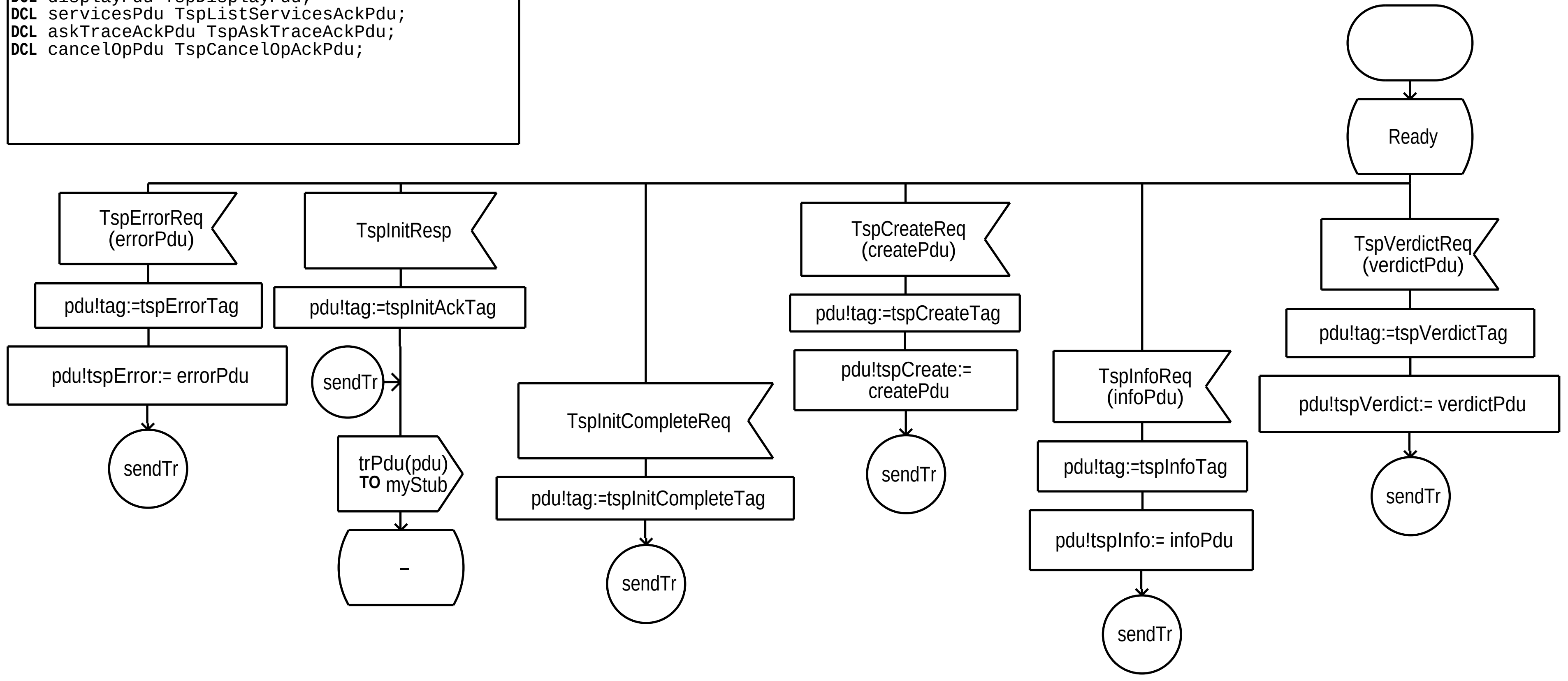


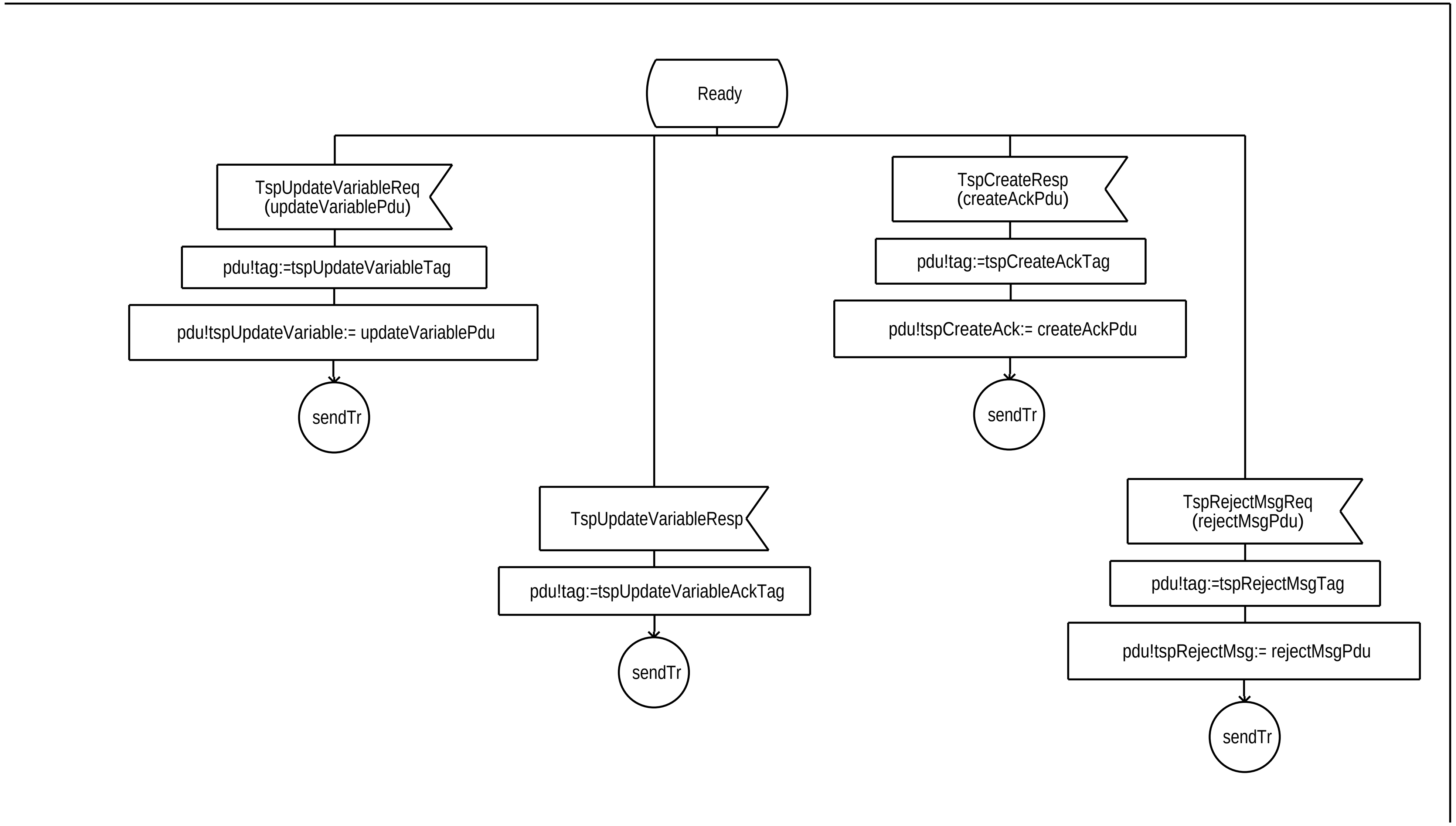
service FeCommon
DCL dispPdu TspDisplayPdu;
DCL feId FeId;
DCL dispMsg Msg;
DCL askTracePdu TspAskTracePdu;
Waiting
startCommonServices
Ready2ProvideServices
TspListServicesInd
TspListServicesResp (Make(MkString(tspInitAsp)// MkString(tspChkConfigAsp)// MkString(tspSetParameterAsp)// MkString(tspSetTimeAsp)// MkString(tspListFeServicesAsp)// MkString(tspCreateAsp)// MkString(tspInfoAsp)// MkString(tspVerdictAsp)// MkString(tspAskTraceAsp)// MkString(tspEndAsp)// MkString(tspCancelOpAsp)// MkString(tspDisplayAsp))) TO SELF
-
TspAskTraceInd (askTracePdu)
'store trace request and get trace id list (no trace is used in the current response)'
'trace media'
(File)
(Mail)
(Fax)
(E-Mail)
(others)
TspAskTraceResp (Make(file,noTrace)) TO SELF
TspAskTraceResp (Make(byMail,noTrace)) TO SELF
TspAskTraceResp (Make(byFax,noTrace)) TO SELF
TspAskTraceResp (Make(byEMail,noTrace)) TO SELF
TspAskTraceResp (Make(others,noTrace)) TO SELF
-

