PSF - Q.930/Q.931 (FT/HA)

Software Test Sample
1094144 1.1

PSF - Q.930/Q.931 (FT/HA)

Software Test Sample
1094144 1.1

Trillium Digital Systems, Inc.
12100 Wilshire Blvd., Suite 1800
Los Angeles, CA 90025-7118
Phone: +1 (310) 442-9222
Fax: +1 (310) 442-1162
Web: http://www.trillium.com

PSF - Q.930/Q.931 (FT/HA)
Software Test Sample
1094144 1.1

Preface

Objective

This document describes the software test sample for the PSF - Q.930 / .931 (FT / HA) software (p / n 1000144) designed by Trillium Digital Systems, Inc. This product is referred to as PSF - Q.930 / Q.931 in the rest of the document.

Audience

Trillium assumes that the readers of this document are familiar with telecommunication protocols, specifically ISDN, Trillium's Fault Tolerant / High Availability (FT / HA) concepts, and Trillium's ISDN product.

Document Organization

This document is organized into the following sections:

Section	Description
1 Introduction	Provides information about the software tests and their usefulness
2 Overview	Describes the various test scenarios
3 Test Environment	Describes the protocol stack architecture for tests. It contains diagrams illustrating the architecture and the files used.
4 File Descriptions	Gives a detailed description of various files: acceptance test, product, system services, configuration, and test files.
5 Test Description	Describes the test methodology

Document Set

The suggested reading order for the PSF - Q.930 / Q.931 document set is:

1. *PSF - Q.930/.931 (FT/HA) Functional Specification*

   Highlights and describes the protocol and system characteristics of the software, including the memory characteristics and conformance details.

2. *PSF - Q.930/.931 (FT/HA) Service Definition*

   Describes the procedures and the layer manager interface used to pass information between the software and other software elements. The Interface Primitives section describes the services of the software. The Interface Procedures section describes and illustrates the flow of primitives and messages across the interfaces.

   Note: *Information on porting the software is contained in the Service Definition.*

3. *PSF - Q.930/.931 (FT/HA) Software Test Sample*

   Describes the sample files delivered with the product and the procedures to build a sample test. This test partially demonstrates the initialization, configuration, and execution of the product. It may contain data flow diagrams illustrating the correct operation of the software.

In addition to the above PSF documents, the following documents should also be read for a better understanding of the fault tolerant system:

4. *Fault Tolerant/High Availability (FT/HA) Core Functional Specification* (p / n 1091133).
5. *Fault Tolerant/High Availability (FT/HA) Core Service Definition* (p / n 1092133).

Notations

This table displays the notations used in this document:

Notation	Explanation	Examples
Arial	**Titles**	**1.1 Title**
Palatino	Body text	This is body text.
Bold	**Highlights information**	**Loose coupling, tight coupling, upper layer interface**
ALL CAPS	CONDITIONS, MESSAGES	AND, OR CONNECT ACK
Italics	*Document names, emphasis*	*PSF - Q.930/Q.931 (FT/HA) Software Test Sample* This adds *emphasis.*
Courier New Bold	**Code Filenames, pathnames**	**PUBLIC S16 ZqMiLzqCfgReq(pst, cfg)** **Pst *pst;** **CmPFthaMngmt *cfg;**

Release History

This table lists the history of changes in successive revisions to this document:

Version	Date	Initials	Description
1.1	09/30/99	pk	Initial release

Contents

Illustrations

1 INTRODUCTION

This document describes the software test sample for the Protocol Specific Function (PSF) - Q.930 / Q.931 Fault Tolerant / High Availability (FT / HA) software (p / n 1000144) designed by Trillium Digital Systems, Inc.

The software test sample is a collection of files, usually delivered with the released product, that have been written and used by Trillium in order to test the portable software. Since these test files interface heavily with the protocol layer, the files are extremely useful in porting the software.

Familiarity with Trillium software organization is required to better understand these test files and the organization of the test. The architecture for the PSF - Q.930 / Q931 software is described in the *PSF - Q.930/Q.931 Functional Specification* and the *PSF - Q.930/Q.931 Service Definition*.

Note: *These tests are hardware-independent.*

2 OVERVIEW

The objectives of the software test sample for the PSF - Q.930/Q.931 software are:

- To demonstrate the robustness of PSF - Q.930/Q.931 by running an extensive test suite designed to test the functionality of the software
- To demonstrate conformance to the *PSF - Q.930/Q.931 (FT/HA) Functional Specification*
- To demonstrate how PSF - Q.930/Q.931 can be configured and exercised
- To better understand the portation work required

The files that are provided as part of the software test sample for PSF - Q.930/Q.931 have been used to run tests on the product as part of Trillium's in-house software testing. These test files are not officially released code and they are not fully supported.

The specific test described in this document is hardware-independent. In order to emulate a variety of real situations, a simple configuration is used for this test. A specific script program is activated within the course of the test, which, in turn, launches a series of tests that emulate different classes of realistic situations. Once initiated, the test has been designed to run indefinitely without human intervention.

