PSF - ISUP (FT/HA)

Service Definition
1092146 1.2

PSF - ISUP (FT/HA)

Service Definition
1092146 1.2

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 - ISUP (FT/HA)
Service Definition
1092146 1.2

Preface

Objective

This document provides a detailed description of the services provided by the PSF - ISUP (FT / HA) software (p / n 1000146) designed by Trillium Digital Systems, Inc.

Audience

Trillium assumes that the readers of this document are familiar with telecommunication protocols, specifically SS7 and Trillium's ISUP, PSIF - ISUP, and Fault-Tolerant / High Availability Core products.

Document Organization

This document is organized into the following sections:

Section	Description
1 Introduction	Provides a textual and graphical overview of the product. It also contains Trillium-specific abbreviations.
2 Environment	Describes the assumptions about the operating environment for PSF - ISUP software
3 Interface Primitives	Explains the interface primitives at the PSF - ISUP layer interfaces
4 Interface Procedures	Defines the interface procedures
5 Portation Issues	Describes portation requirements for PSF - ISUP

Document Set

The suggested reading order of this document set is:

1. *Functional Specification*

   Contains the features and highlights that describe the protocol and system characteristics. It includes the memory characteristics and conformance details.

2. *Training Course*

   Offers a detailed overview of the features and interfaces of the software. It contains code samples, data flow diagrams, and a list of files.

3. *Service Definition*

   Describes the procedures and 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.*

4. *Software Test Sample*

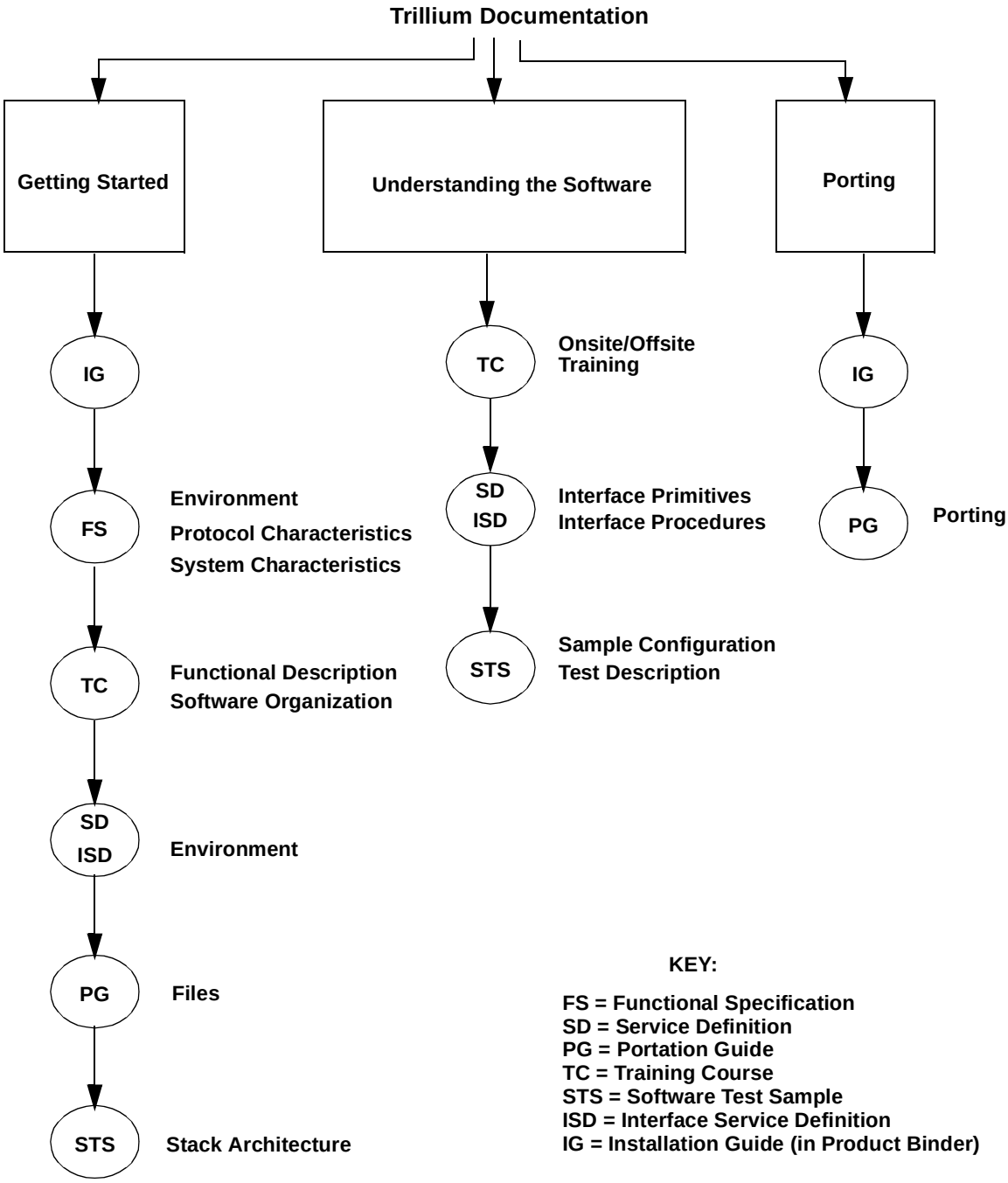
   Describes the sample files delivered with the product and the procedures to build a sample test. This test partially demonstrates the product initialization, configuration, and execution. It may contain data flow diagrams illustrating the correct operation of the software.

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

1. *Fault-Tolerant/High-Availability (FT/HA) Core Functional Specificatio*n
2. *Fault-Tolerant/High-Availability (FT/HA) Core Service Definition*

Using Trillium Documentation

The figure below illustrates the various approaches the user can take when utilizing the software documentation. First time users should read the documents under the **Getting Started** column; important sections and subsections are listed to the right of each document. For users familiar with the documentation but who need to look up certain points concerning the use of the software, the **Understanding the Software** column is suggested. The **Porting** column is for those users who are familiar with Trillium software and related telecommunications protocols and who wish to install the software immediately onto their operating systems.



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 - ISUP (FT/HA) Service Definition* This adds *emphasis*.
Courier New Bold	**Code Filenames, pathnames**	**PUBLIC S16 ZiMiLziCfgReq(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
1.2	December 31, 1999	sk	Changes for software release 1.2, including: • Addition of multiple point code support • Addition of NTT and Bellcore variants • FT / HA support for PSIF - ISUP
1.1	November 16, 1998	rs	• Initial release for software version 1.1

Contents

Illustrations

1 INTRODUCTION

This document describes the services provided by the Protocol Specific Function - ISDN User Part (PSF - ISUP) software designed by Trillium Digital Systems, Inc. This document describes the primitives and the procedures supported by PSF - ISUP at the layer manager interface.

PSF - ISUP adds Fault-Tolerant / High-Availability (FT / HA) functionality to Trillium's ISUP (PSIF - ISUP) product. From an ISUP (PSIF - ISUP) standpoint, PSF - ISUP is a library of functions that ISUP (PSIF - ISUP) invokes only in a fault-tolerant environment.

The PSF - ISUP software provides interfaces to perform the following functions:

- Run-time state update of standby
- Warmstart of an Out-Of-Service (OOS) node to make it standby
- Controlled switchover of active and standby ISUP (PSIF - ISUP) nodes
- Forced switchover on the failure of an active ISUP (PSIF - ISUP) node

PSF implements functions as required to make ISUP (PSIF - ISUP) fault-tolerant. Only the active copy of ISUP (PSIF - ISUP) participates in the protocol execution. While handling an event, the active ISUP (PSIF - ISUP) can modify its internal states. The active PSF updates the state changes to the standby to keep the standby synchronized with the active. The active PSF updates only high-level, stable ISUP (PSIF - ISUP) states to the standby to reduce the run-time update overhead and the complexity involved in the update procedure. Transient states (that is, the states that have a small duration) are only updated at the time of a controlled switchover and not during run time. Because any existing transient state information is lost during a forced switchover, the calls in the transient state are lost. The standby resumes operation with its current, stable states.

The state update procedure can be initiated either by an active node or by the system manager. The active ISUP (PSIF - ISUP) node initiates the state update procedure when its state changes as part of the normal protocol operation. The system manager initiates the state update procedure for performing warmstart and controlled switchover procedures.

The PSF module on the standby node updates the protocol state information received from the active node onto the standby ISUP (PSIF - ISUP) to ensure that the two nodes are synchronized during run time.

The system manager sends a sequence of control requests to the PSF and the ISUP (PSIF - ISUP) protocol layer of both active and standby nodes for carrying out either controlled or forced switchover and other fault tolerance activities. All such control requests supported by ISUP (PSIF - ISUP) and PSF are described in detail in the following sections.

Note: *The sequence of system manager control requests is beyond the scope of this document and is described in the Fault-Tolerant/High-Availability (FT/HA) Core Service Definition.*

1.1 Recommended Reading: FT/HA Service Definition

The *Fault-Tolerant/High-Availability (FT/HA) Core Service Definition* describes:

- System level fault-tolerant scenarios (for example, switchover and controlled switchover). Refer to this document for a complete understanding of the functioning of various fault-tolerant layers in a protocol stack residing on different nodes.
- Various system entities which together achieve fault tolerance functionality. It also describes the sequence of various events that perform fault-tolerant procedures between the protocol layers and other system entities, such as controlled switchover and forced switchover.

Within the *Fault-Tolerant/High-Availability (FT/HA) Core Service Definition*, the layer manager functions are split into three parts: system manager, system agent, and stack manager. Trillium provides both the system manager and the system agent components. If the user is not utilizing Trillium's system manager and system agent, all interfaces with the system manager and system agent in the *Fault-Tolerant/High-Availability (FT/HA) Core Service Definition* should be viewed as layer manager interfaces. You can implement the system manager, system agent, and stack manager in a proprietary fashion.

If the user is not using Trillium's router, the router functionality in the *Fault-Tolerant/High-Availability (FT/HA) Core Service Definition* should also be considered a layer manager functionality. It is up to the user to provide the router functionality, in whatever fashion, for the proper functioning of PSF - ISUP in a fault-tolerant environment.

1.2 Terms and Definitions

Term	Definition
Active node	A node that executes software to provide the necessary protocol functionality. The active node processes the protocol messages and updates the new state information in the standby node.
Controlled switchover	A procedure that makes a standby node active and an active node standby
Fault-tolerant node	A pair of nodes with replicated protocol layers. A fault-tolerant node can have an active, standby, or OOS state.
Forced switchover	A procedure that makes a standby node active when an active node goes OOS
Node	A unit that has a processor(s) with private volatile memory inaccessible to all other nodes, and a private clock governing the execution of instructions on this processor. A node also has a network interface connecting it to a communication network through communication channels. The software governs the sequence of instructions executed on a node.
Out-Of-Service (OOS) node	An off-line node that can become an active or standby node
Run-time state update	The active ISUP (PSIF - ISUP) handles protocol events, which can result in internal ISUP (PSIF - ISUP) state changes. The active PSF updates the standby with the state changes to keep the standby synchronized.
Standby node	A node that acts as a backup to an active node
Warmstart	A procedure that makes an OOS node standby. An active node updates this new standby node with current information using a bulk update procedure.

1.3 Abbreviations

The following abbreviations are used in this document:

Abbreviation	Definition
ANSI	American National Standards Institute
ETSI	European Telecommunications Standards Institute
FT / HA	Fault-Tolerant / High-Availability
ISDN	Integrated Services Digital Network
ISUP	ISDN User Part
ITU	International Telecommunications Union
LM	Layer Manager
MTP3	Message Transfer Part Level 3
OOS	Out-Of-Service
PSF	Protocol Specific Function
PSIF	Protocol Specific Interface Function
SAP	Service Access Point
SCCP	Signalling Connection Control Part
SS	System Services
TAPA	Trillium Advanced Portability Architecture

2 ENVIRONMENT

This section describes design assumptions about the operating environment for the PSF - ISUP software.

2.1 Trillium Advanced Portability Architecture (TAPA)

Trillium's ISUP product conforms to Trillium Advanced Portability Architecture (TAPA). TAPA can be visualized as a box surrounded by four outer boxes. The box in the center represents the ISUP software, while the four outer boxes represent other software to which ISUP can be connected. The separation between the center box and outer boxes defines the interfaces across which ISUP interacts with the other software.

For example, in Figure 2-1, the lower layer could be Trillium's MTP Level 3 and the upper layer could be Trillium's Interworking Call Control.

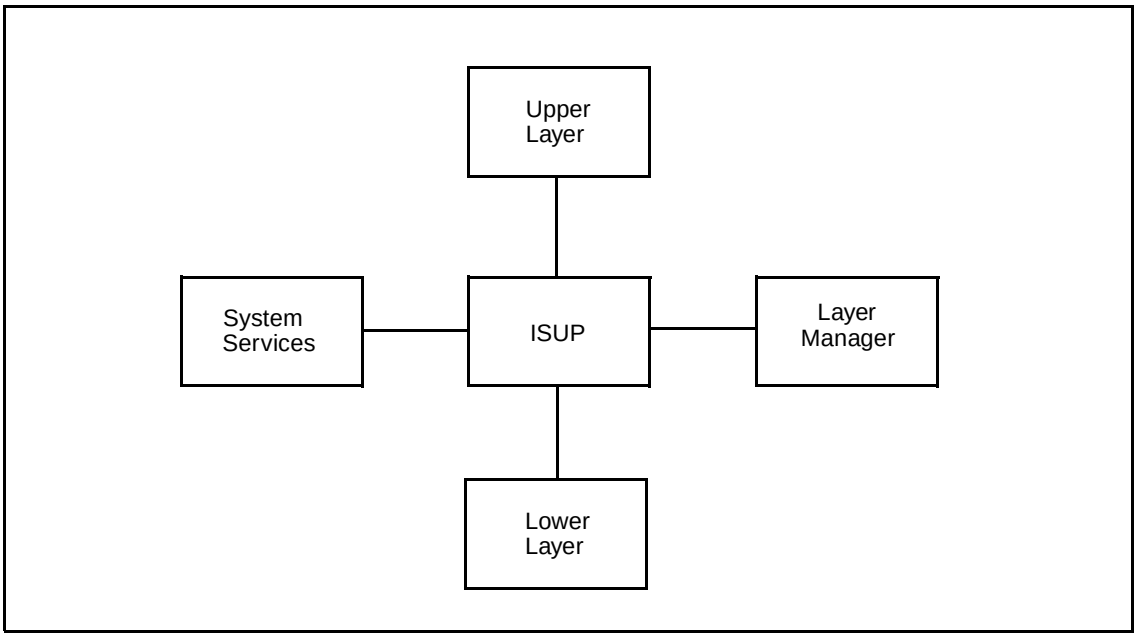


Figure 2-1: Trillium Advanced Portability Architecture (TAPA)

2.2 PSF - ISUP Architecture

The architecture of PSF - ISUP differs from the architecture of other Trillium products that conform to TAPA in the following ways:

- PSF - ISUP does not have an upper layer
- PSF - ISUP does not have a lower layer

However, PSF - ISUP does provide the standard Layer Manager (LM) and System Services (SS) interfaces.