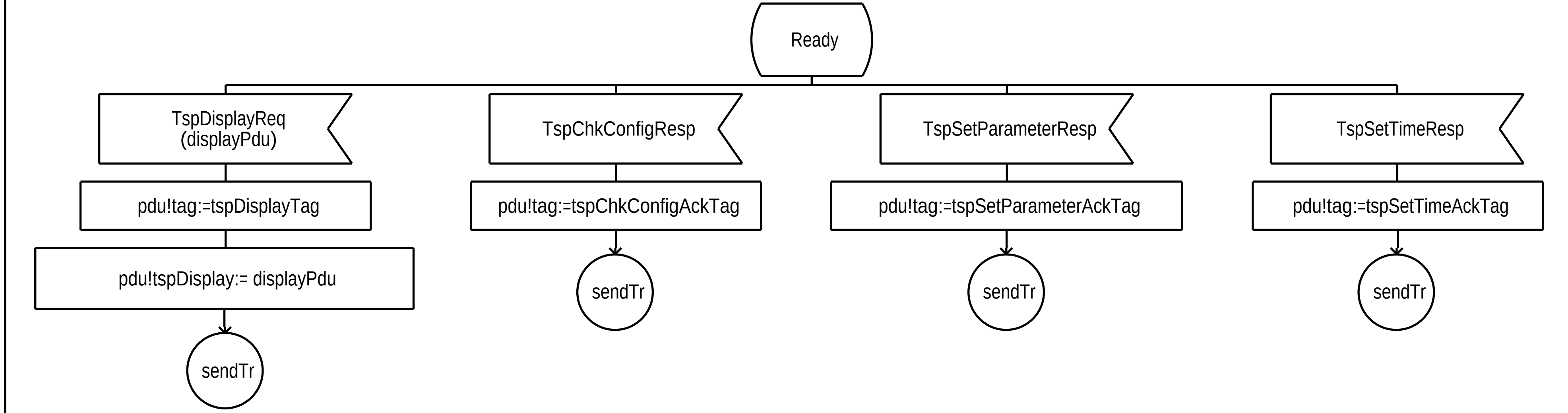

Ready2ProvideServices
TspDisplayInd (dispPdu)
fmiDisplay (feId,dispMsg)
stopCommonServices
fmiDisplay(dispPdu!feId, dispPdu!dispMsg)
TspDisplayReq(Make(feId,dispMsg)) TO SELF
Waiting
-
-