The acceptance test does not measure the performance of the PSF - Q.930/Q.931 software. Since the software under test is portable, any performance issues will depend on the environment.

Note: *These tests are not extensive and do not attempt to test every functionality of the software.*

2.1 Hardware Requirements

This test can run in virtually any computer hardware. It is recommended that the test run on an environment with good debugging capabilities, so that the test is fully utilized.

Trillium's standard environment for this test is a PC DOS machine (286 or 386), or a SUN workstation running Solaris 2.5.1.

2.2 Software Requirements

The test consists of C portable files, similar to the standard Trillium portable software. It is assumed that the test runs on a PC with an ANSI.SYS driver, or on a SUN workstation. The test provides stub modules for the service user and the service provider.

Trillium MOS or any operating system can be used to run these tests. When MOS is used, the only portation work required is that of porting MOS to the specific hardware platform. If MOS is not used, the interface to the system services must be mapped against the operating system used.

3 TEST ENVIRONMENT

This section describes the test environment for the PSF - Q.930/Q.931 software.

3.1 Testing Strategy

Like the rest of Trillium portable software deliverables, the sample files PSF - Q.930/Q.931 are portable C files that do not need special hardware to run. Once portation is done for PSF - Q.930/Q.931, the sample files will run without further effort. To run the tests, the sample files should be linked with both the Q.930/931 and PSF - Q.930/Q.931 software, as well as a system services provider. Within Trillium, the system services provider used is the Multiprocessor Operating System (MOS) product.

Figure 3-1 illustrates the PSF - Q.930/Q.931 test strategy:

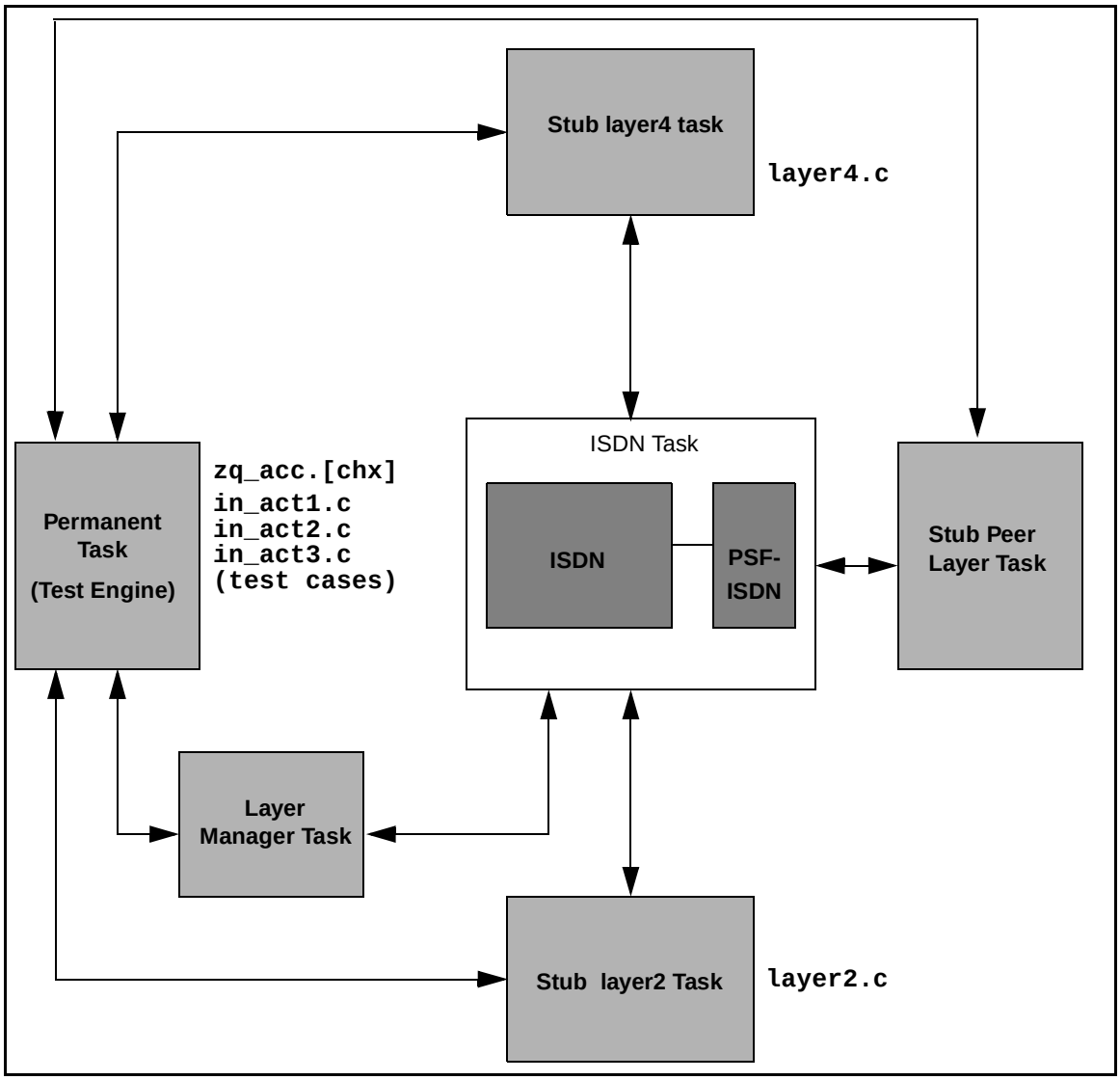

Figure 3-1: PSF - Q.930/Q.931 test strategy

In Figure 3-1, ISDN is linked with the following:

- A stub layer 4 task, which emulates the Call Control (CC)
- A stub layer 2 task, which emulates LAPD
- A stub layer manager task
- A stub peer layer task, which emulates the peer PSF - Q.930 / Q.931

A permanent task (**inPerm**) is also registered with system services. This permanent task is a low-priority task that drives the test engine. System services schedules the permanent task when no other task is to be scheduled. The permanent task injects events (for example, configuration request from the stack manager, data indication from layer 2, or data request from layer 4) into the Q.930 / Q.931 layer and PSF - Q.930 / Q.931.

In the case of loose coupling the scheduled stub task calls a function to give control to the test engine to process the received output. The test cases are the data for the test engine.

The PSF functionality involves state updates from active to standby. Some of the tests require both active and standby PSF - Q.930 / Q.931 to be present; however, only a single instance of PSF - Q.930 / Q.931 can run at one time in the test environment. Therefore the active and standby status is emulated by the same instance of Q.930 / Q.931 and PSF - Q.930 / Q.931 by changing their status (to active or standby) as required by various steps in the test.

For example, when the active PSF - Q.930 / Q.931 sends state update messages, the messages are queued by the stub peer-layer task. If the test requires the behavior of the standby to be tested, then the status of PSF - Q.930 / Q.931 is changed from active to standby and the queued update messages are applied to the standby.

4 FILE DESCRIPTIONS

The **envopt.acc** file delivered with the PSF - Q.930/Q.931 software contains the settings required to build the acceptance test. The **envopt.acc** file must be renamed to **envopt.h** in order to compile the software test sample. The **zq.mak** file compiles the source files required for PSF - Q.930/Q.931 testing. A detailed description of these files and their functions can be found in the *PSF - Q.930/Q.931 (FT/HA) Service Definition.*

This section describes the files needed to run the test.

4.1 System Services Files

These operating system files are portation-dependent. Trillium MOS is recommended. If Trillium MOS is used, the following files should be used:

Name	Description
ms_bdy1.c	MOS body 1
ms_bdy2.c	MOS body 2
ms_bdy5.c	MOS body 5
ms_id.c	MOS ID
ms_ex_ms.c	MOS extended
ms_ptmi.c	MOS portable layer manager interface

4.2 Q.930/Q.931 Files

These files are described in detail in the *ISDN Portation Guide*:

Name	Description
in_id.c	Q.930 / Q.931 - ID
in_bdy1.c	Q.930 / Q.931 - Body - Part 1
in_bdy2.c	Q.930 / Q.931 - Body - Part 2
in_bdy3.c	Q.930 / Q.931 - Body - Part 3
in_bdy4.c	Q.930 / Q.931 - Body - Part 4
in_bdy5.c	Q.930 / Q.931 - Body - Part 5
in_bdy6.c	Q.930 / Q.931 - Body - Part 6
in_bdy7.c	Q.930 / Q.931 - Body - Part 7
in_bdy8.c	Q.930 / Q.931 - Body - Part 8
in_bdy9.c	Q.930 / Q.931 - Body - Part 9
in_bdy10.c	Q.930 / Q.931 - Body - Part 10
in_bdy11.c	Q.930 / Q.931 - Body - Part 11
in_bdy12.c	Q.930 / Q.931 - Body - Part 12
in_bdy13.c	Q.930 / Q.931 - Body - Part 13
in_bdy14.c	Q.930 / Q.931 - Body - Part 14
in_ex_ms.c	Q.930 / Q.931 portable external interface
in_db1.c	Q.930 / Q.931 database - Part 1
in_db2.c	Q.930 / Q.931 database - Part 2
in_ptmi.c	Q.930 / Q.931 portable manager interface (product file)
mf.c	Message functions
mf.h	Include file
in_err.h	Include file
in.h	Include File (Q.930 / Q.931)
in.x	Include File (Q.930 / Q.931)

4.3 PSF - Q.930/Q.931 Files

Name	Description
zq_bdy1.c	PSF - Q.930/Q.931 - Body - Part 1
zq_bdy2.c	PSF - Q.930/Q.931 - Body - Part 2
zq_bdy3.c	PSF - Q.930/Q.931 - Body - Part 3
zq_bdy4.c	PSF - Q.930/Q.931 - Body - Part 4
zq_bdy5.c	PSF - Q.930/Q.931 - Body - Part 5
zq_bdy6.c	PSF - Q.930/Q.931 - Body - Part 6
zq_ex_ms.c	PSF - Q.930/Q.931 - Portable external interface
zq_ptmi.c	PSF - Q.930/Q.931 - Portable management interface
zq_ptpi.c	PSF - Q.930/Q.931 - Portable external interface
zq_id.c	PSF - Q.930/Q.931 ID
zq_err.h	Include file (error codes)
zq.h	Include file (PSF - Q.930/Q.931)
zq.x	Include file (PSF - Q.930/Q.931)

