NewNet Distributed7
User Manual
Part No. 1-1600-1001-01

Release 1.6.0

February 21, 2009

35 Nutmeg Drive
Trumbull, CT 06611

Technical Assistance: (877) 698-5583
(203) 647-0580
fax: (203) 647-0580
email: support@newnet.com

TRADEMARKS

NewNet® and AccessMANAGER® are registered trademarks of NewNet Communication Technologies, LLC.

NewNet AccessMANAGER™, NewNet Connect7™, NewNet Disributed7™, NewNet Easy7™, NewNet SG™, NewNet SGC™, OTAserver™, SMserver™ are trademarks of NewNet Communication Technologies, LLC.

Sun™, Sun-3™, Sun-4™, Sun386i™, SunInstall™, SunOS™, and SPARC Sun Microsystems™, and Sun Workstations™ are trademarks of Sun Microsystems, Inc.

SPARC® is a registered trademark of SPARC International, Inc. SPARC CPU-2CE™ is a trademark of SPARC International, Inc. licensed to FORCE COMPUTERS, Inc.

Solaris® is a registered trademark of Sun Microsystems, Inc.

Motorola® and the Motorola logo are registered trademarks of Motorola, Inc. in the U.S.A. and other countries.

FX Series™ is a trademark of Motorola Computer Group.

AIX®, PowerPC®, RS/6000®, and ARTIC960® are registered trademarks of IBM, Inc.

UNIX® is a registered trademark of UNIX Systems Laboratories, Inc. in the U.S.A. and other countries.

VERITAS®, VERITAS Cluster Server®, VCS®, and the VERITAS logo are registered trademarks of VERITAS Software Corporation in the United States and other countries.

All the brand names and other products or services mentioned in this document are identified by the trademarks or service marks of their respective companies or organizations.

SUCCESSOR IN INTEREST

NewNet Communication Technologies, LLC is the successor in interest to EBS, Inc.; NewNet, Inc.; ADC Enhanced Services Division; ADC ESD, Inc. Any rights or title to the marks or copyrights of these entities, unless otherwise disclosed, are the property of NewNet Communication Technologies, LLC.

NOTICES AND WARRANTY INFORMATION

The information in this document is subject to change without notice and should not be construed as commitment by NewNet Communication Technologies, LLC. NewNet Communication Technologies, LLC assumes no responsibility or makes no warranties for any errors that may appear in this document and disclaims any implied warranty of merchantability or fitness for a particular purpose.

COPYRIGHT INFORMATION

PERFORMANCE SPECIFICATIONS

NewNet Communication Technologies, LLC reserves all the rights to change the equipment performance specifications stated herein at any time without notice. For OEM components, NewNet Communication Technologies, LLC relies on the specifications supplied by the OEM vendors.

ALL RIGHTS RESERVED

NewNet Communication Technologies, LLC
35 Nutmeg Drive
Trumbull, CT 06611
U.S.A.

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

List of Figures

List of Figures

List of Tables

List of Tables

Revision History

The following table lists the revision history of this manual. The Date column shows the date a manual was published or revised. The Associated Software Release column shows the software release number for which the updated manual was published.

Date	Section/ Pages Replaced	Description of Changes	Associated Software Release
2/12/09	Chapter 2	Table 2-7 Host Platform Options: updated for HDC3-LPe board. Updated note following table 2-7.	1.6.0
9/09/08	All	Updated the version and part number.	1.6.0
7/14/08	Chapter 2	Section 2.7.2 edited table 2.7. Added new row to the Sun platform. Added PCIe and the HDCII-LPe board. Section 2.7.3 edited table 2.8. Changed PCI to PCIe.	1.5.0
	Chapter 9	9.4.10 Added note regarding T1/E1 line types not being combined together. 9.4.1, 9.4.10, 9.4.13, and 9.4.17: added note regarding ADAX boards support both low and high speed links on the same card.	
5/23/08	Chapter 2	Section 2.7.3: Table 2-8, added number of ports per controller for PCI cards. Replaced footnote number 11 under Table 2-8.	1.5.0
	Chapter 3	Section 3.3.2.1, Table 3-2, replaced Set Values for Line/Linetype.	
2/26/08	Chapter 2	Section 2.7.2: Added new line item to Table 2.7 Host Platform Options. Section 2.7.3: Added new line item to Table 2.8 SS7 Controller Options.	1.5.0
	Chapter 3	Section 3.3.2: Managed Objects. Added Adaxm to Table 3-2.	
	Chapter 7	Section 7.2.18 SPMD: Replaced a section of the description.	
	Chapter 8	Section 8.2.35 getcfg: Added a new driver and board type.	
	Chapter 9	Section 9.3, table 9-1: added adaxm to the BOARDNM column. Section 9.4.1 Link: Added adaxm to the parameter lists for BOARDNM and PORT. Section 9.4.6 SS7 Board: Added adaxm to the parameter list for BOARDNM. Section 9.4.10 Line: Added adaxm to the parameter list for BOARDNM. Section 9.4.13 Port: Added adaxm to the parameter list for BOARDNM. Section 9.4.17 Time Slot: Added adaxm to the parameter list for BOARDNM. Section 9.4.19 Line Statistics: Added adaxm to the parameter list for BOARDNM. Section 9.4.20 Line History: Added adaxm to the parameter list for BOARDNM.	
1/10/08	All	Updated the company name, logo, address and contact information. Updated the trademark and copyright information.	1.5.0
7/20/07	All	Updated document version for GA release.	1.5.0 GA
	Chapter 9	Updated section 9.4.4 to remove Capability as an rtype value.	
6/15/07	Chapter 3	Updated Table 3-5: MTP Managed Object Descriptions, RTSET	1.5.0 beta
	Chapter 4	Section 4.4.5.3 Recovery Methods Available. Updated the Transaction Abort Policy. Added Section 4.4.8.3 Transaction Recovery APIs.	
	Chapter 6	Updated Figure 6-3: Distributed7 MIB View-MTP Layers.	
	Chapter 9	Updated Table 9-1: MTP Configuration Managed Objects, RTSET Updated section 9.4.4 Route Set (RTSET).	
12/15/06	Chapter 2	Updated Section 2.7.2 and Table 2-7 for x86 and removed references to Sbus. Removed references to Sbus from Section 2.7.3 and Table 2-8.	1.5.0 beta