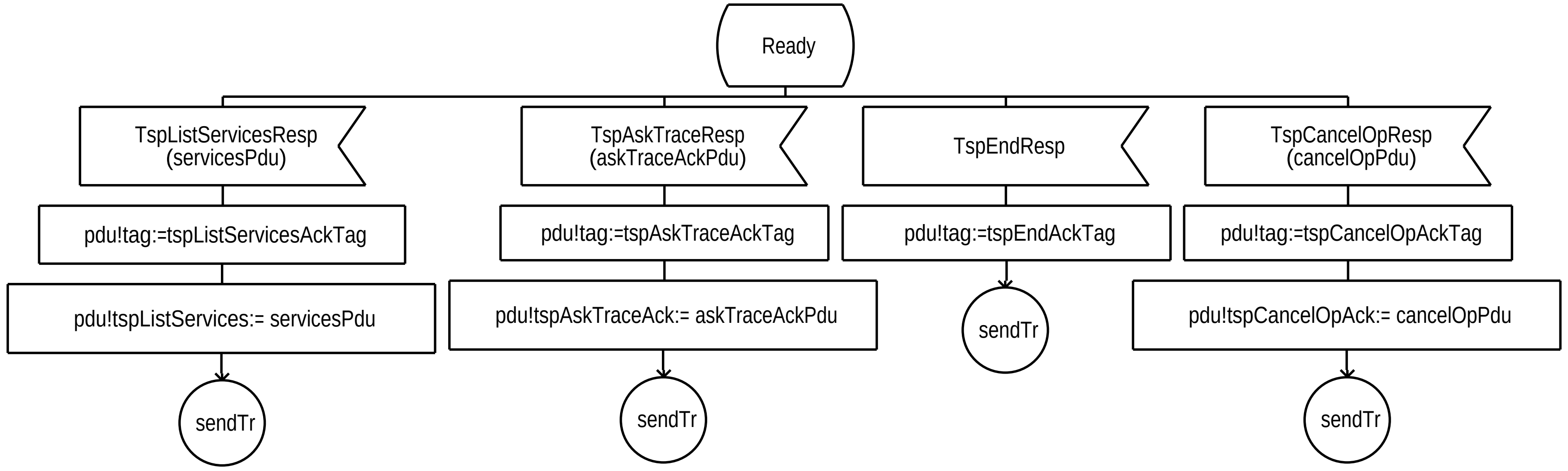
service FeTspEncoder

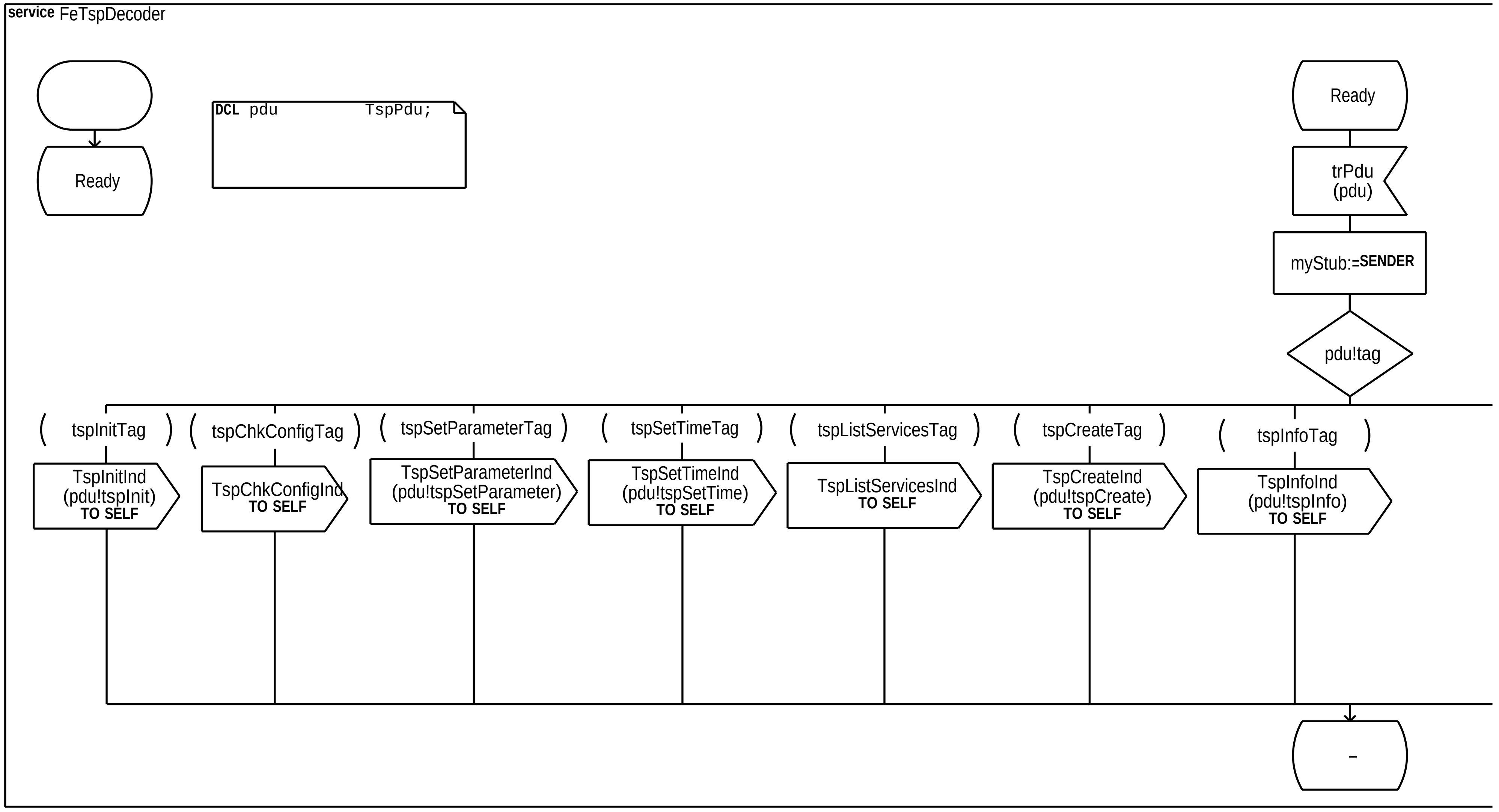
```
DCL pdu TspPdu;
DCL errorPdu TspErrorPdu;
DCL createPdu TspCreatePdu;
DCL createAckPdu TspCreateAckPdu;
DCL infoPdu TspInfoPdu;
DCL verdictPdu TspVerdictPdu;
DCL updateVariablePdu TspUpdateVariablePdu;
DCL rejectMsgPdu TspRejectMsgPdu;
DCL displayPdu TspDisplayPdu;
DCL servicesPdu TspListServicesAckPdu;
DCL askTraceAckPdu TspAskTraceAckPdu;
DCL cancelOpPdu TspCancelOpAckPdu;
```