Figure 2-2 illustrates the architecture of the PSF - ISUP software environment.

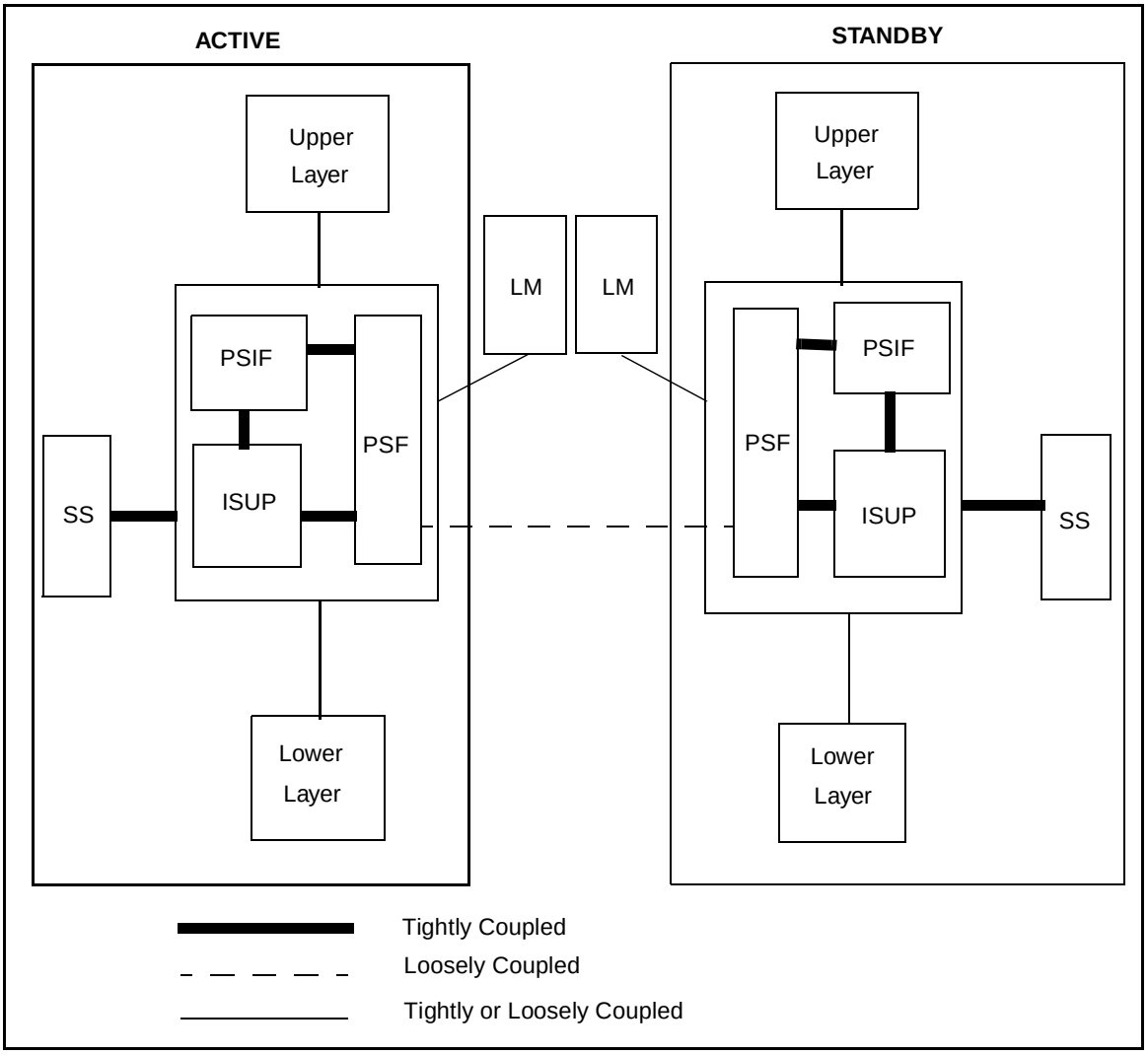


Figure 2-2: PSF - ISUP environment

The interfaces in Figure 2-2 are:

Interface	Description
System Services (SS)	This interface provides the system functions required by PSF - ISUP, including, initialization, timer management, memory management, message and queue management, date and time management, and resource checking. The System Services interface is always tightly coupled.
Layer Manager (LM)	This interface provides the functions required to control and monitor PSF - ISUP. In addition, this interface provides the functions to initialize and modify PSF - ISUP configuration parameters. This interface can be tightly or loosely coupled.
Peer PSF - ISUP	This interface provides functions for the active PSF - ISUP to communicate with the standby PSF - ISUP, and vice versa. These functions involve state updates from active to standby during run time, warmstart, and controlled switchover. This interface is always loosely coupled.
PSF - ISUP and ISUP (PSIF - ISUP)	PSF - ISUP updates the stable state changes during run-time state update. In order to avoid excessive overhead, all transient states are updated during warmstart and controlled switchover, but not during run time. This interface is always tightly coupled.

PSF - ISUP interacts with the layer manager and the peer PSF - ISUP using a set of primitive functions. These primitives, taking the form of requests, indications, responses, and confirms, completely define the interaction between the layers. Information flows between layers through the exchange of primitives across Service Access Points (SAPs).

3 INTERFACE PRIMITIVES

The following sections describe the primitives at the PSF - ISUP interfaces.

In the primitives, the sizes of the primitive data types are defined as:

Mnemonic	# of 8 bit bytes	Sign
S8	1	signed
U8	1	unsigned
S16	2	signed
U16	2	unsigned
S32	4	signed
U32	4	unsigned
PTR	As required	unsigned

Note: *The size of* **PTR** *depends on the specific machine to which the software is ported.*

3.1 System Services Interface

This section provides a brief description of the System Services Interface (SSI) functions that PSF - ISUP uses. PSF - ISUP uses a subset of the functions that system services provides. The *System Services Interface Service Definition* describes all the SSI functions in detail. The system services interface functions used by PSF - ISUP are described in the following section.

Task Scheduling

The following functions register, activate, and terminate a task:

Name	Description
SPstTsk	Post a message buffer to a destination task
SExitTsk	Clean up and exit a task

Timer Management

The following functions register and deregister the timer activation function(s). System services calls back the timer activation function(s):

Name	Description
SRegTmr	Register timer activation function
SDeregTmr	Deregister timer activation function
ziActvTmr	PSF - ISUP timer activation function. This function is provided in PSF - ISUP and is called back by system services.

Message Management

The following functions initialize, add, and remove data to and from messages. The structure of a message is not known to a layer and is specific to system services:

Name	Description
SGetMsg	Allocate a message buffer
SPutMsg	Deallocate a message buffer
SRepMsg	Replace a specified byte in a message with a new value
SCpyMsgMsg	Make a copy of a message into a newly allocated message buffer
SFndLenMsg	Find the length of a message
SPrntMsg	Print the contents of a message buffer
SAddPstMsg	Add a byte to the tail of a message
SRemPstMsg	Remove a byte from the tail of a message
SRemPreMsg	Remove a byte from the front of a message
SAddPstMsgMult	Add multiple bytes to the tail of a message
SRemPreMsgMult	Remove multiple bytes from the front of a message

Packing/Unpacking

The following functions pack and unpack the interface primitive parameters. These functions are required only if any of the PSF - ISUP interfaces are loosely coupled.

Name	Description
SPkU8	Add unsigned 8-bit value to the head of a message
SPkU16	Add unsigned 16-bit value to the head of a message
SPkU32	Add unsigned 32-bit value to the head of a message
SPkS8	Add signed 8-bit value to the head of a message
SPkS16	Add signed 16-bit value to the head of a message
SPkS32	Add signed 32-bit value to the head of a message
SUnpkU8	Remove unsigned 8-bit value from the head of a message
SUnpkU16	Remove unsigned 16-bit value from the head of a message
SUnpkU32	Remove unsigned 32-bit value from the head of a message
SUnpkS8	Remove signed 8-bit value from the head of a message
SUnpkS16	Remove signed 16-bit value from the head of a message
SUnpkS32	Remove signed 32-bit value from the head of a message

Queue Management

The following functions initialize, add, and remove messages to and from queues. The structure of a queue is not known to a layer and is system services-specific.

Name	Description
SInitQueue	Initialize a queue
SQueueLast	Add a message to the tail of a queue
SDequeueFirst	Remove a message from the head of a queue
SFlushQueue	Flush all messages in a queue
SFndLenQueue	Find the length of a queue
SExamQueue	Examine a message at a specified index in a queue

Miscellaneous

The following functions are used for date and time management, error handling, and resource availability checking:

Name	Description
SLogError	Handle fatal system error
SGetDateTime	Get current date and time
SPrint	Print a preformatted string
SPrntMsg	Print the contents of a message
SFndProcId	Find the processor ID on which a task is running

3.2 Layer Manager Interface

The layer manager interface is designated by **Lzi** and is described in detail in the following sections. The interface between the PSF - ISUP and the layer manager provides the following functions:

Configuration

The following functions configure PSF - ISUP:

Name	Description
ZiMiLziCfgReq	Configuration request
ZiMiLziCfgCfm	Configuration confirm

Control

The following functions control PSF - ISUP:

Name	Description
ZiMiLziCntrlReq	Control request
ZiMiLziCntrlCfm	Control confirm

Solicited Status

The following functions indicate the current state of the protocol layer:

Name	Description
ZiMiLziStaReq	Status request
ZiMiLziStaCfm	Status confirm

Unsolicited Status

The following function indicates a change in the status of the protocol layer:

Name	Description
ZiMiLziStaInd	Status indication

3.2.1 Common Structures

Each layer manager primitive uses the following structures in addition to the specific structures for the respective primitive. The common structures are described as follows:

3.2.1.1 Header

The header has the following structure format:

```
typedef struct tds_header       /* header */
{
   U16        msgLen;           /* message length - optional */
   U8         msgType;          /* message type - optional */
   U8         version;          /* version - optional */
   U16        seqNmb;           /* sequence number - optional */
   EntityId   entId;            /* entity id - optional*/
   ElmntId    elmId;            /* element id - mandatory */
#ifdef LMINT3
   TranId     transId;          /* transaction id - mandatory */
   Resp       response;         /* response parameters - mandatory */
#endif /* LMINT3 */
} Header;
```

msgLen

Message length. Not used.

msgType

Message Type. Allowable values:

```
TCFG   (configuration)
TCNTRL (control)
TSTA   (solicited status)
TUSTA  (unsolicited status)
```

version

Version of the software. Not used.

seqNmb

Sequence number of the layer manager primitive. Not used by PSF - ISUP module, but could be used by the system manager or the stack manager.

entId

Structure entity. Structure entity has the following format:

```
typedef struct entityId
{
  Ent  ent;
  Inst inst;
}EntityId;
```

ent

Entity name.

inst

Entity instance.

Note: **entId** *is not used by PSF - ISUP.*

elmId

Structure element. Structure element has the following format:

```
typedef struct elmntId
{
   Elmnt      elmnt;
   ElmntInst1 elmntInst1;
   ElmntInst2 elmntInst2;
   ElmntInst3 elmntInst3;
}ElmntId;
```

elmnt

Element. Allowable values:

STGEN
STSID
STPEERSAP

elmntInst1

Element instance. Not used.

elmntInst2

Element instance. Not used.

elmntInst3

Element instance. Not used.

transId

Transaction ID.

response

Response structure. It has the following format:

```
typedef struct resp
{
   Selector  selector;
   Route     route;
   Priority  prior;
   MemoryId  mem;
}Resp;
```

selector

Response **pst** structure selector field.

route

Response **pst** structure route field.

prior

Response **pst** structure prior field.

mem

Response **pst** structure region and pool.

3.2.1.2 Date and Time

The date and time structure is used in the layer manager primitives to report the date and time when a primitive is called. **dateTime** has the following structure:

```
typedef struct dateTime
{
   U8 month;
   U8 day;
   U8 year;
   U8 hour;
   U8 min;
   U8 sec;
   U8 tenths;
} DateTime;
```

month

The current month.

day

The current day.

year

The current year.

hour

The current hour.

min

The current minute.

sec

The current second.

tenths

Tenths of a second.

3.2.1.3 Post

Destination post structure. Used by system services for intertask communication. It is used in every layer manager primitive to identity of the destination layer and route the message buffer to the destination layer. The post structure has the following format:

```
typedef struct pst
{
   ProcId dstProcId;              /* destination processor id */
   ProcId srcProcId;              /* source processor id      */

   Ent dstEnt;                    /* destination entity       */
   Inst dstInst;                  /* destination instance     */
   Ent srcEnt;                    /* source entity            */
   Inst srcInst;                  /* source instance          */

   Prior prior;                   /* priority                 */
   Route route;                   /* route                    */
   Event event;                   /* event                    */
   Region region;                 /* region                   */

   Pool pool;                     /* pool                     */
   Selector selector;             /* selector                 */
   U16 spare1                     /* spare 1                  */
} Pst;
```

dstProcId

Destination processor's **procId**.

srcProcId

Source processor's **procId**.

dstEnt

Destination entity ID.

dstInst

Instance ID of the destination entity.

srcEnt

Source entity ID.

srcInst

Instance ID of the source entity.

prior

Priority of the message.

route

Route field to be used in the message.

event

Event specifying the type of primitive in the message.

region

Region of the memory from where the buffer should be allocated.

pool

Region of the memory from where the buffer should be allocated.

selector

Selector for resolving tight coupling or loose coupling with the destination entity.

spare1

Spare for alignment. Not used.

3.2.1.4 Common Status

This structure is used in the primitives going from the PSF - ISUP to the layer manager. Common status structure has the following format:

```
typedef struct cmStatus
{
   U16 status;                /* status of request */
   U16 reason;                /* failure reason */
}CmStatus;
```

status

Status to indicate the success or failure of a layer manager request coming to PSF - ISUP. Allowable values:

```
LCM_PRIM_OK                   /* success */
LCM_PRIM_NOK                  /* failure */
LCM_PRIM_OK_NDONE             /* activity in progress */
```

reason

Reason for the failure of the layer manager request. If the status is **LCM_PRIM_NOK**, this field specifies the reason for the request's failure.

```
LCM_REASON_REGTMR_FAIL        /* Timer registeration failed */
LCM_REASON_INV_PAR_VAL        /* Upd msg size parameter invalid */
LCM_REASON_INVALID_ELMNT      /* Invalid element in config req */
LCM_REASON_NOT_APPL           /* Reason not applicable:used with
                                 LCM_PRIM_OK status */
LCM_REASON_PRTLYRCFG_NOT_DONE /* Portable layer configuration not done */
```

3.2.2 Primitive Descriptions

The following sections give detailed descriptions of the primitives at the layer manager interface.