Revision History

Date	Section/ Pages Replaced	Description of Changes	Associated Software Release
7/31/06	Chapter 3	Section 3.1 Chapter Overview: removed reference to GNU facility.	1.4.0
	Chapter 6	Removed Section 6.4.4 Using the MMI/MML History Buffer. Section 6.4.6 History Facility: emacs history facility replaces GNU history facility.	
	Chapter 7	Section 7.2.13 mml: removed out-of-date entries under Files. Section 7.2.14 mmi: removed out-of-date entries under Files.	
	Chapter 9	Section 9.2.2 Rules for Command Line Syntax: removed reference to defunct History command from line 5. Table 9-3 ISUP Configuration Managed Objects: removed histbuf row under MMLCONF. Removed Section 9.7.6 Configuration: removed histbuf parameter. Removed Section 9.7.10 HISTORY.	
9/27/05	Chapter 2	Updated Table 2-7 Host Platform Options.	1.4.0
	Chapter 4	Section 4.4.1.2 Concurrence Support and Restrictions: updated TCAP capacities.	
	Chapter 6	Figure 6-8: Distributed7 MIB—ISUP Layer: added isupcctCic(11).	
	Chapter 7	Section 7.2.4 AccessSNMP: added -h option to snmp_p.	
	Chapter 8	Table 8-1 User Command Summary: added row for i_trace utility. Updated Section 8.2.35 getcfg. Added Section 8.7 ISUP Utilities.	
	Chapter 9	Section 9.6.1 ISUP Circuits: Table 9-23 ISUP Circuit Display Report: added Circuit Identification Code (cic) column, and revised note following table. Table 9-24 ISUP Circuit Display Values: added Circuit Identification Code (cic) column.	
4/30/05	All	Updated version and date for D7 1.4.0.	1.4.0
1/31/05	Chapter 2	Table 2-6 Standard SS7 Database Capacity: for MTP, "511" replaces "255" in ANSI and ITU "links" columns; for TCAP, "262144" replaces "65536" in ANSI and ITU "simultaneously open dialogues" columns. Table 2-7 Host Platform Options: added Solaris 10 to OS column, for Sun PCI bus, added "PMC4539F" to Board column, and added Solaris 10 bullet after table. Table 2-8 Available SS7 Controller Options: updated for PMC4539F board.	1.4.0 beta
	Chapter 3	Table 3-2 SPM Branch Managed Object Descriptions: for ss7board, boardnm, added "/pmc4539" to Set Values column; for line, boardnm, added "/pmc4539" to Set Values column; for line, line_typ, in Set Values column, "E1HSL/T1HSL/J1HSL/E1LSL/T1LSL/J1LSL" replaces "E1/T1/J1"; for port, boardnm, added "/pmc4539" to Set Values column; for port, baud, added "/1544000/2048000" to Set Values column; for linestat, boardnm, added "/pmc4539", to Set Values column; for linehist, boardnm, added "/pmc4539" to Set Values column. Table 3-5 MTP Managed Object Descriptions: for link, boardnm, added "/pmc4539" to Set Values column.	
	Chapter 7	Section 7.2.5 AccessStatus: added Baud Rate under Display. Section 7.2.18 spmd: added artic 8260 and pmc4539 to Description.	
	Chapter 8	Table 8-6 ebs_showlink Output Description: added artic 8260 and pmc 4539 to Type.	
	Chapter 9	Section 9.4.1 Link (LINK): added pmc4539 board and Sequencing parameter. Section 9.4.6 SS7 Board (SS7BOARD): added pmc 4539 to Boardnm parameter. Section 9.4.10 Line (LINE): added pmc 4539 to Boardnm parameter. Section 9.4.13 Port (PORT): added pmc 4539 to Boardnm parameter.	
	Chapter 11	Glossary: revised first paragraph, and added entries for HSL and LSL.	

Revision History

Date	Section/ Pages Replaced	Description of Changes	Associated Software Release
1/28/04	Chapter 2	Revised Figure 2-6: SMServer Software Architecture. Updated Table 2-7 Host Platform Options.	1.3.1
	Chapter 9	Section 9.6.3 ISUP Signaling Node: added the cfnoff parameter. Section 9.6.4 ISUP Configuration: removed Generic value from ANSI and ITU under the Variant parameter.	
12/19/03	Chapter 2	Table 2-4 Available Host Platform Options: added specs for three boards.	1.3.1 beta
	Chapter 3	Table 3-2 SPM Branch Managed Object Descriptions: added Suspend and Resume to values for Conf parameter in SS7board managed object.	
	Chapter 9	Table 9-1 MTP Configuration Managed Objects: added Suspend and Resume to values for Conf parameter in SS7board managed object.	
6/30/03	Chapter 6	Added Section 6.14 CompactPCI Hot-swap (CR14686).	1.3.1 beta
	Chapter 9	Section 9.4.6: added the two values, "suspend" and "resume", to the conf parameter of the SS7BOARD managed object (CR14686). Section 9.5.5: added the value "Redknee" to the protocol parameter of the SCCP command (CR14671).	