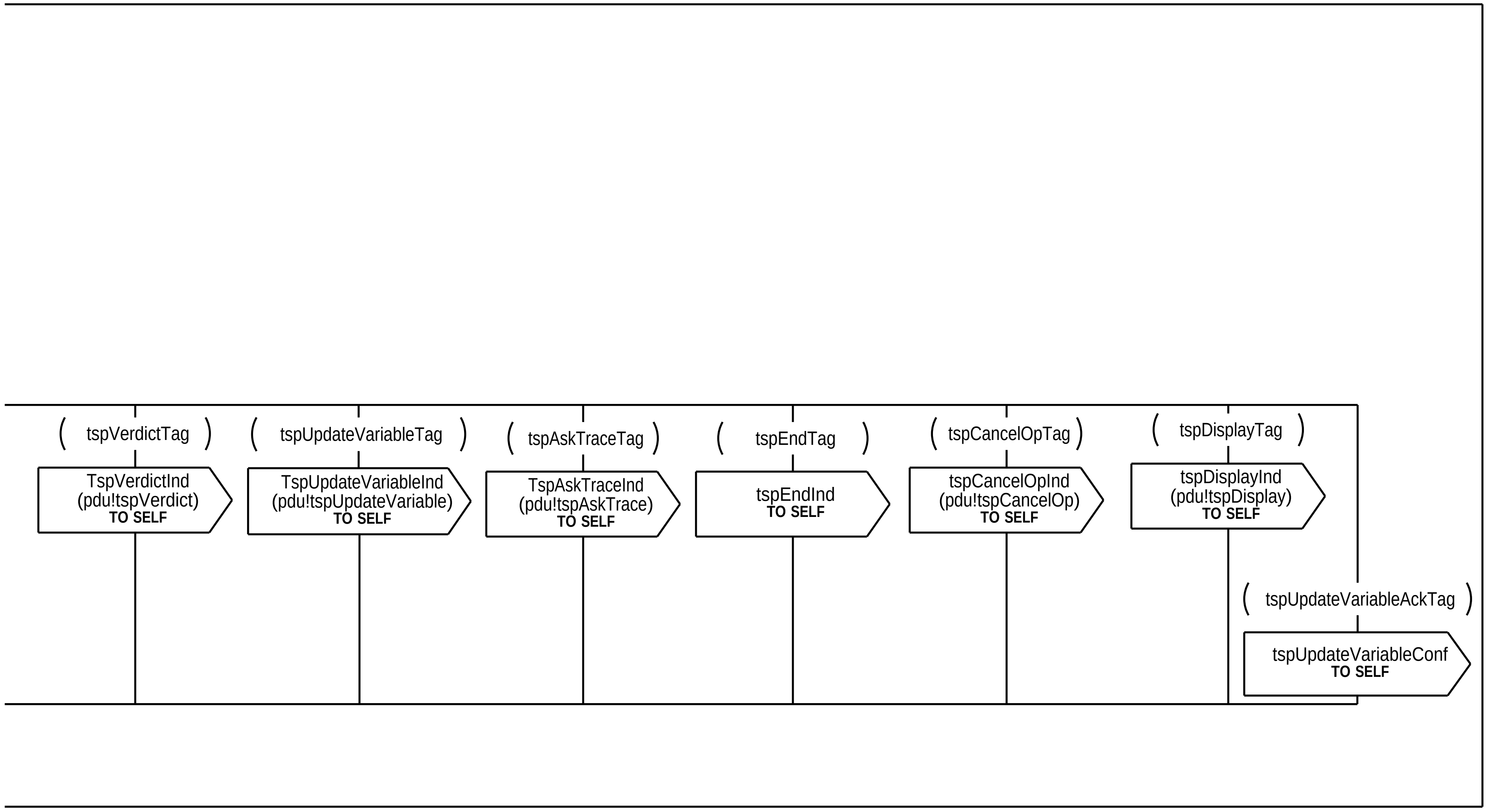


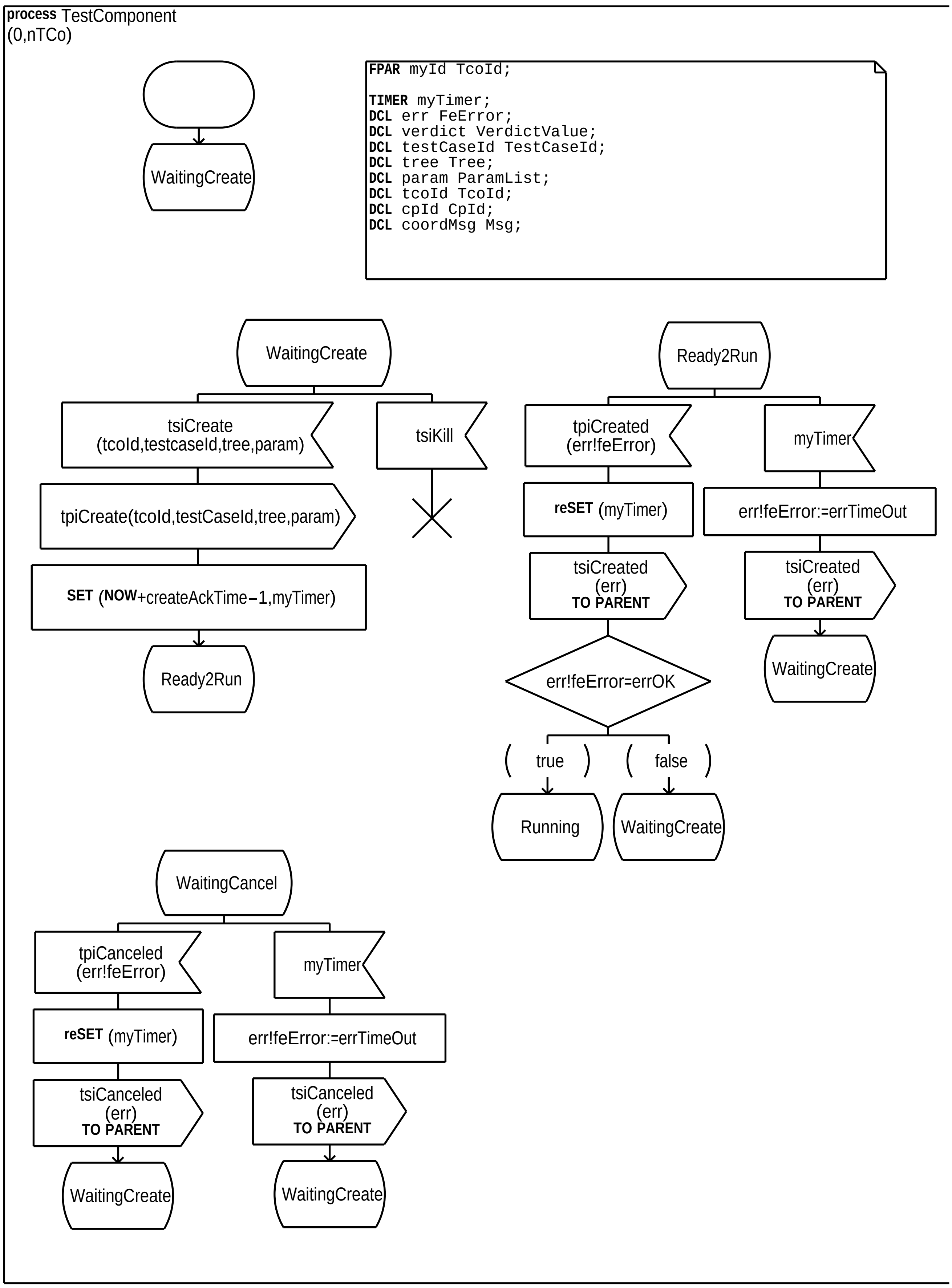

Ready
TspUpdateVariableReq (updateVariablePdu)
pdu!tag:=tspUpdateVariableTag
pdu!tspUpdateVariable:= updateVariablePdu
sendTr
TspUpdateVariableResp
pdu!tag:=tspUpdateVariableAckTag
sendTr
TspCreateResp (createAckPdu)
pdu!tag:=tspCreateAckTag
pdu!tspCreateAck:= createAckPdu
sendTr
TspRejectMsgReq (rejectMsgPdu)
pdu!tag:=tspRejectMsgTag
pdu!tspRejectMsg:= rejectMsgPdu
sendTr