4.4 Common Files

int.h	Include file
dat.h	Include file
lin.h	Include file
mf.x	Include file
int.x	Include file
dat.x	Include file
lin.x	Include file
qn_db.h	Include file
qn_db.x	Include file
cm_lib.x	Include file
cm_lib.c	Common library functions
cm_hash.c	Hashing functions
cm_bdy5.c	Timers functions
cm_hash.x	Include file
cm5.x	Include file
cm5.h	Include file
cm_err.h	Include file
cm_pftha.c	FT / HA common functions
cm_pftha.h	Include file
cm_pftha.x	Include file
gen.h	Include file
gen.x	Include file
lzq.h	Include file
lzq.x	Include file

4.5 Lower Layer Files

layer2.c and **l2_ex_ms.c** are sample files. The remaining three are common files.

Name	Description
layer2.c	Emulates the LAPD service provider
in_ptli.c	Q.930/Q.931 portable lower interface
l2_ptui.c	Lower layer portable upper interface
l2_ex_ms.c	Disassembles task messages from Q.930/Q.931 to the lower layer. This task contains the entry point to the lower layer task (**daActvTskNew**), which is activated by the OS whenever there is a message for the lower layer.
dat.c	Common packing and unpacking functions required by LAPD service provider and Q.930/Q.931

4.6 Upper Layer Files

layer4.c and **l4_ex_ms.c** are sample files. The remaining three are common files.

Name	Description
layer4.c	Emulates the call control service user
in_ptui.c	Q.930/Q.931 portable upper interface
l4_ptli.c	Upper layer portable lower interface
l4_ex_ms.c	Disassembles task messages from Q.930/Q.931 to the upper layer. This task contains the entry point to the upper layer task (**anActvTskNew**), which is activated by the OS whenever there is a message for the upper layer.
int.c	Q.930/Q.931 upper interface packing/unpacking functions

4.7 Layer Manager Files

The key file, **sminbdy1.c**, is a sample file, along with **sminexms.c** and **sminptmi.c**. The remaining files are product files.

Name	Description
sminbdy1.c	This file emulates the layer manager for Q.930 / Q.931
sminptmi.c	Layer manager portable interface. This file is used to pack the stack management interface functions.
sminexms.c	This file is used to disassemble task messages from Q.930 / Q.931 to the layer manager. This task contains the entry point to the layer manager task (**smInActvTsk**).
smzqbdy1.c	This file emulates the layer manager for the PSF - Q.930 / Q.931
smzqexms.c	Layer manager portable interface. This file is used to pack the stack management interface functions for the PSF - Q.930 / Q.931.
smzqptmi.c	This file is used to disassemble task messages from PSF - Q.930 / Q.931 to the layer manager. This task contains the entry point to the layer manager task.
in_ptmi.c	Management loosely coupled service user
lin.c	Packing / unpacking functions for Q.930 / Q.931 layer manager interface

4.8 Software Test Sample Files

The sample files are now described in more detail:

Name	Description
in_acc.c	Permanent task for the test
in_acc2.c	Configuration required by Q.930/Q.931
in_acc.x	Structures and declarations required by the test
in_acc.h	Defines required by the test
in_acc.mak	This is the sample **makefile** for compiling the source files and linking all the objects to create the **.exe** file. This is a DOS/Unix **makefile**. This file has been created from **in_ptcs.mak** (product file).
in_acc.lnk	This sample file defines the files to be linked. This is a DOS **linkfile**.
in_act1.c	Q.930/Q.931 software test cases
in_act2.c	Q.930/Q.931 software test cases
in_act3.c	Q.930/Q.931 software test cases
zq_acc.c	PSF - Q.930/Q.931 software test cases
zq_acc.x	Structures and declarations required by PSF - Q.930/Q.931
zq_acc.x	Defines required by the PSF - Q.930/Q.931 test

5 TEST DESCRIPTION

This section describes the tests for the PSF - Q.930 / Q.931 software.

5.1 Sample Configuration

Figure 5-1 illustrates the layout of the sample network configuration:

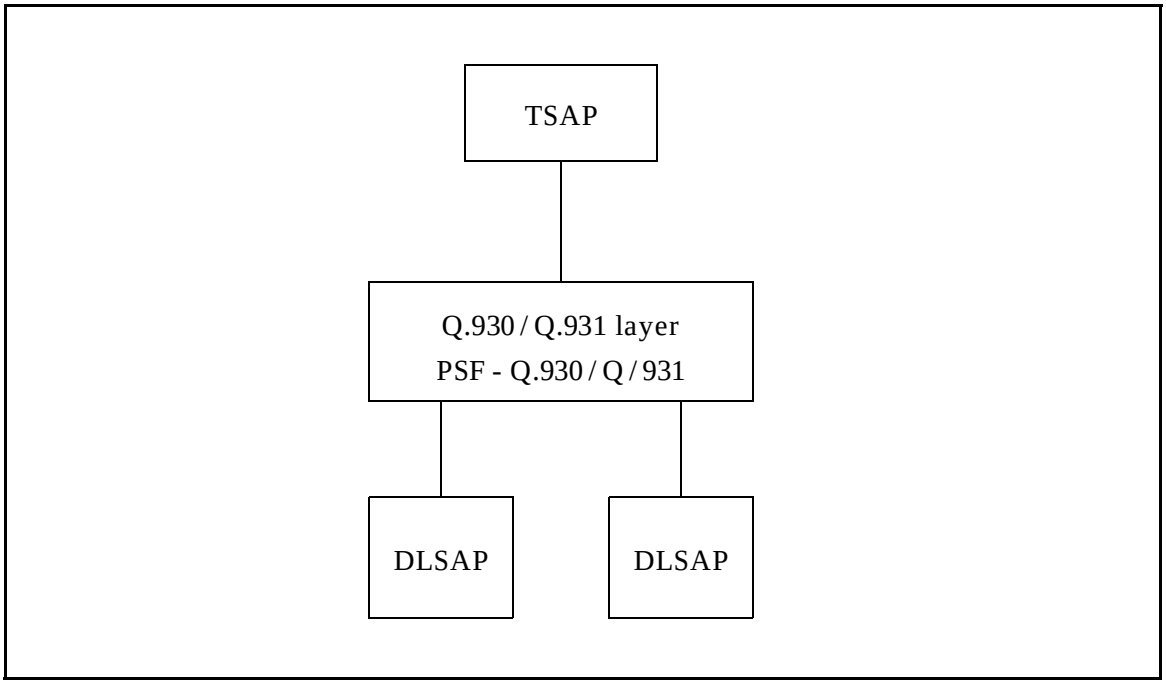

Figure 5-1: Sample network configuration

5.1.1 General Configuration

As a part of the general configuration, one TSAP two DLSAPs, two ces, and 32 channels per DLSAP are configured.

5.1.2 Physical Link Configuration

Two DLSAPS with 32 bearer channels each are configured.

5.1.3 Transport SAP Configuration

One upper SAP is configured for Q.930 / Q.931.

5.1.4 Ces Configuration

One ces is configured for each DLSAP.

5.1.5 Routing Configuration

No routing configuration was done.

5.1.6 Bearer Profile Configuration

One bearer profile was configured.

5.2 Test Case Description

The software test sample consists of a set of test functions that implement state machines and are dispatched in sequence by the permanent task. The entry point for this permanent task is **`inPerm`** in the **`in_acc.c`** file. The actual test sequences are located in **`zq_acc.c`**, **`in_act1.c`**, **`in_act2.c`** and **`in_act3.c`**. The test is activated when the system services invokes the **`inPerm`** function within the **`in_acc.c`** file.

The scope of one test case is one scheduling (or a timer expiry); multiple test cases are linked together to make a test scenario. For example, testing an incoming successful call is a test scenario consisting of multiple test cases, from configuration to releasing the call. Multiple test scenarios are categorized into test groups. A test group is a logical classification of the test scenarios. For example, all test scenarios related to configuration are categorized in a single group.

Each test is associated with a particular value of the task controller's state variable. The dispatching of a new test function happens as soon as the preceding one has reached the initial state, terminating its processing. After completion of a test cycle, the sequence is repeated indefinitely.

5.2.1 Test Preparation & Execution

The **zq.mak** sample file compiles the Q.930 / Q.931 and PSF - Q.930 / Q.931 code to generate the **in_acc** executable. The executable runs a series of test cases in a loop.

When **in_acc** is executed, the **tst** function is invoked. This function registers a task for every layer with the local OS. The following tasks are registered:

- Q.930 / Q.931. Entry point is **inActvTskNew**.
- Lower layer. Entry point is **daActvTskNew**.
- Upper layer. Entry point is **anActvTskNew**.
- PSF - Q.930 / Q / 931 layer. Entry point is **zqActvTskNew**.

A typical test case is as follows:

1. The test function configures Q.930 / Q.931 and PSF - Q.930 / Q.931.
2. Calls are setup on SAPs.
3. After call establishment, a switchover is initiated.
4. Messages are sent to the standby (the new active Q.930 / Q.931) to check if all calls and information have been transferred to the standby during switchover.
5. The Q.930 / Q.931 layer and the PSF is shutdown.

5.2.2 Assumptions

The following assumptions are made for each test case description:

- Each step of a test procedure is independent; the test engine performs only one step at a time. After executing a step, the test should deschedule itself for a sufficient period of time before performing the next step in the sequence (if required). This ensures that all the messages resulting from the first step reach the destination entities and that the destination entities have operated on them before the next step is performed.
- Before starting the execution of a test, the protocol layer and the PSF - Q.930 / Q.931 reset to the same initial state that existed before configuration.
- Testing strategy assumes that the permanent task (that is, the test engine) is scheduled after all other layers (Q.930 / Q.931 and stub layers) have completed their processing. Running the test engine, therefore, has the lowest priority. Any violation in this assumption can result in incorrect results for the tests.
- Once scheduled, the test engine (or the permanent task) should not be pre-empted by any stub layer or by Q.930 / Q.931 prior to giving input for the test case.