Revision History

Date	Section/ Pages Replaced	Description of Changes	Associated Software Release
11/14/02	Chapter 2	Section 2.4.16 JAIN TCAP API Library — Added description for JAIN TCAP API Library	1.3.0
		Section 2.4.17 JAIN ISUP API Library — Added description for JAIN ISUP API Library	
		Table 2-2 Distributed7 Wireless Standards Compliance — Updated document number to ETSI 09.02 version 7.3.0	
		Section 2.7.3 SS7 Controller Options — Updated footnote 9 to include ARTIC1000/2000 boards	
	Chapter 3	Section 3.2.3.1 MTP Database — Added new paragraph to MTP Database Linksets and Links	
		Figure 3-9 SPM Managed Object Containment Structure — Updated graphic to include ***ctbus*** and ***pmlink*** parts	
		Table 3-2 SPM Branch Managed Object Descriptions — Added line_accs parameter to ***line*** object; added artic8260 board set value to all boardnum parameters; added ***ctbus*** object; added ***pmlink*** object	
		Table 3-5 MTP Managed Object Descriptions — Added L2ECM, PCRN1 and PCRN2 to ***mtp*** object values; added artic8260 board set value to link boardnum parameter	
		Table 3-6 SCCP Branch Managed Object Descriptions — Updated ***gtentry*** and ***gt*** managed objects	
		Section 3.6.4.1 Flexibility — Added bullet item	
	Chapter 4	Section 4.4.9 JAIN TCAP API Support — New section added	
		Section 4.5.1 Multiple Instance Support — Modified Call Control information in this section	
		Section 4.5.8 JAIN ISUP API Support — New section added	
		Figure 4-4 Flow of ISUP Messages Received From MTP/L3 — Updated graphic	
	Chapter 6	Figure 6-5 Distributed7 MIB View (MTP Layer) — Updated to include new link parameters. L2ECM, PCRN1 and PCRN2	
		Figure 6-10 Distributed7 MIB View (Hardware Layer) — Added pmlink and ctbusTable branches from ss7board	
	Chapter 7	Section 7.2.4 AccessSNMP — Updated daemon	
		Section 7.2.15 netd — Updated ***netd*** daemon, which now takes '-i' command-line option for Solaris IP network multipathing (IPMP) support	
		Section 7.2.18 spmd — Added artic8260 to the list of boards in the third paragraph of DESCRIPTION	
	Chapter 8	Artic8260 board references added throughout chapter (Sections 8.2.3, 8.2.4, 8.2.6, 8.2.7, 8.2.35 and Table 8-6)	
		Table 8-1 User Command Summary — Added ***ebs_dbconfig*** to table	
		Section 8.2.5 ebs_dbconfig — Added new section	
		Section 8.2.32 ebs_sysinfo — Updated ***ebs_sysinfo*** utility, which now provides '-l' command-line option	
	Chapter 9	Table 9-1 MTP Configuration Managed Objects — Modified LINK object with new parameters L2ECM, PCRN1 and PCRN2; added CTBUS object	
		Table 9-5 Passive Monitor Managed Object — Added new table	
		Section 9.4.1 Link (LINK) — Modified to include L2ECM, PCRN1 and PCRN2	

Revision History

Date	Section/ Pages Replaced	Description of Changes	Associated Software Release
11/14/02	Chapter 9	Section 9.5.1 Concerned Point Code (CPC) — Edited commands and errors	1.3.0
		Section 9.4.6 SS7Board (SS7BOARD) — Added ***pm*** to the parameters section of command description; updated commands, parameters, errors and examples for CT BUS and artic8260	
		Section 9.4.17 Time Slot (TIMESLOT) — Updated commands, parameters, errors and examples for CT BUS and artic8260	
		Section 9.4.21 CT Bus (CTBUS) — Added new section	
		Section 9.5.2 Global Title (GT) — Edited names, commands, parameters, errors and examples	
		Section 9.5.3 Global Title Entry (GTENTRY) — Edited commands, parameters, errors and examples	
		Section 9.5.4 Mate — Edited commands and errors	
		Section 9.5.5 SCCP — Edited commands	
		Section 9.5.6 SCCP Signalling Point — Edited commands and errors	
		Section 9.5.7 Subsystem — Edited commands and errors	
		Section 9.8 Passive Monitor MML Commands — Added new section	
04/19/02	Chapter 1	Section 1.2.2 Related Documents — Added bullets for ITU White Book (1993) Q.701-Q.707, ITU (1997) Q.704 and ANSI T1.111.4, 1996	1.2.0
	Chapter 2	Section 2.3.3.9 Application Programming Guides — Added bullet for ISUP - Advice of Charge (AoC) API	
		Section 2.4.5 Alarm API Library — Added link and description for Alarm API Library	
		Section 2.4.11 ISUP AoC API Library — Added link and description for ISUP - Advice of Charge (AoC) API Library	
		Table 2-1 Distributed7 Standards Compliance — Revised ANSI and ITU standards	
		Table 2-4 Available Host Platform Options — Updated table to include new Sun model	
		Table 2-5 Available SS7 Controller Options — Added CompactPCI card information	
		Table 2-7 Library External Dependencies — Deleted redundant liboam entry	
	Chapter 3	Section 3.2.2.1 SEP-STP Configuration — Changed board capacity from 24 to 64 links	
		Section 3.2.3.1 MTP Database — Added ITU_97 variant to subsection 1 Signalling System and MTP Information and changed maximum number of linksets from 16 to 128 in subsection 3 Linksets and Links	
		Figure 3-9 SPM Managed Object Containment Structure — Updated figure with added ***linehist*** and ***linestat*** branches	
		Table 3-1 SPM Branch Managed Object Descriptions — Added ***linehist*** and ***linestat*** objects to descriptions; updated ***ss7board***, ***line***, ***port*** and ***timeslot*** objects	
		Table 3-2 NETWORK (NTWK) Branch Managed Object Descriptions — Modified ***ntwk*** and ***host*** objects	
		Table 3-3 ALARM Branch Managed Object Descriptions — Modified ***tcpcon, alarm*** and ***strdalm*** objects; added ***almevent*** object	
		Table 3-4 MTP Managed Object Descriptions — Added ITU_97, ETSI97 and BELL to ***mtp*** object values; deleted ltt parameter from ***sp*** object; added CompactPCI cards (cpc3xpq and pcm8260) to ***link*** object values	

Revision History

Date	Section/ Pages Replaced	Description of Changes	Associated Software Release
		Table 3-5 SCCP Branch Managed Object Descriptions — Added ***sccp*** object and edited typo in ***localsubsys*** object name	
		Section 3.6 Product Specifications — Edited paragraph text for this release	
		Section 3.6.4.3 Scalability — Edited paragraph text for this release	
		Section 3.6.4.4 Transparency — Edited paragraph text for this release	
	Chapter 4	Section 4.2.3.1 MTP Capacity and Protocols — Added ANSI 96, ITU 1997 and ETSI 1997 variants bullets to list; changed maximum number of linksets from 16 to 128, max. number of destinations per sp and destinations behind a linkset from 355 to 2048	
		Section 4.5.6 Protocol Specific Issues — Added bullet for Czech ISUP	