3.2.2.1 ZiMiLziCfgReq

Name:

Configuration Request

Direction:

Layer manager to PSF - ISUP

Supplied:

Yes

Synopsis:

```
PUBLIC S16 ZiMiLziCfgReq(pst, cfg)
Pst *pst;
CmPFthaMngmt *cfg;
```

Parameters:

pst

Post structure. See Section 3.2.1.3, "Post."

cfg

```
typedef struct cmPFthaMngmt
{
   Header   hdr;
   CmStatus cfm;
   union
   {
   .
   .
   .
      struct
      {
         union
         {
            CmPFthaGenCfg   genCfg;
            CmPFthaSAPCfg   peerSAPCfg;
         }s;
      }cfg;
   .
   .
   .
   }t;
 }CmPFthaMngmt;
```

hdr

Header structure. See Section 3.2.1.1, "Header." The layer manager initializes the following fields:

```
hdr.msgType             /* TCFG */
hdr.entId               /* entity id of PSF-ISUP */
hdr.inst                /* instance of PSF-ISUP */
hdr.elmnt.elmnt         /* STGEN or STPEERSAP */
hdr.transId             /* transaction Id for confirm */
hdr.response.selector   /* selector to be used for confirm */
hdr.response.route      /* route to be used for confirm */
hdr.response.prior      /* priority to be used for confirm */
hdr.response.mem.pool   /* memeory pool for confirm */
hdr.response.mem.region /* region for confirm */
```

cfm

Common status. See Section 3.2.1.4, "Common Status."

genCfg

General configuration structure.

```
typedef struct cmPFthaGenCfg
{
   S16 timeRes;                 /* timer resolution */
   ProcId vProcId;              /* virtual processor Id */
   MemoryId mem;                /* self region and pool */
   U8 updateOption              /* update option */
   Pst smPst;                   /* stack manager post structure */
} CmPFthaGenCfg;
```

timeRes

Timer resolution in ticks. System services invokes the PSF - ISUP timer function, **ziPrcPeerSapTq**, in every **timeRes**.

Allowable values: Up to 32767.

This field is not reconfigurable.

vProcId

Virtual processor ID of the node on which ISUP (PSIF - ISUP) and PSF - ISUP reside. The upper and lower layers use the **vProcId** to send messages to ISUP (PSIF - ISUP) for loosely coupled interfaces. An external entity is responsible for converting **vProcId** to the physical processor ID of the currently active ISUP (PSIF - ISUP) node before delivering the message to ISUP (PSIF - ISUP). PSF - ISUP initializes the source processor ID of all the lower SAPs and the upper SAPs of ISUP (PSIF - ISUP) with the **vProcId**.

Allowable values: 0 to 65535.

This field is not reconfigurable.

mem

Region and pool to be used by PSF - ISUP to mail a message to itself during warmstart and controlled switchover.

Allowable values: 0 to 255.

This field is not reconfigurable.

updateOption

Configuration parameter to determine if state updating is to be done for unanswered calls. Allowable values:

0 - Do not update

1 - Update

This field is reconfigurable.

smPst

Post structure for sending alarms to the layer manager. This field is reconfigurable.

CmPFthaSAPCfg

Peer SAP structure.

```
typedef struct cmPFthaSAPCfg
{
   Region region;              /* memory region for peer */
   Pool pool;                  /* memory pool for peer */
   ProcId dstProcId;           /* peer processor id */
   Ent dstEnt;                 /* peer entity id */
   Inst dstInst;               /* peer instance id */
   Priority prior;             /* peer post priority */
   Route route;                /* peer post route */
   Selector selector;          /* peer selector */
   TmrCfg tUpdCompAck;         /* update completion timer */
   U32 maxUpdMsgSize;          /* maximum size of the update message */
} CmPFthaSAPCfg;
```

region

Memory region for allocating message buffers for mailing a message to the peer PSF - ISUP. This field is reconfigurable.

Allowable values: 0 to 255.

pool

Memory pool for allocating message buffers for mailing a message to the peer PSF - ISUP. This field is reconfigurable.

Allowable values: 0 to 255.

dstProcId

Peer to PSF - ISUP' physical processor ID. This field is reconfigurable.

Allowable values: 0 to 65535.

dstEnt

Peer PSF - ISUP's entity ID. This field is reconfigurable.

Allowable values: 0 to 255.

dstInst

Peer PSF - ISUP's instance ID. This field is reconfigurable.

Allowable values: 0 to 255.

prior

Priority for post structure of the peer SAP. This field is user defined.

route

Route for the post structure of the peer SAP. This field is reconfigurable.

Allowable value: Default (**RTESPEC**).

selector

Selector for the post structure of the peer SAP. This field is reconfigurable. Not currently used.

tUpdCompAck

Update completion acknowledgment timer. Value for the timer started by the active PSF - ISUP to wait for a confirm from the standby for warmstart or controlled switchover state update. **tUpdCompAck** has the following structure:

```
typedef struct tmrCfg       /* timer configuration */
{
   Bool  enb;               /* flag to enable/disable timer */
   U16   val;               /* value of the timer */
}TmrCfg;
```

Allowable values: 0 to 65535.

This field is reconfigurable.

maxUpdMsgSize

Maximum message update size. This field is reconfigurable.

Allowable values: X to (2^{32} - 1).

Note: *The minimum size of the update message must be greater than the maximum size of a table to be packed by PSF - ISUP. The minimum size required by PSF - ISUP is governed by the maximum size of any control block to be packed, plus the size of* **CmPFthaMsgHdr,** *plus 1 (for packing table type, which occupies 1 byte). The layer manager fills this* **maxUpdMsgSize** *field greater than this minimum value.*

Description:

The stack manager uses this primitive to send the general and peer SAP configuration to the PSF - ISUP module. For more information, see Section 4.2.1, "Management–Configuration." The stack manager configures the timer resolution and the timer values for warmstart and controlled switchover procedures. PSF - ISUP calls **`SRegTmr`** with the configured timer resolution to register the timer function with the system services.

Returns:

`ROK`	OK
`RFAILED`	Failed

3.2.2.2 ZiMiLziCfgCfm

Name:

Configuration Confirm

Direction:

PSF - ISUP to layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 ZiMiLziCfgCfm(pst, cfm)
Pst *pst;
CmPFthaMngmt *cfm;
```

Parameters:

pst

Pointer to post structure. See Section 3.2.1.3, "Post."

cfm

Common management structure. Only **hdr** and **cfm** are used by this primitive.

```
typedef struct cmPFthaMngmt
{
   Header hdr;
   CmStatus cfm;
   union
   {
      ...
      ...
   }t;
 }CmPFthaMngmt;
```

hdr

Header structure. See Section 3.2.1.1, "Header."

cfm

Common status structure. See Section 3.2.1.4, "Common Status." Allowable values for its fields are:

status

Status to indicate success or failure of a layer manager request coming to the PSF - ISUP. Allowable values:

```
LCM_PRIM_OK           /* configuration request is successful */
LCM_PRIM_NOK          /* configuration request is not successful */
```

reason

Reason for the failure. Allowable values:

```
LCM_REASON_REGTMR_FAIL        /* Timer registeration failed */
LCM_REASON_GENCFG_NOT_DONE    /* General configuration not done */
LCM_REASON_INV_PAR_VAL        /* Upd msg size parameter invalid */
LCM_REASON_QINIT_FAIL         /* Queue init failed */
LCM_REASON_INVALID_ELMNT      /* Invalid element in config req */
LCM_REASON_NOT_APPL           /* Reason not applicable: used
                                 with LCM_PRIM_OK status */
LCM_REASON_PRTLYRCFG_NOT_DONE /* Portable layer configuration
                                 not done */
```

Description:

PSF - ISUP uses this primitive to acknowledge the configuration request primitive (**ZiMiLziCfgReq)** sent by the stack manager. PSF - ISUP indicates the success or failure of the configuration request received from the layer manager. For more details on the configuration procedure, see Section 4.2.1, "Management–Configuration."

Returns:

ROK OK

RFAILED Failed

3.2.2.3 ZiMiLziStaReq

Name:

Status Request

Direction:

Layer manager to PSF - ISUP

Synopsis:

```
PUBLIC S16 ZiMiLziStaReq(pst, sta)
Pst          *pst;
CmPFthaMngmt *sta;
```

Parameters:

pst

Pointer to post structure. See Section 3.2.1.3, "Post."

sta

Pointer to management structure. Status structure has the following format:

```
typedef struct cmPFthaMngmt
{
   Header   hdr;      /* header */
   CmStatus cfm;      /* confirm */
   union              /* values are returned in sta structure*/
   {
    .
    .
    .
   } t;
 } CmPFthaMngmt;
```

hdr

Header structure. See Section 3.2.1.1, "Header."

Description:

The stack manager uses this primitive to get the status of the protocol layer and the status of the peer SAP from PSF - ISUP. For more information see Section 4.2.2, "Management–Solicited Status."

Returns:

00 OK

3.2.2.4 ZiMiLziStaCfm

Name:

Status Confirm

Direction:

PSF - ISUP to layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 ZiMiLziStaCfm(pst, sta)
Pst *pst;
CmPFthaMngmt *sta;
```

Parameters:

pst

Pointer to post structure. Section 3.2.1.3, "Post." Confirmation is sent to the sender of **ZiMiLziStaReq**.

sta

Pointer to management structure. Status structure has the following format:

```
typedef struct cmPFthaMngmt
{
   Header    hdr;
   CmStatus  cfm;
   union
   {
   .
   .
   .
      struct
      {
         DateTime dt;
         union
         {
            U8 genSta;                      /* general status */
            CmPFthaPeerSapSta peerSapSta;  /* peer sap status */
            SystemId sysId;                 /* system id */
          }s;
      }sta;
   .
   .
   .
   }t;
}CmPFthaMngmt;
```

hdr

Header structure. See Section 3.2.1.1, "Header." **transId** field should be the same as that received in **ZiMiLzi StaReq**.

cfm

Common status structure. See Section 3.2.1.4, "Common Status." Allowable values for its fields are:

status

```
LCM_PRIM_OK              /* status request successful */
LCM_PRIM_NOK             /* status request not successful */
```

reason

If the status is **LCM_PRIM_NOK**, this field specifies the reason for the failure of the request.

```
LCM_REASON_INVALID_ELMNT  /* Invalid element in config req */
LCM_REASON_NOT_APPL       /* Reason not applicable: used with
                             LCM_PRIM_OK status */
```

dt

Date and time structure. See Section 3.2.1.2, "Date and Time."

genSta

Status of the ISUP (PSIF - ISUP) protocol layer. Allowable values:

```
00   ACTIVE
01   STANDBY
02   OOS
```

peerSapSta

Specifies the status of the peer SAP. It has the following format:

```
typedef struct cmPFthaPeerSapSta
{
   U8 bndState;             /* bind state */
   U8 updState;             /* update state */
}CmPFthaPeerSapSta;
```

bndState

Bind status of the peer SAP. Allowable values:

```
CMPFTHA_BND              /* SAP is bound */
CMPFTHA_UBND             /* SAP is not bound */
```

updState

Update status of the peer SAP. Allowable values:

```
CMPFTHA_IDLE             /* No activity at peer SAP */
CMPFTHA_WRMSTRT          /* Peer PSF under warmstart */
CMPFTHA_SYNC             /* Peer PSF under synchronization */
CMPFTHA_WRMSTRT_WAIT     /* waiting for ack from peer PSF for
                            warmstart completion */
CMPFTHA_SYNC_WAIT        /* waiting for ack from peer PSF for
                            synchronization completion */
```

sysId

Specifies the current version of the PSF - ISUP software. It has the following format:

```
typedef struct systemId
{
   S16 mVer;
   S16 mRev;
   S16 bVer;
   S16 bRev;
   S8 *ptNmb;
}SystemId;
```

mVer

Main version.

mRev

Main revision.

bVer

Branch version.

bRev

Branch revision.

ptNmb

Pointer to the part number.

Description:

PSF - ISUP uses this primitive to return the status of the protocol layer, the status of the peer SAP, or the version of the current PSF - ISUP software to the layer manager. If the status request is successful, then the common status structure is **LCM_PRIM_OK** in **ZiMiLziStaCfm**; otherwise, it is **LCM_PRIM_NOK**. For more information, see Section 4.2.2, "Management–Solicited Status."

Returns:

ROK OK

RFAILED Failed

3.2.2.5 ZiMiLziStaInd

Name:

Status Indication

Direction:

PSF - ISUP to layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 ZiMiLziStaInd(pst, usta)
Pst *pst;
CmPFthaMngmt *usta;
```

Parameters:

pst

Pointer to post structure. See Section 3.2.1.3, "Post."

usta

Pointer to unsolicited status indication structure. It has the following format:

```
typedef struct cmPFthaMngmt
{
   Header hdr;
   CmStatus cfm;
   union
   {
   .
   .
   .
     struct
     {
        CmAlarm alarm;                        /* alarm structure */
        U8  evntParm[CMPFTHA_USTA_EP_MAX];    /* event parameters */
     }usta;
   .
   .
   .
   }t;
}CmPFthaMngmt;
```

hdr

Header structure. See Section 3.2.1.1, "Header."

cfm

Common status structure. Not used in this primitive.

alarm

Common alarm structure. It has the following format:

```
typedef struct cmAlarm
{
   DateTime dt;      /* data and time */
   U16 category;     /* alarm category*/
   U16 event;        /* alarm event */
   U16 cause;        /* alarm cause */
}CmAlarm;
```

dt

Date and time structure. See Section 3.2.1.2, "Date and Time."

category

Category of alarm generated by the PSF - ISUP. Allowable values:

Allowable Values	Description
LCM_CATEGORY_RESOURCE	System resources category. An alarm in this category is generated if an error occurs during allocation / deallocation / addition / subtraction of system resources.
LCM_CATEGORY_INTERFACE	Interface category. This category indicates the occurrence of an error during the handing of interface events. An alarm in this category is generated if any interface parameter is in error or the interface event is not expected in the current state of the protocol.
LCM_CATEGORY_PSF_FTHA	PSF category. An alarm in this category is generated if an error occurs during normal PSF operations.

evnt

Type of alarm generated by PSF - ISUP. Allowable values:

Allowable Values	Description
LCM_EVENT_INV_TMR_EVT	Generated by the standby PSF when a protocol timer expires on the standby node. The standby should not run any timer.
CMPFTHA_SEQERR	Reported by the standby PSF when it detects a run-time sequence error. The stack manager should make the standby OOS on getting this event.
CMPFTHA_MEM_FAILURE	Reported by active or standby PSF whenever it encounters memory allocation failures
CMPFTHA_UPDMSG_ERR	Reported by the standby PSF to indicate that a wrong update message has been received from active PSF

cause

This field specifies the cause of alarm generated by the PSF. Allowable values:

```
LCM_CAUSE_PROT_NOT_ACTIVE     /* protocol not active */
LCM_CAUSE_UNKNOWN             /* unknown */
```

Possible combinations of **category**, **cause**, and **event** of each alarm are shown in the table below:

Category	Event	Cause
LCM_CATEGORY_RESOURCE	CMPFTHA_MEM_FAILURE	CAUSE_UNKNOWN
LCM_CATEGORY_PSF_FTHA	CMPFTHA_SEQERR	CAUSE_UNKNOWN
LCM_CATEGORY_PSF_FTHA	CMPFTHA_UPDMSG_ERR	CAUSE_UNKNOWN
LCM_CATEGORY_INTERFACE	LCM_EVT_INV_TMR_EVT	LCM_CAUSE_PROT_NOT_ACTIVE

evntParm

Event parameters. This field provides more details about the alarm. This field is not currently used.

Description:

PSF - ISUP uses this primitive to report alarms to the stack manager. For example, when the standby PSF - ISUP detects sequence error in the run-time state update messages, it sends **CMPFTHA_SEQERR** to the stack manager. For more information, see Section 4.2.3, "Management–Unsolicited Status."

Returns:

ROK OK

RFAILED Failed

3.2.2.6 ZiMiLziCntrlReq

Name:

Control Request

Direction:

System manager to PSF - ISUP

Supplied by:

Yes

Synopsis:

```
PUBLIC S16 ZiMiLziCntrlReq(pst, cntrl)
Pst *pst;
CmPFthaMngmt *cntrl;
```

Parameters:

pst

Pointer to post structure. See Section 3.2.1.3, "Post."

cntrl

Pointer to control structure. It has the following format:

```
typedef struct cmPFthaMngmt
{
   Header hdr;
   CmStatus cfm;
   union
   {
   .
   .
   .
      struct
      {
         DateTime dt;               /* date & time */
         U8 action;                 /* action */
         U8 subAction;              /* subaction */
         union
         {
            CmPFthaDbgCntrl umDbg;  /* debug control */
         }ctlType;
      }cntrl;
   .
   .
   .
   }t;
}CmPFthaMngmt;
```

hdr

Header structure. See Section 3.2.1.1, "Header."

cfm

Common status structure. Not used in this primitive.

dt

Date and time structure. See Section 3.2.1.2, "Date and Time."

action

Control action.

subAction

Subaction. The following table lists the allowable values for the **action**, **subaction**, and **elmnt** (**hdr.elmId.elmnt**) parameters. The destination column shows the allowable combination values for the three parameters.

Note: *Combinations not listed in the following table are not allowed by PSF - ISUP.*

Action	Subaction	elmnt	Destination	Description
AENA	**SAUSTA**	**STGEN**	Active, Standby	Enable alarms
	SADBG	**STGEN**	Active, Standby	Enable debug prints for the specified type (see **Note 1** below table)
ADISIMM	**SAUSTA**	**STGEN**	Active, Standby	Disable alarms
	SADBG	**STGEN**	Active, Standby	Disable debug prints for the specified type (see **Note 1** below table)
AUBND_DIS		**STPEERSAP**	Active	Disable the peer SAP
AGO_ACT	**SAENA_PEER_SAP**	**STGEN**	Standby	Make the protocol layer active and enable the peer SAP
	SADIS_PEER_SAP		Standby, OOS	Make the protocol layer active and disable the peer SAP
AGO_SBY	**SAENA_PEER_SAP**	**STGEN**	Active, OOS	Make the protocol layer standby and enable the peer SAP

Action	Subaction	elmnt	Destination	Description
ASYNCHRONIZE		**STGEN**	Active	Synchronize the peer to perform controlled switchover
AWARMSTART		**STGEN**	Active	Warmstart the peer to make it standby from OOS
AABORT		**STGEN**	Active	Abort the ongoing warmstart or synchronization procedure
ASHUTDOWN		**STGEN**	Active, Standby	Shutdown operations. PSF - ISUP deallocates all the allocated resources

Note 1: *Multiple debug levels can be specified in a single control request. For example, (***DBGMASK_MI | DBGMASK_PI***) is a valid* **dbgMask** *value.*

Note 2: *When the stack manager sends a control request to the active PSF - ISUP to warmstart or synchronize the peer layer, the active responds with two confirmations. The first confirmation indicates that the control request is accepted but not yet completed, and the second indicates the final result in terms of success or failure. The stack manager can send an abort control request before getting the final response.*

umDbg

PSF - ISUP debug structure.

Common debug control structure for PSF. It has the following format:

```
typedef struct cmPFthaDbgCntrl
{
   U32 dbgMask;   /* debug mask */
}CmPFthaDbgCntrl;
```

dbgMask

Debug mask. The default mask is **0**—that is, all the debug prints are disabled. The prints for the following debug levels can be enabled:

```
0                    /* all debug prints are disabled */
DBGMASK_MI           /* layer management interface */
DBGMASK_PI           /* peer interface */
DBGMASK_PLI          /* PSF protocol layer interface */
LZI_DBGMASK_PACK     /* debug mask for packing functions */
LZI_DBGMASK_UNPACK   /* debug mask for unpacking functions */
LZI_DBGMASK_WARN     /* Misc warnings in ISUP-PSF*/
```

Description:

The system manager sends control requests to PSF - ISUP for performing the fault tolerance procedures described in Section 4.2.4, "Management–Control." PSF - ISUP sends confirmation for the control request via the **`ZiMiLziCntrlCfm`** primitive.

Returns:

`ROK` OK

`RFAILED` Failed

3.2.2.7 ZiMiLzi CntrlCfm

Name:

Control Confirm

Direction:

PSF - ISUP to system manager

Supplied:

No

Synopsis:

```
PUBLIC S16 ZiMiLziCntrlCfm(pst, cfm)
Pst *pst;
CmPFthaMngmt *cfm;
```

Parameters:

pst

Pointer to post structure. See Section 3.2.1.3, "Post."

cfm

Common management structure. Only **hdr** and **cfm** are used by this primitive.

```
typedef struct cmPFthaMngmt
{
   Header hdr;
   CmStatus cfm;
   union
   {
      ...
      ...
   }t;
 }CmPFthaMngmt;
```

hdr

Header structure. See Section 3.2.1.1, "Header."

cfm

Common status structure. See Section 3.2.1.4, "Common Status." Allowable values for its fields are:

status

Status to indicate the success or failure of a layer manager request coming to the PSF - ISUP. Allowable values:

```
LCM_PRIM_OK           /* control request is accepted */
LCM_PRIM_NOK          /* control request is not accepted */
LCM_PRIM_OK_NDONE     /* control request is accepted but the
                         activity is not complete */
```

Note: **LCM_PRIM_OK_NDONE** *is returned only in response to warmstart or synchronization control requests. After this, the stack manager should wait for the final response:* **LCM_PRIM_OK** *or* **LCM_PRIM_NOK.**

reason

Reason of the failure. Allowable values:

```
LCM_REASON_INVALID_SUBACTION  /* Invalid subaction */
LCM_REASON_INVALID_ACTION     /* Invalid action */
LCM_REASON_PEER_SAP_NOT_CFG   /* Peer SAP is not configured */
LCM_REASON_INVALID_STATE      /* The PSF state is invalid */
LCM_REASON_INVALID_ELMNT      /* Element in the control request
                                 is invalid
```

Description:

PSF - ISUP uses this primitive to acknowledge the control request sent by the system manager.

Returns:

ROK OK

RFAILED Failed

3.3 Peer Layer Interface

The peer PSF - ISUP interface is used to transfer state update information from the active PSF - ISUP to the standby PSF - ISUP. This interface is always loosely coupled.

The primitives between peer PSF - ISUPs are characterized by the prefix **ZiPiOub** (PSF - ISUP peer interface outbound) or **ZiPiInb** (PSF - ISUP peer interface inbound). The following primitives are used for data transfer:

Name	Description
ZiPiOubDatReq	Outbound data request
ZiPiInbDatReq	Inbound data request
ZiPiOubDatCfm	Outbound data confirm
ZiPiInbDatCfm	Inbound data confirm

The primitives are described in detail in the following sections.

3.3.1 ZiPiOubDatReq

Name:

Outbound Data Request

Direction:

Active PSF - ISUP to standby PSF - ISUP

Supplied:

Yes

Synopsis:

```
PUBLIC S16 ZiPiOubDatReq(pst, mBuf)
Pst *pst;
Buffer *mBuf;
```

Parameters:

pst

Pointer to post structure. See Section 3.2.1.3, "Post."

mBuf

Pointer to the PSF - ISUP update message. Each ISUP (PSIF - ISUP) update message contains state information of one or more control blocks. The PSF - ISUP on the active node copies the state information into this message as part of warmstart, controlled switchover, or run-time state update procedures.

Description:

This primitive is used by PSF - ISUP on the active node to transfer state information to the standby. The active PSF - ISUP copies the state information for various control blocks (for example, upper SAP control block or lower SAP control block) in one update message. The active PSF - ISUP can copy a fixed maximum number of bytes into the message; therefore, the active PSF - ISUP can send multiple update messages to send the complete state information. PSF - ISUP posts this message to system services (outbound). System services then routes the message to the PSF - ISUP on the standby node. For more information, see the following sections:

- Section 4.3.1, "Run-time State Update"
- Section 4.3.2, "Warmstart State Update"
- Section 4.3.3, "Controlled Switchover State Update"

Returns:

00 ROK

3.3.2 ZiPiInbDatReq

Name:

Inbound Data Request

Direction:

Active PSF - ISUP to standby PSF - ISUP

Supplied:

Yes

Synopsis:

```
PUBLIC S16 ZiPiInbDatReq(pst, mBuf)
Pst *pst;
Buffer *mBuf;
```

Parameters:

pst

Pointer to post structure. See Section 3.2.1.3, "Post."

mBuf

Pointer to the PSF - ISUP update message. Each PSF - ISUP update message contains state information of one or more control blocks. The active PSF - ISUP copies the state information into this message as part of warmstart, controlled switchover, or run-time state update procedures.

Description:

This primitive is used by the standby PSF - ISUP to receive the state information of ISUP (PSIF - ISUP) control blocks from the active PSF - ISUP. The direction of this primitive is from the operating system to the standby PSF - ISUP (inbound). For more information, see the following sections:

- Section 4.3.1, "Run-time State Update"
- Section 4.3.2, "Warmstart State Update"
- Section 4.3.3, "Controlled Switchover State Update"

Returns:

00 ROK

3.3.3 ZiPiOubDatCfm

Name:

Outbound Data Confirm

Direction:

Standby PSF - ISUP to active PSF - ISUP

Supplied:

Yes

Synopsis:

```
PUBLIC S16 ZiPiOubDatCfm(pst, status)
Pst *pst;
U8 status;
```

Parameters:

pst

Pointer to post structure. See Section 3.2.1.3, "Post."

status

Status. Allowable values:

CMPFTHA_OK
CMPFTHA_NOK

Description:

This primitive is used by the standby PSF - ISUP to acknowledge the receipt of all the update messages (data requests) sent by the active PSF - ISUP as part of warmstart or controlled switchover. The standby PSF - ISUP sends only one data confirm in response to all the data requests received during warmstart or controlled switchover.

The active PSF - ISUP places a sequence number in all the data requests. The standby PSF - ISUP keeps track of each sequence number received. If the standby PSF - ISUP finds a data request that has an out-of-order sequence number, it immediately sends a data confirm with a **CMPFTHA_NOK** status to the active PSF - ISUP. When the standby PSF - ISUP detects the last data request (for warmstart or controlled switchover procedure), it sends data confirm with a **CMPFTHA_OK** status to the active PSF - ISUP.

The standby PSF - ISUP posts this message to outbound system services. System services then routes the message to PSF - ISUP on the active processor.

For run-time state update messages, the standby PSF - ISUP does not send any confirmation. For more information, see the following sections:

- Section 4.2.4.6, "Warmstart"
- Section 4.2.4.7, "Synchronize"

Returns:

00 ROK

3.3.4 ZiPiInbDatCfm

Name:

Inbound Data Confirm

Direction:

Standby PSF - ISUP to active PSF - ISUP

Supplied:

Yes

Synopsis:

```
PUBLIC S16 ZiPiInbDatCfm(pst, status)
Pst *pst;
U8 status;
```

Parameters:

pst

Pointer to post structure. See Section 3.2.1.3, "Post."

status

Status. Allowable values:

CMPFTHA_OK
CMPFTHA_NOK

Description:

This primitive is used by the active PSF - ISUP to receive the confirmation from the standby PSF - ISUP for the data requests sent during the warmstart or controlled switchover procedure. The direction of this primitive is from the operating system to the active, inbound PSF - ISUP.

For run-time state update messages, the standby PSF - ISUP does not send any confirmation. For more information, see the following sections:

- Section 4.3.2, "Warmstart State Update"
- Section 4.3.3, "Controlled Switchover State Update"

Returns:

00 ROK

4 INTERFACE PROCEDURES

The interface procedures define the mechanisms by which the PSF - ISUP software interacts, via primitives, with any adjacent software within the system that it resides. The procedures in this section explain only the PSF - ISUP behavior in various fault tolerant scenarios. Refer to the *Fault-Tolerant/High Availability (FT/HA) Core Service Definition* document for details about the fault-tolerant scenarios and for the time sequencing of primitives among other entities apart from PSF - ISUP.

Numerous flow diagrams appear in this section. For each flow diagram, the following rules apply:

- Time flows towards the bottom of the page
- The mnemonic above a line represents a function call or PSF - ISUP primitive
- The **+** indicates an OR condition (one path or another can be taken)
- The **o** indicates an AND condition (both paths are taken in parallel)

The labels above each flow diagram have the following meaning:

Name	Description
LM	Layer Manager
SS	System Services
SI	ISUP
PSF	Protocol Specific Function
PSIF	Protocol Specific Interface Function
Peer PSF	Peer Protocol Specific Function

The following interface procedures are described:

Interface	Procedure description
System services interface	Procedures related to timer activations
Layer manager interface	Procedures related to the control and monitoring of PSF - ISUP software (for example, configuration, solicited status, unsolicited status, control)
Peer layer interface	Procedures related to run-time, warmstart, and controlled switchover state update

4.1 System Services Interface

PSF - ISUP is not registered as a separate task with system services. PSF - ISUP has the same entity ID as ISUP (PSIF - ISUP) as they are tightly coupled. ISUP calls both the activation initialization function and the task activation function of the PSF - ISUP, along with PSIF - ISUP.

4.1.1 Task Initialization

System services calls ISUP's initialization activation function. As a result, ISUP calls PSF - ISUP's initialization activation function to initialize PSF - ISUP's global variables and structures. PSIF - ISUP layer is also initialized by ISUP, similar to PSF - ISUP initialization.

The data flow is:

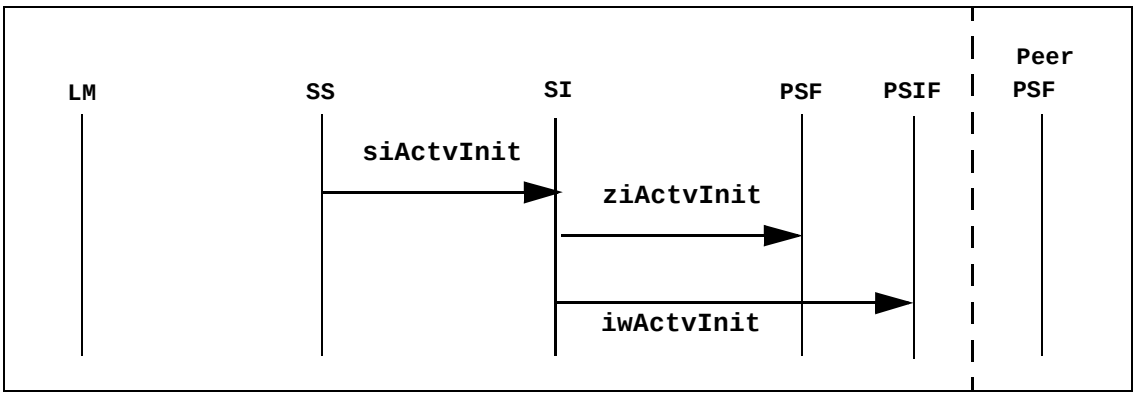


Figure 4-1: Data flow—task initialization procedure

4.1.2 Task Activation

System services calls ISUP's task activation function to deliver a message. Consequently, ISUP calls PSF - ISUP's task activation function if the received message is not for ISUP.

The data flow is:

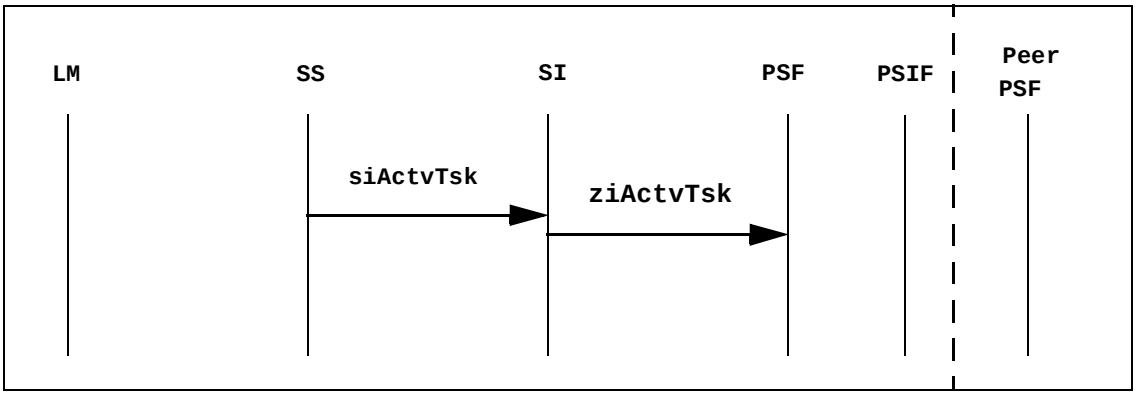


Figure 4-2: Data flow—task activation procedure

4.1.3 Timer Activation

PSF - ISUP registers the timer activation function (**ziActvTmr**) with system services using the **SRegTmr** function call. PSF-ISUP also specifies the timer activation periodicity (timer resolution) as a number of system ticks while registering the timer activation function. Consequently, system services recalls PSF - ISUP's timer activation function after every timer resolution, so that PSF - ISUP can manage its own timers.

PSF - ISUP calls **SRegTmr** during the general configuration after PSF - ISUP has received the timer resolution from the layer manager.

The data flow is:

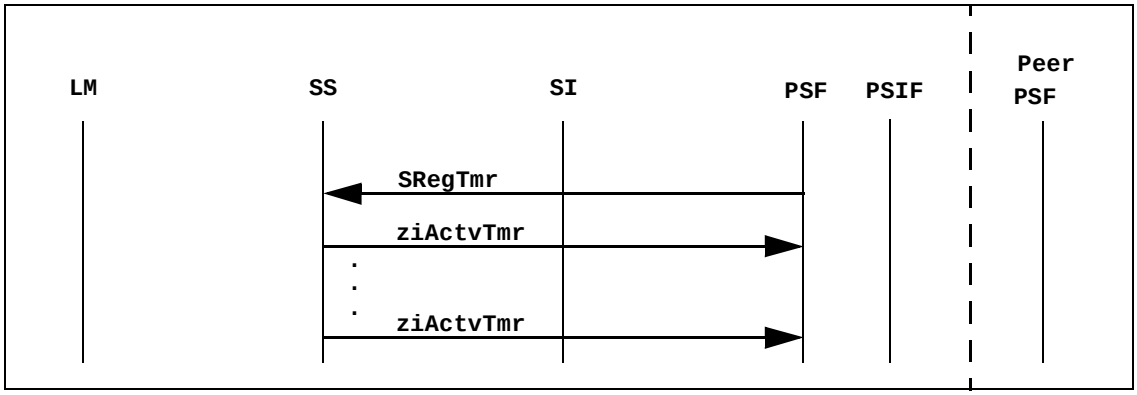


Figure 4-3: Data flow—timer activation procedure

4.2 Layer Manager Interface

This section outlines the components of the layer manager interface. The layer manager interface provides control to the management entity to execute various fault tolerant procedures on ISUP (PSIF - ISUP), such as warmstart and controlled switchover.

4.2.1 Management–Configuration

The management–configuration procedure is used by the layer manager to configure the various elements of the PSF - ISUP software. This procedure is initiated by the layer manager. The PSF - ISUP configuration request primitive (**ZiMiLzi CfgReq**) can be called more than once, but it must be called before ISUP (PSIF - ISUP) binds with its service provider and its service user.

The following PSF - ISUP configuration request primitive types can be called:

Name	Description
General	Passes parameters that apply to the entire PSF - ISUP. Used primarily to register the PSF - ISUP timer with system services and to configure the post structure of the layer manager. It can be called more than once for reconfiguration at run time.
Peer SAP	Passes parameters that apply to the peer SAP. Used primarily to configure the post structure towards the peer PSF - ISUP. It can be called more than once for reconfiguration at run time.

The configuration request primitive type is specified by the **cmPFthaMngmt.hdr.elmId** field.

For proper operation, the configuration request primitive types must be called in the following order:

1. **STGEN**
2. **STPEERSAP**

System services primitives are called during the management–configuration procedure.

Note: *The register timer (**SRegTmr**) system services primitive will be called during the general configuration request primitive to register the PSF timer activation (**ziActvTmr**) function.*

The **cmPFthaMngmt.t.cfg** structure is used to specify parameters used by the configuration request primitive. See Section 3.2.2.1, "**ZiMiLziCfgReq**."

The data flow is:

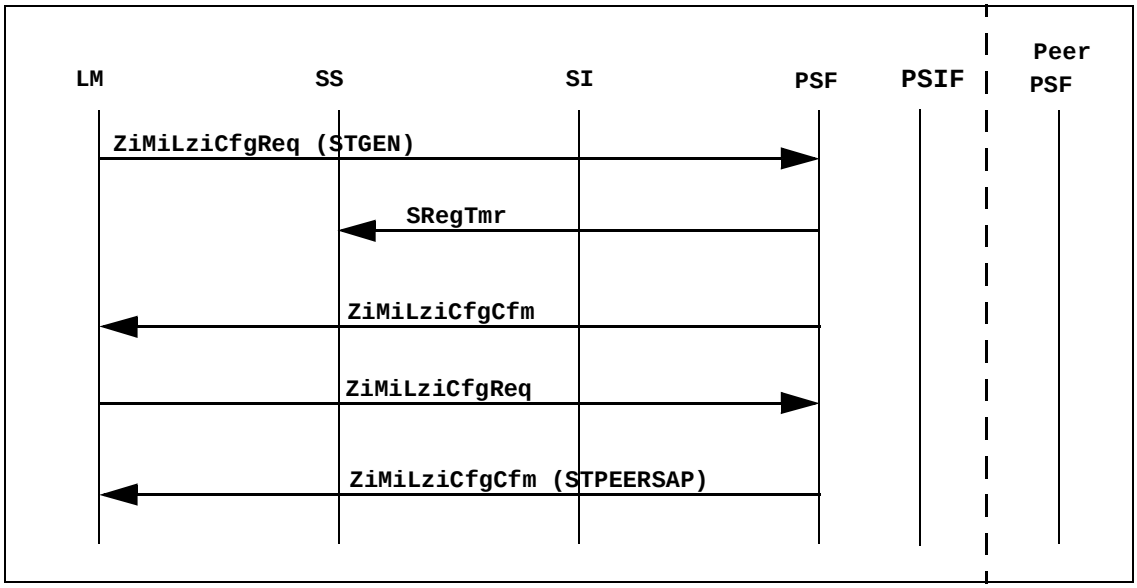


Figure 4-4: Data flow—configuration procedure

Note: *PSF - ISUP must be configured after ISUP (PSIF - ISUP)'s general configuration.*

4.2.1.1 Retry of Configuration Request

The layer manager can resend the same configuration request if it does not receive a configuration confirm from the PSF - ISUP. The configuration request or confirm can get lost due to unreliable communication media between the layer manager and PSF - ISUP. If PSF - ISUP receives a configuration request that is already in a configured state, it handles the request as reconfiguration request (that is, it configures only those parameters that are reconfigurable at run time). Therefore, repeating the same configuration request more than once does not affect the PSF - ISUP operation.

4.2.2 Management–Solicited Status

The management–solicited status procedure is used by the layer manager to gather solicited status information about the various elements of the PSF - ISUP software. This procedure is initiated by the layer manager. The PSF - ISUP status request primitive (**ZiMiLzi StaReq**) can be called more than once, any time after the management–configuration procedure.

The following PSF status request primitive types can be called:

1. General
2. System ID
3. Peer SAP

The status request primitive type is specified by the **cmPFthaMngmt.hdr.elmId** field.

The status confirm **(ZiMiLzi StaCfm**) primitive and other system services primitives are called during the status procedure to return the status value.

The **cmPFthaMngmt.t.sta** structure is used to specify parameters used by the status request and status confirm primitives. See Section 3.2.2.3, "**ZiMiLzi StaReq**" and Section 3.2.2.4, "**ZiMiLzi StaCfm**."

The data flow is:

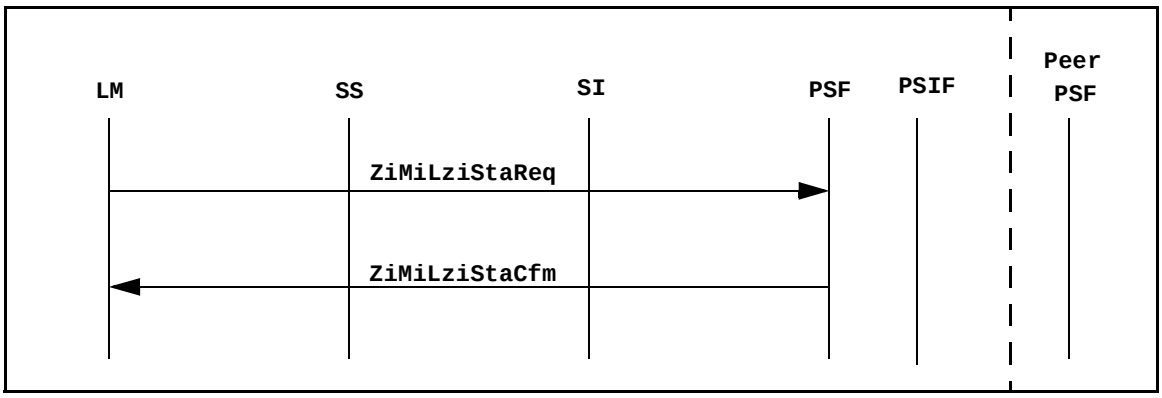


Figure 4-5: Data flow—management–solicited status procedure

4.2.2.1 Retry of Status Request

The layer manager may resend the same status request if it does not receive a status confirm from the PSF - ISUP in response. The status request or confirm may get lost due to unreliable communication media between the layer manager and PSF - ISUP. PSF - ISUP handles the status request in the same manner each time, responding with a status confirm.

4.2.3 Management–Unsolicited Status

The management–unsolicited status (alarm) procedure indicates the occurrence of an abnormal condition in PSF - ISUP. PSF - ISUP initiates this procedure and sends the status indication primitive (**ZiMiLziStaInd**) to the layer manager each time an abnormal condition is detected. PSF - ISUP's status indication primitive can be called any time after the configuration procedure if the unsolicited status has been enabled. PSF - ISUP's status indication primitive will not be called if the unsolicited status has been disabled. The unsolicited status can be enabled or disabled with the management–control procedure.

The **cmPFthaMngmt.t.usta** structure is used to specify parameters used by the status indication primitive. See Section 3.2.2.5, "**ZiMiLziStaInd**."

The data flow is:

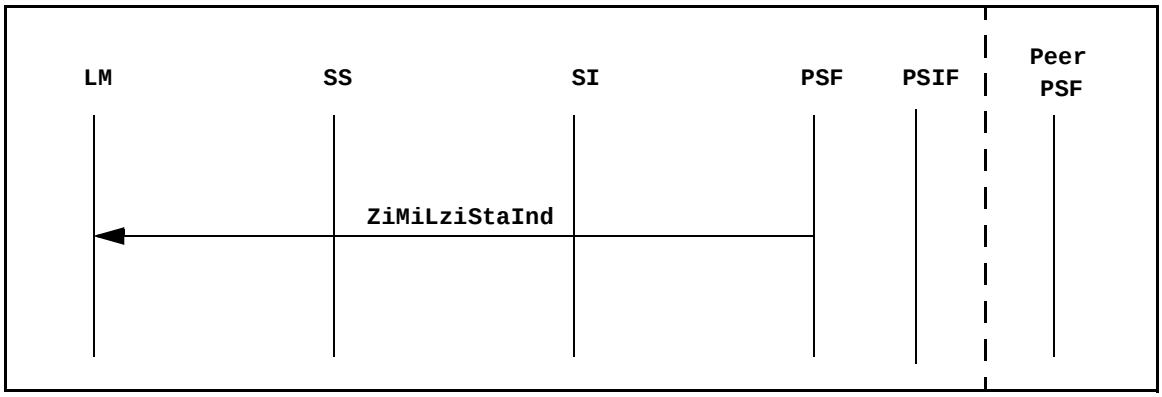


Figure 4-6: Data flow—management–unsolicited status procedure

4.2.4 Management–Control

The management–control procedure is used by the layer manager to control various elements of the PSF - ISUP software. This procedure is initiated by the layer manager. The PSF - ISUP control request primitive (**`ZiMiLziCntrlReq`**) can be called more than once, any time after the management–configuration procedure.

In response to every control request, PSF - ISUP sends a control confirm (**`ZiMiLziCntrlCfm`**), indicating success or failure of the control request in the status field of the **`cfm`** structure. See Section 3.2.2.7, "**`ZiMiLziCntrlCfm`**."

Management control from layer manager is grouped into two types: general control and peer SAP control. General control refers to specific actions that PSF - ISUP has to perform. Peer SAP control indicates to PSF - ISUP whether the peer SAP (active or standby) is available.

General Control

The following control actions are provided in general control :

- Go Active
- Retry Go Active control request
- Go Standby
- Retry Go Standby control request
- Shutdown
- Retry Shutdown control request
- Disable / Enable Alarms
- Retry Disable / Enable Alarms control request
- Disable / Enable Debug Prints
- Retry Disable / Enable Debug Prints control request
- Warmstart
- Retry Warmstart control request
- Synchronize
- Retry Synchronize control request
- Abort
- Retry Abort control request

Peer SAP Control

The following control actions are provided in peer SAP control:

- Disable Peer SAP
- Retry Disable Peer SAP control request

The data flow is:

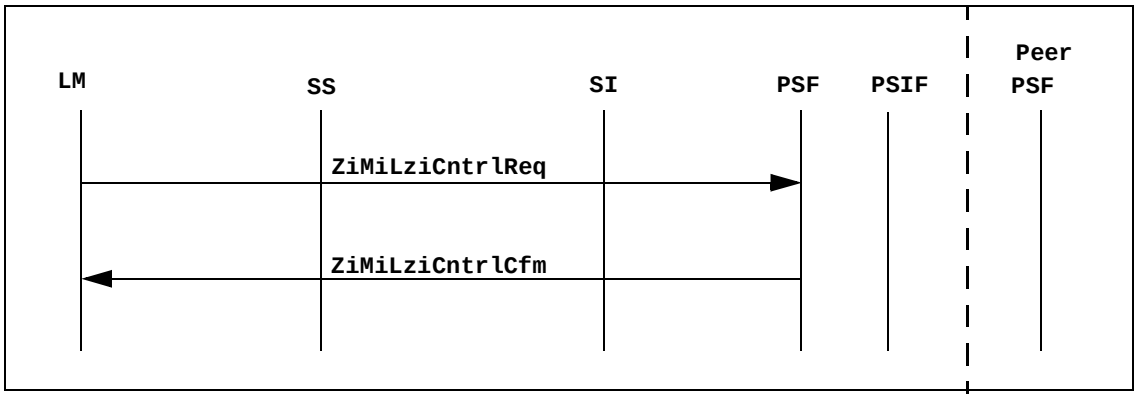


Figure 4-7: Data flow—management–control procedure

4.2.4.1 Go Active

This action is used by the layer manager to provide ISUP (PSIF - ISUP) active status. The **action** and the **subAction** fields in the **cntrl** structure are as follows:

cmPFthaMngmt.t.cntrl.action = AGO_ACT
cmPFthaMngmt.t.cntrl.subAction = ADIS_PEER_SAP or
cmPFthaMngmt.t.cntrl.subAction = ADIS_ENA_PEER_SAP

The **subAction** is **ADIS_ENA_PEER_SAP** if the peer is standby. For example, to perform controlled switchover, the layer manager sends a request to the standby to become active and enable the peer SAP.

The **subAction** is **ADIS_PEER_SAP** if the peer is Out-Of-Service (OOS). For example, the layer manager sends a request to the standby to become active with peer SAP disabled to perform forced switchover.

The layer manager can send this control request to PSF - ISUP in the following conditions:

1. When the current status of ISUP (PSIF - ISUP) is OOS:

   During system initialization, after ISUP (PSIF - ISUP) and PSF - ISUP have been configured, the layer manager requests that PSF - ISUP make ISUP (PSIF - ISUP) active from OOS state. ISUP (PSIF - ISUP) starts its operation (for example, binding) only after becoming active.

2. When the current status of ISUP (PSIF - ISUP) is standby:

   The layer manager sends the request to standby PSF - ISUP under the following conditions:

   - During controlled switchover, when the layer manager requests that the active PSF - ISUP update the standby PSF - ISUP with the current transient states of ISUP (PSIF - ISUP). Subsequently, the layer manager requests that the standby PSF - ISUP make ISUP (PSIF - ISUP) active. The new active ISUP (PSIF - ISUP) starts handling the protocol events from the upper or lower layer.
   - During forced switchover, when the layer manager requests that the standby PSF - ISUP make ISUP (PSIF - ISUP) active. The layer manager initiates this activity when the current active ISUP (PSIF - ISUP) has to be made OOS. To keep the loss minimal, the layer manager holds the traffic coming to ISUP (PSIF - ISUP) from the upper and lower layers before making the standby active. See Section 5.1, "Forced Switchover," in the *Fault-Tolerant/High-Availability (FT/HA) Core Service Definition*.

3. When the current status of ISUP (PSIF - ISUP) is active:

   During controlled switchover, the layer manager requests that the active PSF - ISUP update the standby with transient state information. The layer manager then changes the status of active to standby and the status of standby to active to complete the controlled switchover procedure. The layer manager can abort the controlled switchover procedure in between; that is, it may not swap the status of active and standby even though the active PSF - ISUP has updated the standby with the transient state information. Before the current active ISUP (PSIF - ISUP) resumes operation, the layer manager must send a request to the active to become active again.

4.2.4.1.1 Retry of Go Active Control Request

If the layer manager does not receive the control confirm for the control request to make PSF - ISUP active, the layer manager can retry the request. If PSF - ISUP has already received the control request, it does not take any action except for sending a control confirm indicating success. Therefore, this control request can be retried any number of times in case of lost primitives.

4.2.4.2 Go Standby

This action is used by the layer manager to provide ISUP (PSIF - ISUP) standby status. The **action** and the **subAction** fields in the **cntrl** structure are as follows:

```
cmPFthaMngmt.t.cntrl.action = AGO_SBY
cmPFthaMngmt.t.cntrl.subAction = SAENA_PEER_SAP
```

The **subAction** in this control request is always **SAENA_PEER_SAP**. When a PSF - ISUP is made standby, the active PSF - ISUP must already exist.

The layer manager can send this control request to PSF - ISUP under the following conditions:

1. When the current status of ISUP (PSIF - ISUP) is OOS:

   After configuring ISUP (PSIF - ISUP) and PSF - ISUP, the layer manager requests that PSF - ISUP make ISUP (PSIF - ISUP) standby from OOS state (this occurs when the active ISUP (PSIF - ISUP) already exists).

2. When the current status of ISUP (PSIF - ISUP) is active:

   During controlled switchover, the layer manager requests that the active PSF - ISUP update the standby PSF - ISUP with the current transient states of ISUP (PSIF - ISUP). Subsequently, the layer manager requests that the active PSF - ISUP make ISUP (PSIF - ISUP) standby. The new standby ISUP (PSIF - ISUP) starts receiving run-time state updates from the active PSF - ISUP.

3. When the current status of ISUP (PSIF - ISUP) is standby:

   During controlled switchover, the layer manager requests that the active PSF - ISUP update the standby with transient state information. The layer manager then changes the status of active to standby and the status of standby to active to complete the controlled switchover procedure. The layer manager can abort the controlled switchover procedure in between—that is, it may not swap the status of active and standby even though active PSF - ISUP has updated the standby with the transient state information. Before the current active ISUP (PSIF - ISUP) resumes operation, the layer manager has to send a request to the standby to remain standby.

For data flow, see Figure 4-7.

4.2.4.2.1 Retry of Go Standby Control Request

If the layer manager does not receive the control confirm for the control request to make PSF - ISUP standby, the layer manager can retry the request. If PSF - ISUP has already received the control request, it does not take any action except to send a control confirm indicating success. Therefore, this control request can be retried any number of times in case of lost primitives.

4.2.4.3 Shutdown

The layer manager sends this control request to PSF - ISUP to shutdown its operation. PSF - ISUP deletes all its dynamic resources, as well as any timers that are running. PSF - ISUP can be fully reconfigured after the shutdown.

The **action** and **subAction** fields in the **cntrl** structure are as follows:

```
cmPFthaMngmt.t.cntrl.action = ASHUTDOWN
cmPFthaMngmt.t.cntrl.subAction = UNUSED
```

For the data flow, see Figure 4-7.

4.2.4.3.1 Retry of Shutdown Control Request

If the layer manager does not receive the control confirm for the control request to shutdown the PSF - ISUP operation, the layer manager can retry the request. If PSF - ISUP has already received the control request, it takes no action and sends a control confirm indicating success.

4.2.4.4 Disable/Enable Alarms

The layer manager can disable or enable alarm generation from PSF - ISUP. By default, the alarms are disabled.

To enable the alarm generation, the **action** and the **subAction** fields in the **cntrl** structure are as follows:

```
cmPFthaMngmt.t.cntrl.action = AENA
cmPFthaMngmt.t.cntrl.subAction = SAUSTA
```

To disable the alarm generation, the **action** and **subAction** fields in the **cntrl** structure are as follows:

```
cmPFthaMngmt.t.cntrl.action = ADISIMM
cmPFthaMngmt.t.cntrl.subAction = SAUSTA
```

For data flow, see Figure 4-7.

4.2.4.4.1 Retry of Disable/Enable Alarms Control Request

If the layer manager does not receive the control confirm for the control request to disable / enable alarms in PSF - ISUP, the layer manager can retry the request. PSF - ISUP takes the same action again without affecting PSF - ISUP states and sends a control confirm indicating success. Therefore, this control request can be retried any number of times in case of lost primitives.

4.2.4.5 Disable/Enable Debug Prints

The layer manager can disable or enable debug prints for various debug levels from PSF - ISUP. By default, the debug prints are disabled. To enable debug prints, the **`action`** and **`subAction`** fields in the **`cntrl`** structure are as follows:

```
cmPFthaMngmt.t.cntrl.action = AENA
cmPFthaMngmt.t.cntrl.subAction = SADBG
```

To disable the debug prints, the **`action`** and **`subAction`** fields in the **`cntrl`** structure are as follows:

```
cmPFthaMngmt.t.cntrl.action = ADISIMM
cmPFthaMngmt.t.cntrl.subAction = SADBG
```

For data flow, see Figure 4-7.

4.2.4.5.1 Retry of Disable/Enable Debug Prints Control Request

If the layer manager does not receive the control confirm for the control request to disable / enable debug prints in PSF - ISUP, the layer manager can retry the request. PSF - ISUP takes the same action again, without affecting PSF - ISUP states, and sends a control confirmation indicating success. This control request can, therefore, be retried any number of times in the case of lost primitives.

4.2.4.6 Warmstart

The layer manager initiates this procedure when an OOS node must be made standby. The layer manager performs the following steps to warmstart an OOS node:

1. Configures the standby ISUP (PSIF - ISUP) and PSF - ISUP
2. Sends a control request to PSF - ISUP to make ISUP (PSIF - ISUP) standby
3. Sends a control request to the active PSF - ISUP to warmstart the standby with the **`action`** and **`subAction`** fields in the **`cntrl`** structure as follows:

   ```
   cmPFthaMngmt.t.cntrl.action = AWARMSTART
   cmPFthaMngmt.t.cntrl.subAction = NOTUSED
   ```

Two possible results of this procedure are:

1. Success:

   The active PSF - ISUP successfully sends the state update messages to the peer. In this case, the active sends a control confirmation primitive to the layer manager with the **`status`** in the **`cfm`** structure as **`LCM_PRIM_OK`**, indicating that warmstart is successful.

2. Failure:

   The active PSF - ISUP encounters a failure during the state update procedure (for example, a state update message gets lost because of unreliable communication media between the active and the standby node). The active sends a control confirmation primitive to the layer manager with the **`status`** in the **`cfm`** structure as **`LCM_PRIM_NOK`**, indicating that warmstart has failed. The active PSF - ISUP also disables the peer SAP so that the run-time state update to the peer is stopped.

The data flow is:

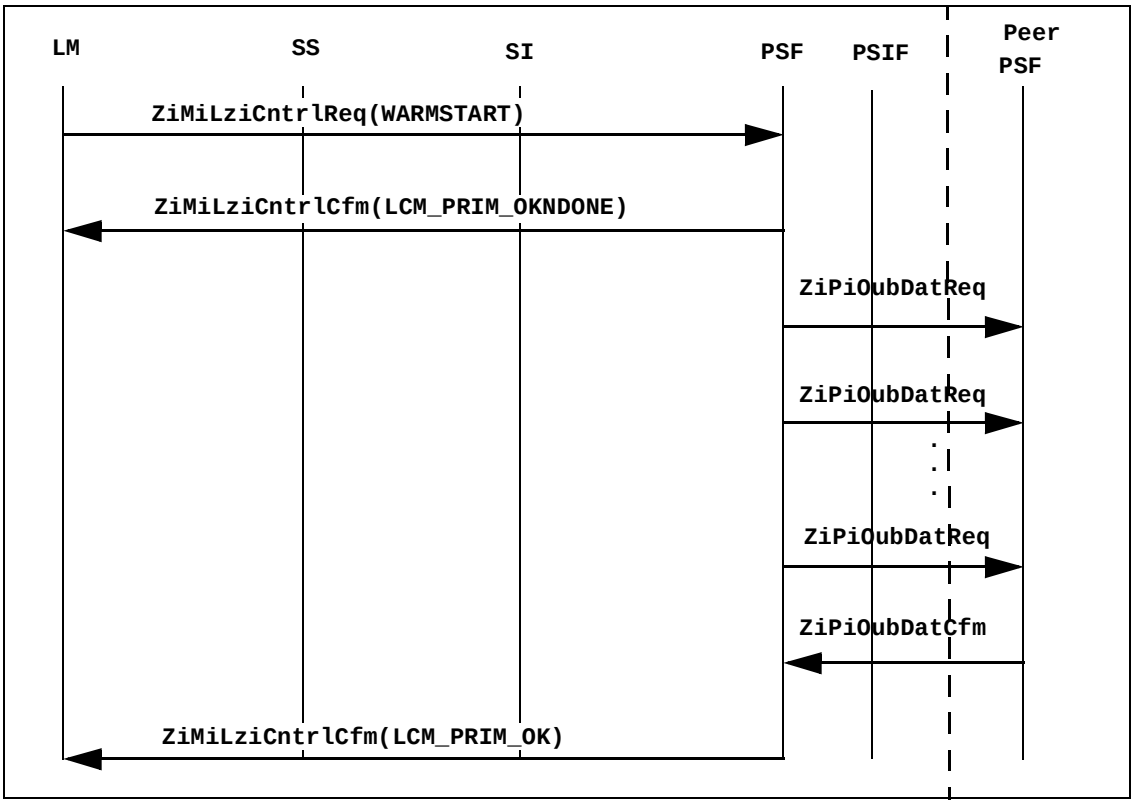


Figure 4-8: Data flow—warmstart procedure

4.2.4.6.1 Retry of Warmstart Control Request

If the warmstart control request or control confirm is lost, the layer manager can resend the control request. The layer manager can lose either the first or second control confirm, resulting in the layer manager resending the control request. If the layer manager retries, PSF - ISUP can receive the control request in any of the following states:

- PSF - ISUP has completed the warmstart successfully (which is the case when the second control confirm gets lost) and, thus, the peer SAP is enabled. In this case, PSF - ISUP returns a control confirmation indicating that the warmstart is complete.
- The warmstart is still continuing (which is the case when the first control confirm gets lost). PSF - ISUP sends a control confirm indicating that the warmstart is continuing.
- The peer SAP is disabled—that is, either the warmstart never happened (which is the case when the control request gets lost) or the warmstart failed (which is the case when the second control confirm gets lost, indicating failure). PSF - ISUP starts the warmstart procedure and sends a control confirm indicating warmstart is continuing. After completing the warmstart, PSF - ISUP sends the second control confirm indicating that the warmstart is complete.

4.2.4.7 Synchronize

The layer manager initiates this procedure during the controlled switchover of an ISUP (PSIF - ISUP) node. Before swapping the status of the active and standby ISUP (PSIF - ISUP) nodes, the layer manager sends this control request to the active PSF - ISUP to update the standby with the current transient state information. The layer manager sends this control request to the active only after freezing the traffic coming to ISUP (PSIF - ISUP), so that ISUP (PSIF - ISUP) states do not change while the active is continuing to update the standby PSF - ISUP.

The layer manager sends a control request to the active PSF - ISUP with the **`action`** and the **`subAction`** fields in the **`cntrl`** structure as follows:

```
cmPFthaMngmt.t.cntrl.action = ASYNCHRONIZE
cmPFthaMngmt.t.cntrl.subAction = NOTUSED
```

The active PSF - ISUP sends a control confirmation immediately with **`status`** in the **`cfm`** structure as **`LCM_PRIM_OKNDONE`**, indicating that synchronization is progressing.

The active PSF - ISUP freezes its timers so that timer firing does not result in any state changes. The active then updates the peer with all its transient state update information.

Two possible results of this procedure are:

1. Success:

   The active PSF - ISUP successfully sends the state update messages to the peer. In this case, the active sends a control confirmation primitive to the layer manager with **`status`** in the **`cfm`** structure as **`LCM_PRIM_OK`**, indicating that the warmstart is successful.

2. Failure:

   The active PSF - ISUP encounters a failure during the state update procedure (for example, a state update message becomes lost due to unreliable communication media between the active and the standby node). The active PSF - ISUP sends a control confirmation primitive to the layer manager with **`status`** in the **`cfm`** structure as **`LCM_PRIM_NOK`**, indicating that synchronization has failed. The active does not disable the peer SAP so that the active node can continue to send run-time state updates to the standby node. Before restarting the traffic to ISUP (PSIF - ISUP), the layer manager must send a control request to active PSF - ISUP to remain active and to standby PSF - ISUP to remain standby.

The data flow is:

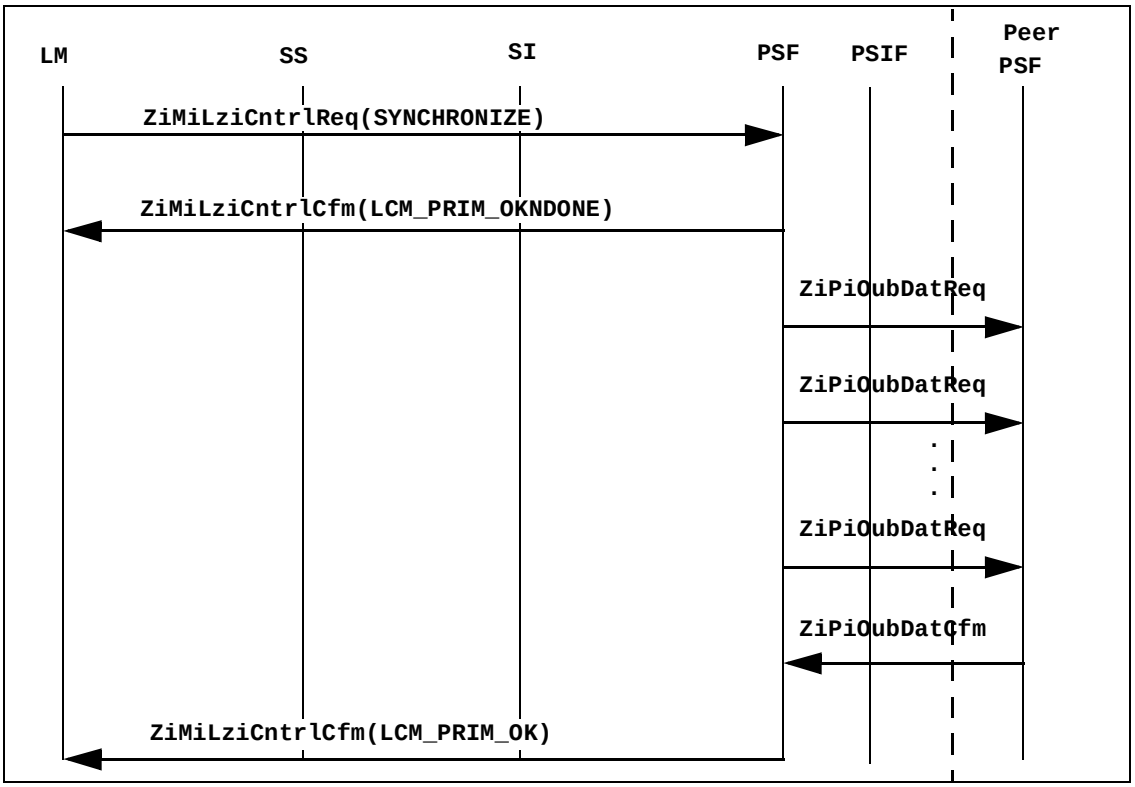


Figure 4-9: Data flow—synchronization procedure

4.2.4.7.1 Retry of Synchronize Control Request

If the synchronize control request or control confirm is lost, the layer manager can resend the control request. The layer manager can lose either the first control confirm or the second, resulting in the resend. If the layer manager retries the send, PSF - ISUP can receive the control request in one of the following states:

- The synchronization procedure is ongoing

  This occurs when the first control confirm gets lost. PSF - ISUP sends a control confirm indicating that the synchronization is continuing.

- PSF - ISUP is not currently carrying out the synchronization

  In this case, either the synchronization never happened (which is the case when the control request itself gets lost) or the synchronization was completed (which is the case when the second control confirm gets lost). PSF - ISUP starts the synchronization procedure and sends a control confirmation indicating that synchronization is continuing. After completing the synchronization, PSF - ISUP sends the second control confirm indicating that synchronization is complete.

4.2.4.8 Abort

The warmstart or controlled switchover state update can be a long procedure, depending on the ISUP (PSIF - ISUP) data to be updated. While the warmstart or controlled switchover state update is continuing (that is, the layer manager has received the first control confirm indicating that state update is continuing), the layer manager can send this control request to the active PSF - ISUP to abort the state update procedure. The active PSF - ISUP responds with a control confirm for the stop request. The active PSF - ISUP does not send the second control confirm to indicate the completion of warmstart or synchronize the state update.

If a warmstart state update is aborted, the active PSF - ISUP disables the peer SAP. If the synchronize state update is aborted, the layer manager must send a control request to the active PSF - ISUP to remain active and to the standby PSF - ISUP to remain standby before releasing the traffic to ISUP (PSIF - ISUP).

If the active PSF - ISUP receives this request in idle state (that is, the active PSF - ISUP is not carrying out the warmstart state update or controlled switchover state update), then the PSF - ISUP sends a control confirm to the layer manager indicating success. The PSF - ISUP does not perform any other action

The layer manager sends a control request to the active PSF - ISUP with the **`action`** and **`subAction`** fields in the **`cntrl`** structure as follows:

```
cmPFthaMngmt.t.cntrl.action = AABORT
cmPFthaMngmt.t.cntrl.subAction = NOTUSED
```

For the data flow, see Figure 4-7.

4.2.4.8.1 Retry of Abort Control Request

If the layer manager does not receive the control confirm for the control request to abort warmstart or the controlled switchover, the layer manager can retry the request. PSF - ISUP takes the same action again without affecting PSF - ISUP states and sends a control confirm indicating success. This control request can be retried any number of times, therefore, when primitives are lost.

4.2.4.9 Disable Peer SAP

The layer manager sends this control request to the active PSF - ISUP when the standby PSF - ISUP goes OOS. The active PSF - ISUP stops sending run-time state update messages to the peer. The **action** and the **subAction** fields in the **cntrl** structure are as follows:

```
cmPFthaMngmt.t.cntrl.action = AUBND_DIS
cmPFthaMngmt.t.cntrl.subAction = UNUSED
```

For the data flow, see Figure 4-7.

4.2.4.9.1 Retry of Disable Peer SAP Control Request

If the layer manager does not receive the control confirm for the control request to disable the peer SAP in active PSF - ISUP, the layer manager can retry the request. PSF - ISUP takes the same action again without affecting PSF - ISUP states and sends a control confirm indicating success. Therefore, this control request can be retried any number of times in case of lost primitives.

4.3 Peer Layer Interface

This interface consists of procedures to carry out state updates from active PSF - ISUP to standby PSF - ISUP. The procedures related to this interface are as follows:

- Run-time state update
- Warmstart state update
- Controlled switchover state update

4.3.1 Run-time State Update

The active ISUP (PSIF - ISUP) can change its internal stable states while handling protocol events. When the active ISUP (PSIF - ISUP) changes its internal states, the active PSF - ISUP sends the modified state information to the standby (if a standby exists—that is, the peer SAP is enabled) in the update messages. Each run-time state update message carries an incremented sequence number. If the standby PSF - ISUP detects a sequence error in the run-time update messages, it sends an alarm to the layer manager. See Section 3.2.2.5, "**ZiMiLziStaInd**."

The data flow is:

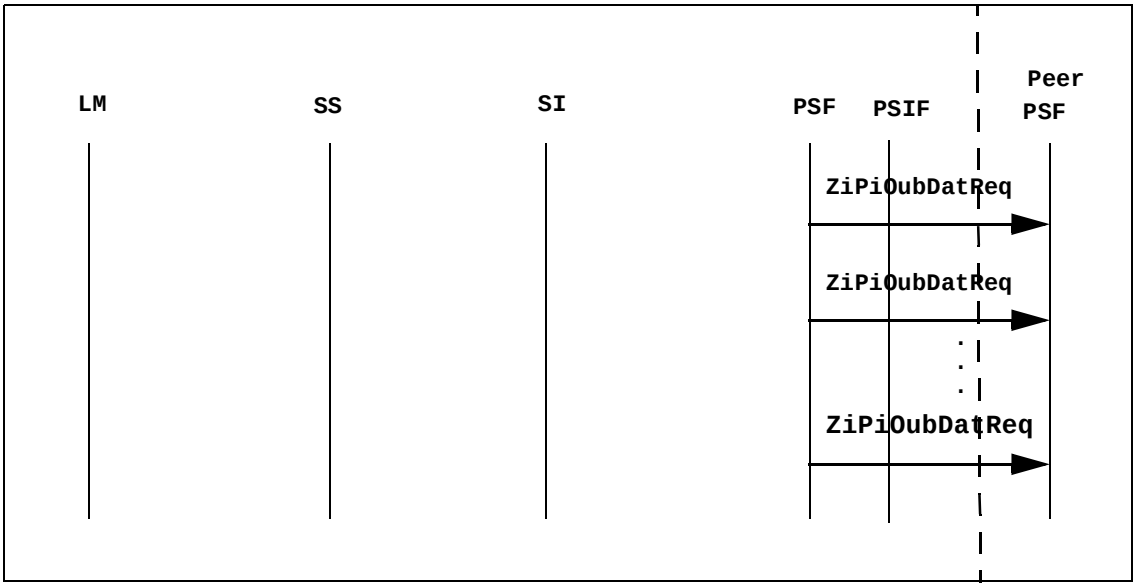


Figure 4-10: Data flow—run-time state update procedure

4.3.2 Warmstart State Update

The active PSF - ISUP sends the entire stable state information to the peer when the active PSF - ISUP receives a control request from the layer manager to warmstart the peer. The active PSF - ISUP can send multiple update messages to accommodate the complete stable state information. The active PSF - ISUP sends all warmstart messages with an incremented sequence number. If the standby receives all the update messages, it sends a confirmation to the active indicating success. If the standby finds a sequence error in the update messages, it sends a confirmation to the active indicating failure.

For the data flow, see Figure 4-8.

4.3.3 Controlled Switchover State Update

The active PSF - ISUP sends the entire transient state information to the peer when the active PSF - ISUP receives a control request from the layer manager to synchronize the peer. The active PSF - ISUP can send multiple update messages to accommodate the complete transient state information. The active PSF - ISUP sends all synchronization messages with an incremented sequence number. If the standby receives all the update messages, it sends a confirmation to the active indicating success. If the standby finds a sequence error in the update messages, it sends a confirmation to the active indicating failure.

For the data flow, see Figure 4-9.

5 PORTATION ISSUES

PSF - ISUP is not an independent product. It compiles only with Trillium's ISUP and PSIF - ISUP. Most of the general portation issues that affect PSF - ISUP are described in the *ISUP Portation Guide*. See the References section of this manual for more information.

The following table lists ISUP source files:

Name	Description
zi_bdy1.c	Contains all the incoming primitives to PSF - ISUP from the layer manager, peer PSF - ISUP, PSIF - ISUP, and ISUP protocol layer
zi_bdy2.c	Contains the support functions to carry out tasks such as forced switchover and controlled switchover
zi_bdy3.c	Contains all the packing functions invoked by the active PSF - ISUP for sending state update messages to the standby PSF - ISUP
zi_bdy4.c	Contains all the unpacking functions invoked by the standby PSF - ISUP when receiving state update messages from the active PSF - ISUP
zi_ptmi.c	Contains all the packing functions invoked by PSF - ISUP for sending primitives to layer manager
zi_ptpi.c	Contains all the packing functions invoked by PSF - ISUP for sending primitives to the peer PSF - ISUP
zi_ex_ms.c	Contains all the unpacking functions invoked by PSF - ISUP when it receives primitives from the layer manager or peer PSF - ISUP
zi_id.c	Contains all the current versions of the PSF - ISUP software
zi.h	The header file for PSF - ISUP internal defines
zi.x	The header file for PSF - ISUP internal structures and function prototypes
lzi.h	The header file containing defines for the events between PSF - ISUP and the layer manager
lzi.x	The header file containing the prototypes of the primitives between PSF - ISUP and the layer manager
zi_err.h	The header file for PSF - ISUP error defines
zi_acc1.c	Contains the test engine for running PSF - ISUP acceptance tests
zi_acc2.c	Contains PSF - ISUP acceptance tests
zi_acc.h	The header file for PSF - ISUP acceptance test-specific defines
zi_acc.x	The header file for PSF - ISUP acceptance test-specific **typedef**s
zi.mak	**Makefile** for building portable (PSF - ISUP and ISUP portable files) and acceptance tests executables for various environments
cm_pftha.h	Common header file for portable FT / HA defines
cm_pftha.x	Common header file for portable FT / HA structures and function prototypes

Name	Description
cm_pftha.c	Common portable FT/HA functions
smzibdy1.c	PSF - ISUP stack manager - Body file
smziptmi.c	PSF - ISUP stack manager - Portable manager interface
smziexms.c	PSF - ISUP stack manager - External interface file
smzi_err.h	PSF - ISUP stack manager - Error code defines

In addition to the above list, PSF - ISUP uses both the common files and the ISUP stack manager files described in the *ISUP Portation Guide.*

5.1 Prepare Files for Make

PSF - ISUP software can only be compiled with Trillium's ISUP software (release 2.8 and any future releases). The following steps must be taken before the PSF - ISUP software can be compiled:

1. ISUP release 2.18 offers backward compatibility in the upper layer, lower layer, and layer manager interfaces. Backward compatibility can be achieved by commenting out the **SITVER2** (upper interface), **SNT2** (lower, MTP-3 interface), **SPT2** (lower, SCCP interface), and **ST_LMINT3** (layer manager interface) compilation flags in the **#ifdef SI** section of the **envopt.h** file. Backward compatibility is not available when ISUP software is combined with PSF - ISUP software. Customers should make sure that the upper and lower layers and the management entity are compliant with any new interfaces and that no backward compatibility in any interface is required.
   - The **#ifdef ZI** section of **envopt.h** file must be reviewed for the following compile-time options:

Flag	Description
LCZIMILZI	This flag should be enabled to compile the loosely coupled layer manager interface of PSF - ISUP. At run time, the loosely coupled layer manager interface is selected by setting **pst->selector** value to 0. If this flag is not defined, tight coupling is assumed. At run time, the tightly coupled layer manager interface is selected by setting **pst->selector** value to 1. For a tightly coupled layer manager interface, the **zi_ptmi.c** file should be compiled with the **SM** option.
CUSTENV	PSF - ISUP assumes a similar environment (compiler and operating system) on active and standby nodes. Based on this assumption, Trillium supplies non-portable update message packing / unpacking functions. For a different environment, customers can enable this flag and provide portable packing / unpacking in the **ziPtPkStruct (zi_bdy3.c)** and **ziPtUnpkStruct (zi_bdy4.c)** functions.

2. The **DEBUGP** flag in **envopt.h** file can be enabled so that debug prints from PSF - ISUP are enabled. If this flag is enabled, the layer manager can send a control request to enable the debug prints at different levels. See Section 3.2.2.6, "**ZiMiLziCntrlReq**."
3. The **LMINT3** flag is enabled by default in the top section of the **envopt.h** file. This flag must remain enabled for PSF - ISUP.

4. To avoid race conditions at the management interface, PSF - ISUP should use the same management coupling as ISUP.
5. PSF - ISUP uses the **cmPFthaYield()** function (file **cm_pftha.c**) to avoid starving other system processes during dynamic control blocks update. This is not an issue if PSF - ISUP is ported on an operating system that supports time slicing, as the operating system itself takes care of descheduling the task in that case. For porting PSF - ISUP on a non-preemptive operating system, the **cmPFthaYield** function must be modified to make a descheduling system call.
6. ISUP and PSF - ISUP files must be compiled with both **ZI** and **SI** compile-time flags. For details, refer to the **zi.mak** file.
7. PSF - ISUP also compiles with PSIF - ISUP for FT / HA support in PSIF - ISUP. In this case, PSF - ISUP files must be compiled with **IW** compile-time flag.

5.2 PSF - ISUP Compilation

To run the PSF - ISUP acceptance test, compile **acc** target in **zi.mak** file. This file also compiles the ISUP and PSIF - ISUP code. Therefore, the ISUP or PSIF - ISUP **makefile** should not be used when PSF - ISUP is being used with ISUP and PSIF - ISUP. To compile PSF - ISUP's portable code, use **pt** target in **zi.mak** file.

SS7 Glossary

A

Adjacent Signalling Points: Two signalling points directly connected through signalling links.

Authentication Center (AC): A database that stores security codes embedded into the memory of cellular phones. This code, along with the particular serial number of a given phone, prevents the use of unauthorized cellular devices within a particular network. See **Equipment Identity Register (EIR)**.

B

Base Station Subsystem (BSS): Within a cellular communications network, antenna sites (also known as cell sites) are made up of a **Base Transceiver Station (BTS)** and a **Base Station Controller (BSC).** The BSS is the pairing of the BTS and BSC. The BTS communicates with cellular phones within a given network, connecting the caller with the cell. The BSC is the interface between the BTS and any switching facilities that may be needed by the caller. See **Mobile Switching Center (MSC)**.

C

Combined Link Set: A collection of link sets that perform load sharing.

Common Channel Signalling: A signalling technique in which the signalling information is sent across the network separately from the voice and data that it is related to.

Consultative Committee International Telegraph and Telephone (CCITT): An international organization that developed communication standards such as Recommendation X.25. Replaced by the United Nation's ITU-T.

Cyclic Redundancy Check (CRC): A mathematical algorithm that derives a numerical value based on the bits in a block of data prior to transmission. If the receiving layer finds any discrepancies between the bit value of the received data packet and the accompanying frame check sequence field, then a transmission error is assumed.

D

Data Communications Equipment (DCE): Devices that handle routing and switching functions for a given network. See also **Data Terminal Equipment (DTE)**.

Destination Point Code (DPC): A node ID that identifies the destination point of a message in a signalling network. See **Originating Point Code (OPC)**.

Data Terminal Equipment (DTE): A device, such as a PC or main frame computer, that is attached to a network and is either the point of origin or the destination point of data.

E

End-to-End Signalling: Signalling that is transmitted directly between network endpoints.

Equipment Identity Register (EIR): A database that stores the serial numbers of each cellular telephone in use within a particular coverage area. In conjunction with the AC, the EIR prevents unauthorized use of cellular phones within a given geographic area.

F

Flow Control: A function that regulates the transmission of messages between adjacent protocol layers.

G

Global Title: An address which does not contain all the information necessary to route it to a specific point within the network. An example of a global title are digits dialed by the customer; in order for them to be correctly routed across an SS7 stack the SCCP translation function is needed.

Group Special Mobile (GSM): The European cellular network. Due to the use of SS7, the GSM is a more reliable network than its North American counterparts, which are still primarily analog in nature.

H

Home Location Register (HLR): A database which stores information about all cellular subscribers within a service provider's home service area. See **Visitor Location Register (VLR)**.

I

Institute of Electrical and Electronics Engineers (IEEE): Professional organization that defines network standards in a number of communication fields.

Integrated Service Digital Network (ISDN): Introduced in 1984, ISDN uses the existing infrastructure of the Public Switched Telephone Network (PSTN) to provide digital communication services for user-to-network interfaces.

Intelligent Networks Application Part (INAP): Protocol which separates switching from services in the SS7 network. Databases can be directly accessed in INAP, rather than through a Service Control Point (SCP).

International Standards Organization (ISO): Based in Geneva, Switzerland, the ISO establishes voluntary telecommunication technology standards among its ninety member countries.

ISDN User Part (ISUP): The protocol used to set up, manage, and release circuits used for voice and data transmission in the PSTN. ISUP uses out of band signalling, in which separate paths carry signalling and voice transmissions.

International Telecommunications Union - Telecommunications Standardization Sector (ITU - TSS, or more commonly ITU-T): Formerly the CCITT, this international body defines and implements recommendations and standards pertaining to the development of global telecommunications.

L

Line Information Database (LIDB): A database containing information relating to customer services, such as whether a customer subscribes to call waiting, caller identification, conference calls, and call forwarding. Also used for verification purposes with calling card services.

Load Distribution Function (LDF): Distributes traffic loads among various instances of the portable layer software (SCCP, TCAP, MTP3) based upon configuration parameters.

Local Exchange (LE): The primary switching node that provides access to the PSTN. In a basic telephony network, when a customer lifts the receiver they are connected to a local exchange, also known as a **Service Switching Point (SSP)**. The SSP provides the numbers of both the calling and called parties to a router, the **Signal Transfer Point (STP)**, so that the call can be completed across the network.

M

Media Access Control (MAC): Data link layer protocols that control traffic and data flow in multi-access channels. MACs are important in keeping LANs congestion free.

Message Transfer Part (MTP): Levels 1 through 3 of the SS7 protocol stack. Provides the upper levels with node-to-node transmission, message sequencing, and error detection / correction.

Mobile Application Part (MAP): A layer in the SS7 stack that runs on top of TCAP to query HLRs and VLRs within wireless networks. There are two standards for the MAP protocol, IS-41 and GSM. IS-41 is the ANSI standard and is used primarily in North America, while GSM is the ITU standard and is used in Europe, the U.S., and Asia.

Mobile Switching Center (MSC): Entity which receives signal strength reports from cell sites (antennas) regarding the particular strength or weakness of a cellular phone connection. Based upon these reports, the MSC determines the cell site that will then handle the given call. The MSC directly communicates with the BSC through digital facilities in the 64 Kbps range.

Multiprocessor Operating System (MOS): Trillium portable C source code designed to operate as an operating system on any embedded system or as a guest operating system under DOS, UNIX, or Solaris.

N

Network Service Part: The combination of Message Transfer Part (MTP) and Signalling Connection Control Part (SCCP).

O

Operations Administration and Maintenance (OAM): Non-data cells that provide the network with basic data management and performance diagnosis functions in order to prevent a catastrophic network or system failure.

Origination Point Code (OPC): A node ID that identifies the originating point of a message in a signalling network. See **Destination Point Code (DPC)**.

Open Systems Interconnection (OSI): An architectural system, developed by the ISO, for the interconnection of multiple data communication systems. The seven standardized layers of this model, with their associated layer managers, are: application, presentation, session, transport, network, data link, and physical. Each layer builds upon the services provided by the layers beneath it.

P

Peer Entities: Communicating entities residing in the same layer but within different nodes.

Protocol Data Unit (PDU): A unit of data used to exchange information between peer protocols communicating across a network, typically in the form of a packet with headers and/or trailers.

Protocol Specific Function (PSF): As stand-alone software, PSF provides fault tolerance functionality to portable protocol layers. In conjunction with Load Distribution Function (LDF), PSF also provides high availability functionality to the portable layer.

Protocol Specific Interface Function (PSIF): PSIF provides a generic interface from the underlying protocol layer to Trillium's Interworking Call Control (ICC) software. PSIF - ISUP understands the interface as implemented in the underlying protocol layer and maps it to a uniform interface required by ICC.

Public Switched Telephone Network (PSTN): Basic telephony system, through which calls are established and torn down between the user and a local exchange.

Q

QoS (Quality of Service): Quality of Service parameters center upon the working or contractual relationship between the service user and the service provider. These parameters, which are negotiated in advance, deal with the speed of the required service, the duration of the service, the rate of delivery, as well as desired or needed network characteristics, such as acceptable delay variations and errors in transmission.

S

Service Access Point (SAP): Information flow between the layers of a network is via Service Access Points (SAPs). The standardized interface of primitives and SAPs allows layers to be defined independently of each other. As long as the requirements of the layer interface are met, modifications may be made to the peer-to-peer protocol of one layer without affecting any upper or lower layer protocols.

Service Control Point (SCP): The interface between the STP and the database needed by the service user. The SCP is normally a computer connected to a mainframe computer that actually stores the needed information.

Service Switching Point (SSP): See **Local Exchange.**

Signal Transfer Point (STP): See **Local Exchange**

Signalling Connection and Control Protocol (SCCP): With TCAP, SCCP is part of layer 4 in the OSI protocol stack. In an SS7 network, SCCP provides end-to-end routing through the network of STPs. Unlike the layer below it (MTP 3), SCCP knows the entire route of the call. Connection Oriented SCCP is circuit-switched, while Connectionless SCCP is packet-switched.

Signalling Link: Abstract representation of the communications channel or connection between two logical nodes, including physical links and **Virtual Path Connections (VPCs)**.

Signalling System 7 (SS7): An ITU communication standard that first appeared in 1983, SS7 enables wireless and wireline call setup, call management, and call teardown over a digital signalling network. In addition to these basic functions, SS7 enables local number portability, the sharing of databases and connections within the PSTN, toll-free (800) and toll (900) wireline services, call forwarding, caller identification, and three-way calling.

Subsystem Number: Unique, fixed address given to a database, such as an HLR or VLR, used to route queries from SSPs through the network to the database itself.

T

Transaction Capabilities Application Part (TCAP): Layer 4 SS7 protocol that allows for remote database access (such as 800 or 900 numbers) from disparate networks using end-to-end switching.

Telephone User Part (TUP): The European equivalent to ISUP, except that TUP only supports analog circuits rather than digital circuits or data transmission.

V

Virtual Path Connection (VPC): A unidirectional concatenation of VPLs. Resources taken up in establishing individual VCCs are reduced or reserved by setting aside a given capacity within VPCs.

Visitor Location Register (VLR): Accesses the HLR through the SS7 network and stores information about subscribers who are outside of their home service area. See **Home Location Register (HLR)**.

Virtual Path Identifier (VPI): A distinct numerical marker, created within an 8-bit field in the ATM cell header, that identifies a particular VP for use by the cell.

Virtual Path Link (VPL): The unidirectional transmission of ATM cells within the life of a given VPI value—that is, from the assigning of the VPI at point A to its removal at point B.

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).

ISDN User Part Functional Specification, Trillium Digital Systems, Inc. (p / n 1091029).

ISDN User Part Portation Guide, Trillium Digital Systems, Inc. (p / n 1093029).

ISDN User Part Service Definition, Trillium Digital Systems, Inc. (p / n 1092029).

ISDN User Part Software Test Sample, Trillium Digital Systems, Inc. (p / n 1094029).

ISDN User Part Training Course, Trillium Digital Systems, Inc. (p / n 1095029).

PSF - ISUP (FT/HA) Functional Specification, Trillium Digital Systems, Inc. (p / n 1091146).

PSF - ISUP (FT/HA) Software Test Sample, Trillium Digital Systems, Inc. (p / n 1094146).

PSF - ISUP (FT/HA) Training Course, Trillium Digital Systems, Inc. (p / n 1095146).

PSIF-ISUP Functional Specification, Trillium Digital Systems, Inc. (p / n 1091141).

PSIF-ISUP Service Definition, Trillium Digital Systems, Inc. (p / n 1092141).

PSIF - ISUP Training Course, Trillium Digital Systems, Inc. (p / n 1095141).

SIT Interface Service Definition, Trillium Digital Systems, Inc. (p / n 1100010)

System Services Interface Service Definition, Trillium Digital Systems, Inc. (p / n 1111011).

Index

L

M

N

O

P

Q

R

S

T

U

W

Z