Ready
TspDisplayReq (displayPdu)
pdu!tag:=tspDisplayTag
pdu!tspDisplay:= displayPdu
sendTr
TspChkConfigResp
pdu!tag:=tspChkConfigAckTag
sendTr
TspSetParameterResp
pdu!tag:=tspSetParameterAckTag
sendTr
TspSetTimeResp
pdu!tag:=tspSetTimeAckTag
sendTr

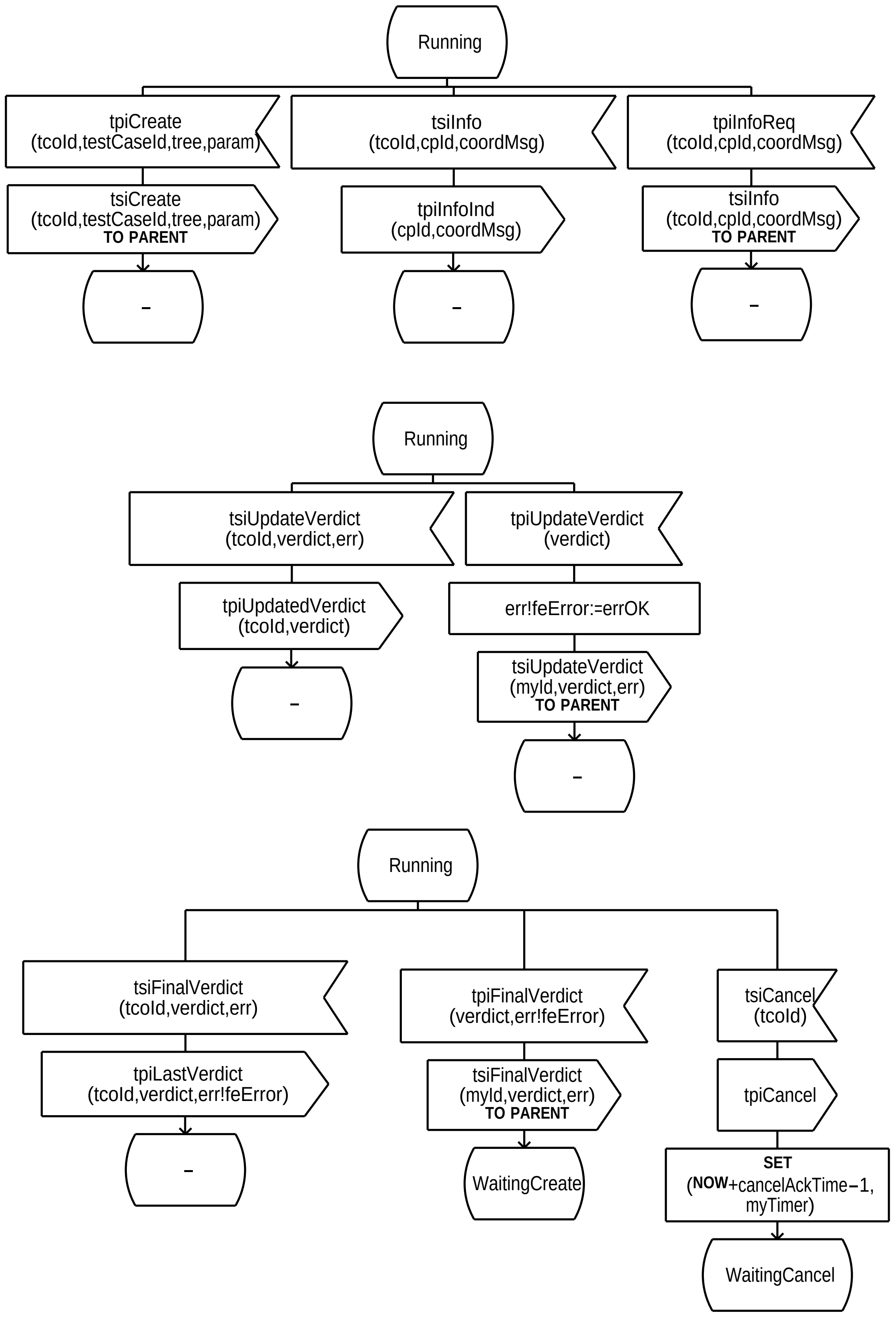

Ready
TspListServicesResp (servicesPdu)
pdu!tag:=tspListServicesAckTag
pdu!tspListServices:= servicesPdu
sendTr
TspAskTraceResp (askTraceAckPdu)
pdu!tag:=tspAskTraceAckTag
pdu!tspAskTraceAck:= askTraceAckPdu
sendTr
TspEndResp
pdu!tag:=tspEndAckTag
sendTr
TspCancelOpResp (cancelOpPdu)
pdu!tag:=tspCancelOpAckTag
pdu!tspCancelOpAck:= cancelOpPdu
sendTr