Revision History

Date	Section/ Pages Replaced	Description of Changes	Associated Software Release
04/19/02	Chapter 6	Updated MML (Man-Machine Language) to MMI/MML throughout chapter	1.2.0
		Section 6.2.2.1 New Installation Distributed7 Start-up — Changed path in Step 6(c) from $EBSHOME/access/lib/tcl_lib to $EBSHOME/access/lib/tk_lib	
		Section 6.4 Using MMI/MML — Updated section header and revised paragraph text	
		Section 6.4.3 Logging MMI/MML Commands — Changed RUN0 to RUN* in file path	
		Section 6.11.9.2 Display Only Alarms with Certain Severities — Added FATAL to list of alarm severities in paragraph reference and added bullets for MTP Levels 1 and 2 and APM; updated example text	
		Section 6.5.5.2 Alarm Reporting with SNMP Agent (Traps) — Updated alarm list	
	Chapter 7	Table 7.1 Daemon Summary — Added MMI (Man-Machine Language Interface) to list	
		Section 7.2.1 AccessAlarm — Modified synopsis, description (***-d dir***, ***-n nfile***, ***-m msize***) and references	
		Section 7.2.12 mlogd — Modified synopsis, description (***-d dir***, ***-a asize***, ***-m msize***), notes and references	
		Section 7.2.13 mml — Updated section with new mmi information	
		Section 7.2.14 mmi — New section added	
		Section 7.2.18 spmd — Modified description (adding boards) and references	
		Section 7.3.1 apmconfig — Modified description (***action, dirpath, priclass*** and ***clparams***), sample file and references	
	Chapter 8	Table 8-1 User Command Summary — Added ***ebs_alarm*** and ***apm_trcapture*** commands to list	
		Section 8.2.1 ebs_alarm — New section added	
		Section 8.2.3 ebs_brdinfo — Modified description with new boards (CPC37xPQ and PMC8260)	
		Section 8.2.4 ebs_cfgbrd — Modified description with new boards (CPC37xPQ and PMC8260)	
		Section 8.2.5 ebs_dnlbrd — Modified description with new boards (CPC37xPQ and PMC8260)	
		Section 8.2.6 ebs_mngbrd — Modified description with new boards (CPC37xPQ and PMC8260)	
		Section 8.2.24 ebs_report — Modified synopsis and description (***-d dir***)	
		Table 8-6 ebs_showlink Output Description — Modified type values with new boards (CPC37xPQ and PMC8260)	
		Section 8.2.34 getcfg — Modified description and sample output with new board (CPC37xPQ and PMC8260) info; ; fixed PCI370APQ; added boards to Note text	
		Section 8.3.6 apm_report — Modified synopsis and description (***-d dir***)	
		Section 8.3.10 apm_trcapture — New section added	
		Section 8.3.11 apm_trclear — Modified references	
		Section 8.3.12 apm_trgetmask — Modified synopsis, description and references	
		Section 8.3.13 apm_trinit — Modified references	
		Section 8.3.14 apm_trsetmask — Modified synopsis, description and references	
		Section 8.3.15 apm_trshow — Modified synopsis, description and references	
		Section 8.7.16 db2text — Deleted text from description and notes regarding database files	

Revision History

Date	Section/ Pages Replaced	Description of Changes	Associated Software Release
04/19/02	Chapter 9	Table 9-1 MTP Configuration Managed Objects — Modified LINK object SLC, Priority, Boardnm, INST and Port parameters and added HOSTSTATUS parameter; modified LSET object Loaded and Active parameters and added EMERGENCY parameter; modified MTP object Protocol, Variant and RESTART parameters and added new parameters; modified Boardnm, ports and clock parameters for SS7BOARD and added new parameters; modified Span, Boardnm, INST and LINE_LEN parameters for LINE object and added LINE_TYP parameter; modified Boardnm, type, INST, LPMODE and portnum parameters for PORT object and added CLASS parameter; modified Boardnm, INST, desttype, destslot, origtype and origslot parameters for TIMESLOT object and added destspan, class and origspan parameters; modified ROUTE object Priority parameter and added new parameters; modified L3TIMER object Timer parameter; modified RTSET object; added LINESTAT and LINEHIST objects	1.2.0
		Table 9-2 SCCP Configuration Managed Objects — Added SPNO to SCCP object	
		Table 9-3 ISUP Configuration Managed Objects — Added Czech variant to range for ITU	
		Table 9-4 System Configuration Managed Objects — Modified STRDALM object; modified ALARM object CONS_THRS and USER_THRS parameters (deleting "none" value and adding "fatal"); modified ALMEVENT object Threshold parameter (deleting "none" value and adding "fatal") and updated GROUP and MODULE values; modified ALMGRP object Group parameter (adding "MTPL1" and "MTPL2" values, deleting "SNM" and adding "fatal") and CONS_THRS and USER_THRS parameters (deleting "none" value and adding "fatal"); modified MMLCONF object; modified NTWK object; modified TCPCON object; modified SET-LOG object	
		Section 9.4.1 Link (LINK) — Modified SLC, Priority, Boardnm and Port parameters; added boards to Note text	
		Section 9.4.2 Link Set (LSET) — Modified Loaded and Active parameters	
		Section 9.4.3 Message Transfer Part (MTP) — Modified Protocol, Variant and Restart parameters	
		Section 9.4.5 Route (ROUTE) — Modified Priority parameter from 0-3 to 0-7	
		Section 9.4.6 SS7Board (SS7BOARD) — Modified Boardnm and Clock parameters; added clockspan parameter; added boards to Note text; updated sample output, example and errors	
		Section 9.4.9 Level 3 Timer (L3TIMER) — Modified Timer parameter	
		Table 9-8 SS7BOARD Display Values — Added class IV boards to list	
		Table 9-11 MTP-L3 Timer Definitions — Added Timer 31 and modified footnote 3 to include ITU 1997	
		Section 9.4.10 Line (LINE) — Modified Span, Boardnm and Line_len parameters; added line_typ parameter; added boards to Note text; updated sample output and examples	
		Table 9-12 LINE Display Values — Added class IV boards to list	
		Section 9.4.13 Port (PORT) — Modified Boardnm and portnum parameters; added boards to Note text	
		Section 9.4.14 MTP SLTM Timer (SLTIMER) — Modified footnote 3 to include ITU 1997	
		Section 9.4.17 Time Slot (TIMESLOT) — Modified Boardnm, desttype, destslot, origtype and origslot parameters and added destspan and origspan parameters; updated sample output; added boards to Note text	
		Table 9-16 TIMESLOT Display Values — Added class IV boards to list	