5.3 Test Configuration

Figure 5-2 illustrates the test engine configuration:

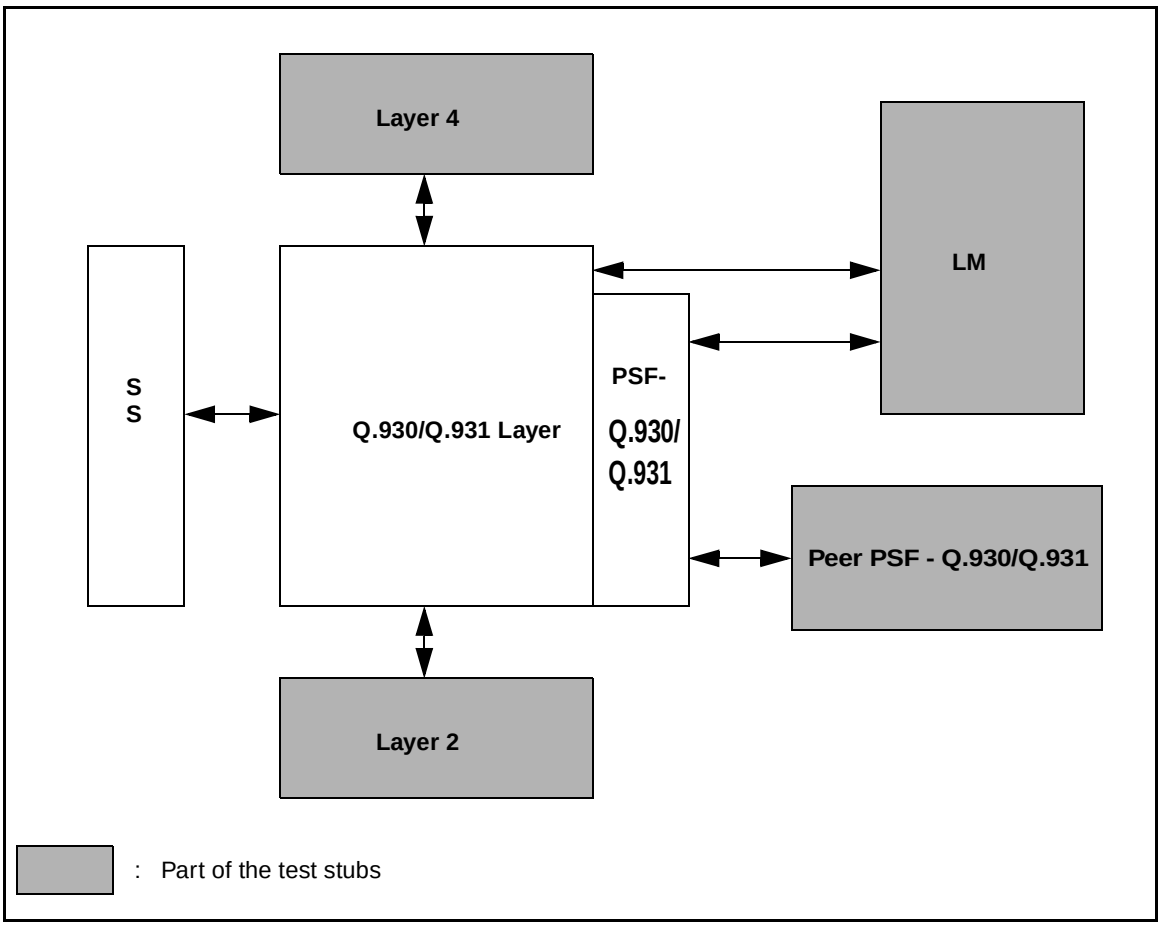

Figure 5-2: Test engine configuration

The test stubs simulate the lower layer, upper layer, layer manager interface, and peer PSF around the Q.930/Q.931 layer under test. Figure 5-2 illustrates the interaction of the test stub parts with the Q.930/Q.931 layer. The test stub around the Q.930/Q.931 and the PSF - Q.930/Q.931 layer interacts with the test engine to verify the received event and to generate the next input. This configuration is independent of the status of the layer; that is, the layer can be in-service, standby, or OOS. The peer PSF - Q.930/Q.931 part of the test engine handles update messages coming from the Q.930/Q.931 layer for run-time, warmstart, or controlled switchover.

5.4 Test Cases

This section describes the test cases required to test the functionality of PSF - Q.930/Q.931. The test cases are grouped according to function.

ID No.	Test Case Name	Test Case Descriptions
1.	zqTest1_1_1	Run-time update test - normal call processing case
2.	zqTest1_2_1	Run-time update test - restart procedures
3.	zqTest1_3_1	Run-time update test - service procedures
4.	zqTest2_1_1	Warmstart switchover test case
5.	zqTest3_1_1	Synchronization test

5.4.1 Test ID 1: ZqTest1_1_1

This test checks the functionality of the peer PSF when it receives a run-time update from the active PSF during normal call setup/clearing procedures.