service FeTspDecoder
Ready
DCL pdu TspPdu;
Ready
trPdu (pdu)
myStub:=SENDER
pdu!tag
tspInitTag
tspChkConfigTag
tspSetParameterTag
tspSetTimeTag
tspListServicesTag
tspCreateTag
tspInfoTag
TspInitInd (pdu!tspInit) TO SELF
TspChkConfigInd TO SELF
TspSetParameterInd (pdu!tspSetParameter) TO SELF
TspSetTimeInd (pdu!tspSetTime) TO SELF
TspListServicesInd TO SELF
TspCreateInd (pdu!tspCreate) TO SELF
TspInfoInd (pdu!tspInfo) TO SELF
-


tspVerdictTag
TspVerdictInd (pdu!tspVerdict) TO SELF
tspUpdateVariableTag
TspUpdateVariableInd (pdu!tspUpdateVariable) TO SELF
tspAskTraceTag
TspAskTraceInd (pdu!tspAskTrace) TO SELF
tspEndTag
tspEndInd TO SELF
tspCancelOpTag
tspCancelOpInd (pdu!tspCancelOp) TO SELF
tspDisplayTag
tspDisplayInd (pdu!tspDisplay) TO SELF
tspUpdateVariableAckTag
tspUpdateVariableConf TO SELF



```
FPAR myId TcoId;

TIMER myTimer;
DCL err FeError;
DCL verdict VerdictValue;
DCL testCaseId TestCaseId;
DCL tree Tree;
DCL param ParamList;
DCL tcoId TcoId;
DCL cpId CpId;
DCL coordMsg Msg;
```



Running
tpiCreate (tcoId,testCaseId,tree,param)
tsiCreate (tcoId,testCaseId,tree,param) TO PARENT
-
tsiInfo (tcoId,cpId,coordMsg)
tpiInfoInd (cpId,coordMsg)
-
tpiInfoReq (tcoId,cpId,coordMsg)
tsiInfo (tcoId,cpId,coordMsg) TO PARENT
-
Running
tsiUpdateVerdict (tcoId,verdict,err)
tpiUpdatedVerdict (tcoId,verdict)
-
tpiUpdateVerdict (verdict)
err!feError:=errOK
tsiUpdateVerdict (myId,verdict,err) TO PARENT
-
Running
tsiFinalVerdict (tcoId,verdict,err)
tpiLastVerdict (tcoId,verdict,err!feError)
-
tpiFinalVerdict (verdict,err!feError)
tsiFinalVerdict (myId,verdict,err) TO PARENT
WaitingCreate
tsiCancel (tcoId)
tpiCancel
SET (NOW+cancelAckTime−1, myTimer)
WaitingCancel

Object	View	Page
SYSTEM TSP1		
PARTITION DefaultPartition	1	1
		2
		3
		4
		5
		6
		7
		8
		9
		10
PROCEDURE AddElem		
PARTITION DefaultPartition	2	11
PROCEDURE GetPid		
PARTITION DefaultPartition	3	12
PROCEDURE GetId		
PARTITION DefaultPartition	4	13
PROCEDURE RemoveId		
PARTITION DefaultPartition	5	14
BLOCK SystemSupervisor		
PARTITION DefaultPartition	6	15
PROCESS Supervisor(1,1)		
PARTITION DefaultPartition	7	16
		17
SERVICE SuMain		
PARTITION DefaultPartition	8	18
		19
		20
		21
		22
		23
		24
		25
		26
		27
		28
		29
PROCEDURE routingInit		
PARTITION DefaultPartition	9	30
PROCEDURE openSession		
PARTITION DefaultPartition	10	31
PROCEDURE getFrontEnd		
PARTITION DefaultPartition	11	32
PROCEDURE getSessionFEnds		
PARTITION DefaultPartition	12	33
PROCEDURE updateStubPid		
PARTITION DefaultPartition	13	34
PROCEDURE addFeError		
PARTITION DefaultPartition	14	35
PROCEDURE startTco		
PARTITION DefaultPartition	15	36
PROCEDURE addTcoError		

History

Document history	
January 1997	First Edition

ISBN 2 7437-0921 9
Dépôt légal : Janvier 1997