Revision History

Date	Section/ Pages Replaced	Description of Changes	Associated Software Release
04/19/02	Chapter 9	Section 9.4.19 Line Statistics — Added MML Command information for LINESTAT, including new Table 9-17 LINESTAT Display Values	1.2.0
		Section 9.4.20 Line 24-Hour Performance Data — Added MML Command information for LINEHIST, including new Table 9-18 LINEHIST Display Values	
		Section 9.6.4 ISUP Configuration — Added Czech variant to values for ITU	
		Table 9-31 ISUP Timers — Added Trunk Offering Timer to table	
		Section 9.7.2 Stored Alarm (STRDALM) — Modified Last_occur parameter and corrected command name from "Standard Alarm" to Stored Alarm"; updated display values; modified Group parameter (adding "MTPL1" and "MTPL2" and "APM" and "ETMOD" values)	
		Section 9.7.3 Alarm — Modified CONS_THRS and USER_THRS parameters (adding "fatal" value); modified ***update*** description	
		Section 9.7.4 Alarm Event — Modified Threshold parameter (deleting "none" value)	
		Section 9.7.5 Alarm Group — Modified Group parameter (adding "MTPL1" and "MTPL2" and "APM" and "ETMOD" values) and Severity parameter (deleting "none" value); updated note text	
10/29/01	Chapter 2	Table 2-1 Distributed7 Standards Compliance — Updated table to reflect ANSI 96 and ITU 97 support Table 2-3 Standard SS7 Database Capacity — Updated ISUP and TCAP capacities Table 2-5 Available SS7 Controller Options — Changed the number of ports per controller available to T1 and E1 interfaces; added alert on maximum number of links usable under load	1.1.0
10/29/01	Chapter 5	Section 5.2.2.1 New Installation Distributed Software — Added procedure on setting environment variables	1.1.0
10/29/01	Chapter 8	Table 8-6 ebs_showlink Output Description — Added pci3xapq card as a valid value for the TYPE output Section 8.2.33 getcfg — Added PCI3xAPQ board to *driver* and *board type*; added alert on maximum number of links usable under load	1.1.0

Revision History

Date	Section/ Pages Replaced	Description of Changes	Associated Software Release
10/29/01	Chapter 9	Table 9-1 MTO Configuration Managed Objects — Added PCI3xAPQ card to LINK, SS7BOARD, LINE, PORT, and TIMESLOT MOs; changed Class II values for TIMESLOT MO; added NONE as a possible value for CONS_THRS and USER_THRS parameters of the ALARM and ALMGRP MOs, and for the THRESHOLD parameter of the ALMEVENT MO Table 9-3 ISUP Configuration Managed Objects — Updated valid range for parameters of the ISUPCCT, ISUPGRP, and ISPNODE managed objects; added LOCATION, MAXCCT, and FIRSTCIC parameters to the ISUPNODE managed object; added AUTORESP, EXCHODC, and UPMIND parameters to the ISUP managed object Sections 9.4.1 Link, 9.4.6 SS7Board, 9.4.10 Line, 9.4.13 Port, and 9.4.17 Time Slot — Added pci3xapq to *boardnm* parameter; added alert on maximum number of links usable under load Section 9.4.16 Alias Point Code (ALIAS) — Added alert for ADD, MODIFY, and DELETE operations Section 9.6.1 ISUP Circuits — Updated descriptions of *cctnum* and *range* parameters Section 9.6.2 ISUP Circuit Group — Changed valid values for *cctnum* and *trnkgrpid* parameters Section 9.6.3 ISUP Signaling Node — Changed valid values for pcno parameter; removed 14-bit length indication of point code from *dpc* and *newdpc* parameters Sections 9.7.3 ALARM, 9.7.4 ALMEVENT, and 9.7.5 ALMGROUP — Added *NONE* as a possible alarm value	1.1.0
09/27/01	Chapter 1	Section 1.2.1 Related Documents — Updated citations to reflect support for ANSI 96, ITU 97, Bellcore 91, and ETSI 97 protocols	1.1.0 B
09/27/01	Chapter 2	Table 2-3 SS7 Database Capacity — Changed MTP link set capacity from 32 to 64	1.1.0 B
09/27/01	Chapter 3	Table 3-4 MTP Managed Object Description — Added *emergency* parameter to the ***lset*** MO; added *nicheck* and *dpccheck* parameters to the ***mtp*** MO	1.1.0 B
09/27/01	Chapter 4‘	Section 4.2.3.0 MTP Capacity and Protocols — Changed link set and links per SP capacities	1.1.0 B
09/27/01	Chapter 6	Section 6.5 Using SNMP — Updated MIB diagrams	1.1.0 B
09/27/01	Chapter 9	Whole chapter — Extensively edited Section 9.4.2 Link Set — Added EMERGENCY parameter Section 9.4.3 MTP — Added AT&T variant support for ANSI92 Section 9.6.2 ISUP Circuit Group — Changed the range of valid values for GRPID and TRNKID parameters Section 9.6.3 ISUP Node — Added LOCATION, MAXCCT, and FIRSTCIC parameters; changed the range of valid values for PCNO parameter Section 9.6.4 ISUP Configuration — Added ANSI96, DSC, ITU97, Q767, ETSI97, and country variants to Variant parameter; added AUTORES, EXCHODC, and UPMIND parameters Table 9-3 ISUP Configuration Managed Objects — Added country variants to Range of ITU values for ISUP MO Table 9-29 ISUP Timers — Added Timer IDs 51-56 for FINLAND and MEXICO	1.1.0 B
05/15/01	Chapter 1	Whole chapter — Extensively edited; no change bars indicated for line edits	1.0.5
05/15/01	Chapter 2	Whole chapter — Extensively edited; no change bars indicated for line edits	1.0.5