Test Description

1. Configures PSF - Q.930/Q.931.
2. Configures Q.930/Q.931 layer with two DLSAPs, 1 TSAP, and one ces.
3. Sends a GO_ACTIVE control request to PSF to make it active.
4. Binds and connects the Q.930/Q.931 SAP when a control confirm is received from the layer manager indicating success of the GO_ACTIVE requests.
5. Initiates a call setup from layer 4.
6. Sends out the appropriate responses (Call Proceeding, Alerting, Connect) to Q.930/Q.931 when the setup message is received at layer 2.
7. Sends out a CONNECT ACK when a connect confirm is received at layer 4.
8. Shuts down the Q.930/Q.931 layer.
9. Shuts down the active PSF.
10. Configures the Q.930/Q.931 and the PSF - Q.930/Q.931 again.
11. Sends a GO_STBY (Go Standby) control request to the PSF.
12. Shoots all the pending update messages to the PSF once the control confirm is received.
13. Sends a GO_ACTIVE control request to PSF to make it active.
14. Sends out a disconnect for the call which was setup before the switchover.
15. On receiving the disconnect at layer 2, responds with release message.
16. The test case is considered successful if layer 4 receives a release indication and a release complete is sent out for the call. This implies that the call details were transferred accurately to the standby during run-time update.

5.4.2 Test ID 2: ZqTest1_2_1

This test checks the functionality of the peer PSF when it receives a run-time update from the active PSF during restart procedures.

Test Description

1. Configures PSF - Q.930/Q.931.
2. Configures Q.930/Q.931 layer with two DLSAPs, 1 TSAP, and one ces.
3. Sends a GO_ACTIVE control request to PSF to make it active.
4. Binds and connects the Q.930/Q.931 SAP when a control confirm is received from the layer manager indicating success of the GO_ACTIVE requests.
5. Initiates two call setups from layer 4.
6. Sends out the appropriate responses (Call Proceeding, Alerting, Connect) to Q.930/Q.931 when the setup messages are received at layer 2.
7. Sends out a CONNECT ACK when the connect confirms for each call are received at layer 4.
8. Sends a restart message for the interface.
9. Start switchover procedures once the RESTART ACK is received from Q.930/Q.931.
10. Shuts down the Q.930/Q.931 layer.
11. Shuts down the active PSF.
12. Configures the Q.930/Q.931 and the PSF - Q.930/Q.931 again.
13. Sends a GO_STBY (Go Standby) control request to the PSF.
14. Shoots all the pending update messages to the PSF once the control confirm is received.
15. Sends a GO_ACTIVE control request to PSF to make it active.
16. Sends out a disconnect for the call which was setup before the switchover.
17. The test case is considered successful if layer 4 receives a release indication for the call. The call should not exist on the standby since it would have been deleted when the interface was being restarted on the active side. If a release indication with the appropriate cause was received at layer 4 it would imply that the call details were transferred accurately to the standby during run-time update.

5.4.3 Test ID 3: ZqTest1_3_1

This test checks the functionality of the peer PSF when it receives a run-time update from the active PSF during D Channel switchover procedures.

Test Description

1. Configures PSF - Q.930/Q.931.
2. Configures Q.930/Q.931 layer with two DLSAPs (one signalling and the other as a backup), 1 TSAP, and one ces.
3. Sends a GO_ACTIVE control request to PSF to make it active.
4. Binds and connects the Q.930/Q.931 SAP when a control confirm is received from layer manager indicating success of the GO_ACTIVE requests.
5. Initiates two call setups from layer 4.
6. Sends out the appropriate responses (Call Proceeding, Alerting, Connect) to Q.930/Q.931 when the setup messages are received at layer 2.
7. Sends out a CONNECT ACK when the connect confirms for each call are received at layer 4.
8. Sends a service message on the backup D Channel.
9. Waits for a service acknowledge on the backup D Channel.
10. Shuts down the Q.930/Q.931 layer.
11. Shuts down the active PSF.
12. Configures the Q.930/Q.931 and the PSF - Q.930/Q.931 again.
13. Sends a GO_STBY (Go Standby) control request to the PSF.
14. Shoots all the pending update messages to the PSF once the control confirm is received.
15. Sends a GO_ACTIVE control request to PSF to make it active.
16. Sends out a disconnect for the call on the new signalling interface. This call was originally setup on the old signalling interface (i.e., before switchover).
17. On receiving the disconnect at layer 2, responds with release message.
18. The test case is considered successful if layer 4 receives a release indication and a release complete is sent out for the call. This implies that the call transfer from signalling to backup during switchover was transferred accurately to the standby during run-time update.

5.4.4 Test ID 4: ZqTest2_1_1

This test checks the functionality of the peer PSF when it receives a warmstart update from the active PSF during normal call setup / clearing procedures.