Revision History

Date	Section/ Pages Replaced	Description of Changes	Associated Software Release
05/15/01	Chapter 3	Whole chapter — Extensively edited; no change bars indicated for line edits Figure 3-4 Object Server Internal Architecture — Updated diagram by removing CMIP from Man-Machine Network Agents	1.0.5
05/15/01	Chapter 9	Section 9.5.2 Global Title — Corrected EXAMPLES by removing IO parameter from DISPLAY command	1.0.5
04/16/01	Chapter 2	Figure 2-2 Distributed7 Software Architecture — Updated graphic Table 2-3 Standard SS7 Database Capacity — Updated maximum number of SCCP destination point codes Table 2-4 Available Host Platforms — Updated Development Environment column Sections 2.4.9 Raw TCAP API Library — Added section Section 2.4.11 Gateway API Library — Added section Section 2.4.12 IS41-D API Library — Added section Section 2.4.13 GSM MAP — Added section	1.0.5 B
04/16/01	Chapter 3	Figure 3-5 Managed Object Containment Structure — Added "alias" under mtp Table 3-4 MTP Managed Object Descriptions — Added "alias" managed object	1.0.5 B
04/16/01	Chapter 6	Figure 6-3 Distributed7 MIB View (MTP Layers) — Added "alias" managed object Section 6.5.5.3 Adding New Managed Object Definitions — Under Command Table File, added "aliasEntry, ALIAS;" for SNPM requests regarding MTP-L3 managed objects Section 6.11.1.2 Adding Alias Point Code — Added section Section 6.11.4.1 Modifying — Added Alias Point Code description Section 6.11.4.2 Deleting — Added Alias Doing Code description	1.0.5 B
04/16/01	Chapter 7	Section 7.2.15 omapd — Added entire section describing the AccessOMAP daemon process Section 7.2.19 upmd — Added $EBSHOME/access/RUN<sp#>/Dbfiles/mtp_alias.DB file	1.0.5 B
04/16/01	Chapter 9	Section 9.4.4 Route Set — Added "restricted" state to the STATE parameter of the MODIFY command Section 9.4.15 Signaling Point — Updated sample output Section 9.4.16 Alias Point Code — Added section Section 9.5.5 SCCP — Added section Section 9.7.4 Alarm Event — Added section	1.0.5 B
04/16/01	Chapter 10	Section 10.3 Driver Installation/Removal — Removed section Section 10.5.2 Link Creation/Activation - vbrd driver vs. Actual SS7 Card — Removed second sentence of first paragraph that described hardware specifications not used by the *vbrd* driver	1.0.5 B
01/15/01	Chapter 9	Section 9.6.4 ISUP Configuration — Removed JAPAN and JAPAN_CTM variants (not supported).	1.0.4
9/27/00	Chapter 2	Section 2.7.2 Host Platform Options — Updated table 2-4 Available Host Platform Options; added Solaris 2.7 and Solaris 2.8 to the O/S column; replaced SPARCompiler with SunWorkshop 4.2 or 5.0.	1.0.2
9/27/00	Chapter 8	Section 8.2.19 ebs_pkgrm MML added to the Platform Utilities	1.0.2
08/04/00	Chapter 5	Section 5.4.10: ISUP API Library — added Table 5-3: Standard SS7 Database Capacity — ISUP material added	1.0.1 B

Revision History

Date	Section/ Pages Replaced	Description of Changes	Associated Software Release
08/04/00	Chapter 6	Figure 6-7: ISUP Managed Object Containment Structure — added Table 6.6: ISUP Branch Managed Object Descriptions — added Section 6.6: Product Specifications, ISDN User Part (ISUP) bullet item added Section 6.6.4: ISUP Product Specifications — added	1.0.1 B
08/04/00	Chapter 7	Section 7.1:Introduction, ISDN User Part bullet item added Section 7.5: ISDN User Part (ISUP) — added	1.0.1 B
08/04/00	Chapter 8	Section 8.6::ISUP MMS Commands — added	1.0.1 B
08/04/00	Chapter 9	Section 9.11.3: Configuring ISUP — added Section 9.11.5.1: Modifying, ISUP Nodes, and ISUP Circuits — added Section 9.11.5.2: Deleting, ISUP Circuits, ISUP Circuit Groups, and ISUP nodes — added Section 9.11.7: Changing Timers, added line "ISUP-related timers are changed with the MODIFY-ISUPMR MML command (Section 9.6.5 on page 9-72) Section 9.11.9:Changing General ISUP Settings — added Section 9.11.11.2: Display Only Alarms with Severities, ISUP - ISUP management (ANSI and ITU/CCITT and ISUPMOD - ISUP management bullet items added	1.0.1 B
06/19/00	Full Manual	General Availability release	1.0.0
03/17/00	Full Manual	Beta release	1.0.0 B

Chapter 1: Introduction

1.1 General

Distributed7 software is used for control and configuration of Signaling System 7 (SS7) network nodes and devices. The Distributed7 software and supporting hardware components are designed, assembled, and shipped by NewNet Communication Technologies, LLC.

Distributed7 features a UNIX Streams-based Application Programming Interface (API) that gives customers the flexibility to develop SS7 User Parts or Applications using TCAP/ SCCP or message transfer part (MTP) services. Distributed7 also provides efficient, connectionless Streams-based inter-process communication (IPC) capabilities among multiple processes comprising your application.

1.2 Scope

This user manual forms part of a complete documentation package that describes the NewNet Communication Technologies, LLC Distributed7 software. This manual is organized so that all the information is presented in a logical sequence. It consists of ten chapters, covering the following topics:

- ***Chapter 1: Introduction*** - provides an introduction to the manual and NewNet Communication Technologies, LLC documentation.
- ***Chapter 2: Distributed7 Overview*** - provides an overview of the Distributed7 software platform.
- ***Chapter 3: Concepts*** - provides a general overview of SS7, the Distributed7 managed object concept, and product specifications.
- ***Chapter 4: Distributed System Operations*** - provides information for the distributed operations of MTP, SCCP, TCAP, and ISUP layers.
- ***Chapter 5: User/Kernel-space Data Distribution Methods*** - provides a description of the functionality of User-Space and Kernal-Space with distributed operations.
- ***Chapter 6: Operations*** - provides detailed coverage of using the Distributed7 software, the Managed Object Browser, and the Command File Navigator.
- ***Chapter 7: System Processes*** - provides descriptions of the Distributed7 system processes (daemons) and their configuration files.
- ***Chapter 8: User Commands*** - provides descriptions of the Distributed7 platform service provider module (SPM), application process manager (APM), distributed shared memory

(DSM), distributed kernel memory (DKM), Transaction Capabilities Application Part (TCAP), and virtual board (VB) user commands.

- ***Chapter 9: Man-Machine Language Commands*** - describes Distributed7 Man-Machine Language (MML), the terminal handler, the rules and conventions for using MML, and the MML commands.
- ***Chapter 10: Users Guide for Virtual SS7 Connections*** - describes the use of the Virtual Board and virtual connections.

It is assumed that the hardware and software are installed, and the user is familiar with the SS7 protocol. The SS7 protocol and network architecture are covered in NewNet Communication Technologies, LLC training courses.

1.2.1 Revisions and Updates

NewNet Communication Technologies, LLC seeks to provide total quality and customer satisfaction through continuous improvement. Toward that goal, revisions to the manual occur from time to time due to software enhancements or documentation enhancements. The revisions are in change packets that contain only the pages that have modifications. Change packets are automatically sent with software fix packages.

NewNet Communication Technologies, LLC requests that you register your manuals for update notices. Customers may register their manual ownership via the NewNet Communication Technologies, LLC Web site. Please include the documentation number, your name, address, and e-mail address.

1.2.2 Related Documents

This section lists the related documents or materials that are beyond the scope of the Enhanced Services Division of manuals provided with your system. The cited materials contain additional information that may be helpful to the user:

- CCITT Blue Book (1988) Q.701 - Q.707
- CCITT Blue Book (1988) Q.711 - Q.714
- CCITT Blue Book (1988) Q.721 - Q.724
- CCITT Blue Book (1988) Q.761 - Q.764
- CCITT Blue Book (1988) Q.771 - Q.775
- ITU White Book (1993) Q.711 - Q.714
- ITU White Book (1992) Q.730 group
- ITU White Book (1992) Q.761 - Q.764
- ITU-T Recommendation (1996) Q.711 - Q.714
- ITU White Book (1993) Q.771 - Q.775
- ITU White Book (1993) Q.701 - Q.707
- ITU (1997) Q.704
- ITU (1997) Q.771 - Q.775
- Bellcore Computer Based Training - SS7: A Brief Look, CCS/SS7 Reference
- ANSI T1.111.3, 1992
- ANSI T1.111.4, 1992
- ANSI T1.112.x, 1992
- ANSI T1.113.x, 1992
- ANSI T1.114.x, 1992
- ANSI T1.112.x, 1996
- ANSI T1.114.x 1996
- ANSI T1.111.4, 1996
- EIA/TIA/IS-41.1B-41.5B

1.3 How to Use this Manual

The *Distributed7 User Manual* is part of a complete family of publications that describe the Distributed7 products. These manuals are published separately and bound together in two 3-ring binders for your convenience. To find the information you need, refer to the list above and the Table of Contents or Index of this or any other NewNet Communication Technologies, LLC manual.

The *Distributed7 User Manual* is intended for use during operation of your system. This manual, together with the additional documentation shipped with your unit and documents referenced herein, is required for proper operation of your system.

1.3.3 Notations and Conventions

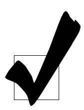
This section describes the notations and conventions that are used consistently throughout this manual.

1.3.3.1 Acronyms and Mnemonics

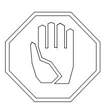
Acronyms and mnemonics are introduced in this manual at first usage as follows "*... the Integrated Services Digital Network (ISDN);*" the acronym or mnemonic is used thereafter throughout the related manual without further introduction. Additionally, each acronym used in this manual is referenced in the Index, and is listed together with its introduction in the Glossary.

1.3.3.2 Alert Messages

ANSI A535 specifications define specific words and icons that alert the reader to dangerous situations. The situations may result in injury to a person or the equipment. The ANSI A535 specifications have been adapted for use with NewNet Communication Technologies, LLC software.



Important: *Recommendations, guidance, hints, tips, and shortcuts to alert readers to situations and procedures that, if not followed properly, may complicate or prevent proper operation of the software.*



Notice: *Situations that, if not avoided, may cause damage to the equipment.*

Caution: *Situations that, if not avoided, may corrupt the software or stop it entirely.*

This page is intentionally blank.

Chapter 2: Distributed7 Overview

2.1 Chapter Overview

This chapter provides an overview of the Distributed7 software platform.

2.2 General Description

Distributed7 is an open-architecture, real-time, scalable, reliable, and high-performance telecommunications application development platform that provides a rapid development and deployment environment for service providers in the telecommunications domain. Distributed7 provides value-added application components on open-architecture computer platforms, and integrates industry standard boards into computers with standard backplanes.

The Distributed7 platform is a collection of telecommunications software building blocks such as SS7 (MTP, SCCP, TCAP, ISUP), IS-41, GSM-MAP Interface, and GSM A-Interface. The building blocks are implemented on industry-standard, open-architecture platforms and the UNIX operating system. The platform frequently takes advantage of UNIX STREAMS to provide a truly layered software architecture, modularity, and performance (See Figure 2-1).

Using a fast-packet switch software backplane implemented in UNIX STREAMS, the Distributed7 software also provides Inter-Process Communications (IPC) and extended timer facilities essential for telecommunications applications. The services of Distributed7 are available to applications through dynamic binding and a series of Applications Programming Interface (API) library calls. Consistent with its object-oriented architecture and rapid, simple application development philosophy, Distributed7 supports protocol-related communications and IPC on the same application interface.

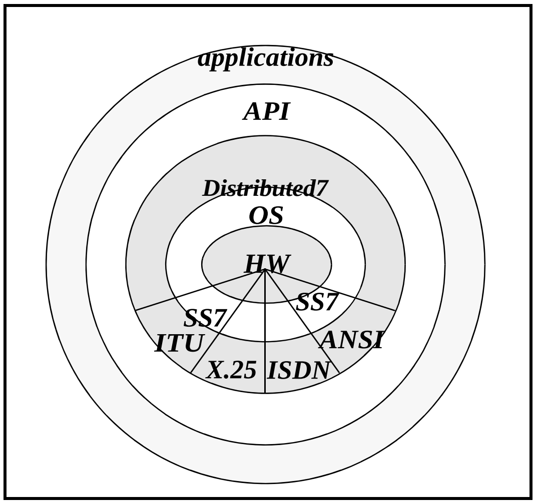


Figure 2-1: Distributed7 Layered Architecture

2.3 Features

2.3.1 UNIX Open-Architecture

The Distributed7 software runs on open-architecture UNIX platforms and takes advantage of the UNIX STREAMS framework and symmetric multiprocessor (SMP) capabilities.

2.3.2 Common Channel Signalling System No. 7

The Distributed7 software environment provides the building blocks and development tools needed for the dynamic creation and support of SS7 network signalling applications. All layers of the SS7 protocol stack are fully integrated, and are available as modules that can be added on as needed.

2.3.2.1 Message Transfer Part

The Distributed7/MTP provides the signalling data link functions and procedures related to reliable, real-time message routing and signalling network management. The Distributed7/MTP supports all signalling end point (SEP) and signalling transfer point (STP) procedures, and comes with an Application Programming Interface (API) accessible with C or C++ programming languages.

2.3.2.2 Signalling Connection Control Part

The Distributed7/SCCP enhances the services of the MTP to provide immediate connectionless network services and address translation capabilities for advanced voice, data, ISDN, and cellular services. The Distributed7/SCCP supports Class 0 and Class 1 connectionless services, Class 2 connection-oriented services, global title translation, and subsystem management. It comes with an API accessible with C or C++ programming languages.