Test Description

1. Configures PSF - Q.930 / Q.931.
2. Configures Q.930 / Q.931 layer with two DLSAPs, 1 TSAP, and one ces.
3. Sends a GO_ACTIVE control request to PSF to make it active.
4. Binds and connects the Q.930 / Q.931 SAP when a control confirm is received from layer manager indicating success of the GO_ACTIVE requests.
5. Initiates a call setup from layer 4.
6. Sends out the appropriate responses (Call Proceeding, Alerting, Connect) to Q.930 / Q.931 when the setup message is received at layer 2.
7. Sends out a CONNECT ACK when a connect confirm is received at layer 4.
8. Sends a control request to warmstart the peer PSF.
9. Shuts down the Q.930 / Q.931 layer after receiving control confirm for warmstart.
10. Shuts down the active PSF.
11. Configures the Q.930 / Q.931 and the PSF - Q.930 / Q.931 again.
12. Sends a GO_STBY (Go Standby) control request to the PSF.
13. Shoots all the pending update messages to the PSF once the control confirm is received.
14. Sends a GO_ACTIVE control request to PSF to make it active.
15. Sends out a disconnect for the call which was setup before the switchover.
16. On receiving the disconnect at layer 2, responds with release message.
17. The test case is considered successful if layer 4 receives a release indication and a release complete is sent out for the call. This implies that the call details were transferred accurately to the standby during warmstart.

5.4.5 Test ID 5: ZqTest3_1_1

This test checks the functionality of the peer PSF when it receives a synchronize update from the active PSF during normal call setup / clearing procedures.

Test Description

1. Configures PSF - Q.930 / Q.931.
2. Configures Q.930 / Q.931 layer with two DLSAPs, 1 TSAP, and one ces.
3. Sends a GO_ACTIVE control request to PSF to make it active.
4. Binds and connects the Q.930 / Q.931 SAP when a control confirm is received from layer manager indicating success of the GO_ACTIVE requests.
5. Initiates a call setup from layer 4.
6. Before the call reaches an active state a switchover is initiated through a control request.
7. Shuts down the Q.930 / Q.931 layer after receiving a control confirm for the synchronize control request.
8. Shuts down the active PSF.
9. Configures the Q.930 / Q.931 and the PSF - Q.930 / Q.931 again.
10. Sends a GO_STBY (Go Standby) control request to the PSF.
11. Shoots all the pending update messages to the PSF once the control confirm is received.
12. Sends a GO_ACTIVE control request to PSF to make it active.
13. Sends out the appropriate responses (Call Proceeding, Alerting, Connect) to Q.930 / Q.931 setup message which was received before the synchronize procedure.
14. The test cases will be considered successful if these call setup messages are passed for the call as it would imply that the call information was transferred to the standby.
15. After the call gets to active state it sends out a disconnect for the call which was setup before the switchover.
16. On receiving the disconnect at layer 2, responds with release message.
17. The test case is considered successful if layer 4 receives a release indication and a release complete is sent out for the call. This implies that the call details were transferred accurately to the standby during the synchronize procedure.

Abbreviations

The following abbreviations are used in this document:

Abbreviation	Definition
CC	Call Control
DLSAP	Data Link Service Access Point
FT / HA	Fault-Tolerant / High-Availability
ISDN	Integrated Services Digital Network
LAPD	Link Access Protocol - D channel
LM	Layer Manager
MOS	Multiprocessor Operating System
OOS	Out-Of-Service
PSF	Protocol Specific Function
SAP	Service Access Point
SS	System Services
TAPA	Trillium Advanced Portability Architecture
TSAP	Transport Service Access Point

References

Refer to the following documents for more information:

Fault-Tolerant/High-Availability (FT/HA) Core Functional Specification, Trillium Digital Systems, Inc. (p / n 1091133).

Fault-Tolerant/High-Availability (FT/HA) Core Service Definition, Trillium Digital Systems, Inc. (p / n 1092133).

INT Interface Service Definition, Trillium Digital Systems, Inc. (p / n 1100018).

PSF - Q.930/Q.931 (FT/HA) Functional Specification, Trillium Digital Systems, Inc. (p / n 1091144).

PSF - Q.930/Q.931 (FT/HA) Service Definition, Trillium Digital Systems, Inc. (p / n 1092144).

Q.930/Q.931 Functional Specification, Trillium Digital Systems, Inc. (p / n 1091009).

Q.930/Q.931 Portation Guide, Trillium Digital Systems, Inc. (p / n 1093009).

Q.930/Q.931 Service Definition, Trillium Digital Systems, Inc. (p / n 1092009).

Q.930/Q.931 Software Test Sample, Trillium Digital Systems, Inc. (p / n 1094009).

Q.930/Q.931 Training Course, Trillium Digital Systems, Inc. (p / n 1095009).

Q.930 (I.450) - ISDN User-Network Interface Layer 3, ITU.

Q.931 (I.451) - ISDN User-Network Interface Layer 3 Specification for Basic Call Control, ITU.

Q.932 (I.452) - ISDN User-Network Interface Layer 3 Specification - Generic Procedures for the Control of ISDN Supplementary Services, ITU.

System Services Interface Service Definition, Trillium Digital Systems, Inc. (p / n 1111001).