2.3.2.3 Transaction Capabilities Application Part

The Distributed7/TCAP provides the capability for a large variety of distributed applications to invoke procedures at remote locations distributed across the SS7 network. The Distributed7/TCAP supports transaction- and component-handling capabilities. It also supports a load sharing feature between multiple instances of the same application. The Distributed7/TCAP services are available by means of an API accessible with C or C++ programming languages.

Distributed7 allows TCAP applications to select between SCCP and TCP/IP transport service providers when using the TCAP protocol. It also supports TCP/IP connectivity to third-party hosts, i.e., hosts that are not equipped with the Distributed7 software.

2.3.2.4 ISDN User Part

The Distributed7/ISUP provides control of circuit-switched network connections including basic voice, data, and supplementary services such as calling line identification, closed user groups, and user-to-user signalling. The Distributed7/ISUP supports all signalling procedures for call processing and circuit maintenance and recovery, and comes with easy-to-use call control and maintenance APIs accessible with C or C++ programming languages.

2.3.2.5 Operations, Maintenance, and Application Part

The Distributed7/OMAP provides interactive testing and maintenance functions to monitor, control, and coordinate resources through the layers of the SS7 protocol. The Distributed7/

OMAP comes with an Application Programming Interface (API) accessible with C or C++ programming languages.

2.3.2.6 Distributed SS7 Stack Operations

The Distributed7 software provides the functionality for distributed systems operations in the MTP, SCCP, TCAP, and ISUP Layers. See Chapter 4: Distributed System Operations for more information.

2.3.3 Platform Services

2.3.3.1 Core Capabilities

The Distributed7 SPM library provides application programs executing under a distributed environment with a set of basic functions to:

- register/deregister with the Distributed7 environment
- retrieve information about other applications
- send/receive IPC and/or SS7 messages
- detect various internal events and perform asynchronous I/O processing
- report/analyze error conditions that may be encountered during operation
- deactivate/activate debug features such as message logging and/or loopback

As part of the distributed environment, the Distributed7 SPM library also includes such functionality as:

- exclusive registration capability
- local vs. network-based registration
- multiple instantiations of an object
- load-sharing among multiple instances of an object
- active vs. standby mode of operation
- distributed IPC/SS7 messaging
- message broadcast capability
- normal vs. expedited [out-of-band] messaging
- selective message retrieval capability
- extended internal event management mechanism
- improved message logging and loopback capabilities

For more details on these capabilities and how application programmers can take advantage of them, refer to the Chapter 2: SPM API Programming Guide of the *Application Development Manual*.

Registration

In the Distributed7 environment, applications establish a service endpoint through a selected module and bind an address to that service endpoint—a step called *registration*. This step allows object-oriented dynamic binding of applications to the environment, creating the opportunity to seamlessly add new services or replace old services with enhanced versions.

Distributed7 supports *named object* and *SS7 object* registration, exclusive vs. non-exclusive registration, and local vs. global (network-based) registration. This release of Distributed7 also supports other object categories in addition to the named object and SS7 object categories. All these capabilities help better classify different types of system/application programs running under a distributed environment. More information about the specifics of these capabilities can be found in the Chapter 2: SPM API Programming Guide of the *Application Development Manual*.

Messaging

Distributed7 allows applications operating under a distributed environment to exchange inter-process communication (IPC) messages in a location-transparent manner. That is, when an application sends messages to another application, it need not specify the host on which the receiving application is running. In case the receiving application features multiple instances, the system uses a built-in algorithm to load-share successive messages among active instances of the object. Capabilities are also provided to bypass the built-in load-share algorithm and send messages to a designated instance at all times.

In addition, Distributed7 features a multitude of new messaging related capabilities. These capabilities include broadcasting messages to multiple instances of an object, forwarding an IPC/SS7 message to a specified object, prioritizing messages being sent by an object, i.e., normal vs. out-of-band or IPC vs. SS7 messages, and the ability to retrieve messages waiting on queue in a somewhat selective manner, e.g., based on priority-band information.

Timer Services

In the telecommunications domain, timers are important. Distributed7 extends the standard UNIX timer services to provide a real-time, high-resolution timer handling facility. It is based on a deferred message delivery mechanism that allows multiple timers to be simultaneously active. The deferred messages are delivered to the applications through the normal Distributed7 application interface. Distributed7 supports deferred message delivery for both IPC and SS7 messages.

Event Management

Distributed7 features a powerful set of event triggering and notification tools. Using events, applications running under a distributed environment can be informed asynchronously about on-going activities in the system. In addition to the standard STREAMS events, Distributed7 supports a new set of non-STREAMS events that allows different types of activities on the system to be detected and acted upon. Examples include start-up/termination of specified system/application processes on local/remote hosts, connection/teardown of TCP/IP connections to a specified remote host, and attachment/detachment of SS7 signalling hardware on a specified host. Lastly, applications can communicate with each other using user-defined events, which can be viewed as an extension of the UNIX signalling mechanism.

Logging Services

Distributed7 allows messages injected by an application vs. messages destined to that application to be logged in an independent manner. Furthermore, capabilities are provided to log messages injected vs. messages delivered by/to an application at user-specified destinations, which may be different from the standard log daemon process.

The message log daemon (***logd***) that is available as part of the Distributed7 product is now capable of providing detailed information about the whereabouts of a message being logged, i.e., whether the message was travelling in the upstream or downstream direction, as well as at which STREAMS multiplexer it is being logged. All these capabilities hold more thorough debugging sessions.

Loopback Services

Distributed7 supports a built-in loopback feature that causes an application's message traffic to be diverted to a selectable address, such as a screen or a file, providing powerful monitoring and validation capabilities. This feature can be activated dynamically by any application or a management interface. For this release, message loopback capabilities have been enhanced to allow messages injected by an application vs. messages destined to that application to be intercepted and looped, i.e., routed, to a user-specified destination in an independent manner.

2.3.3.2 Distributed Process Management

In the Distributed7 environment, users are allowed to create application domains on one or more hosts comprising a distributed environment. Distributed7 allows users to create multiple application domains on a single host and manage them separately. It also allows users to create distributed application domains that span multiple hosts and manage them in the same way as managing processes on a stand-alone host.

Important*: All process management logic is provided by the Application Process Manager Daemon (*apmd*) process. The actions of the* apmd *can be controlled by a configuration file to satisfy the needs of different applications/environments. By default, the* apmd *configuration file contains information about how to start/stop the most essential set of daemon processes that are instrumental in setting up a distributed processing environment. These mandatory daemon processes must be running on every host at all times for the correct operation of a distributed environment.*

Application Manager

Distributed7 provides advanced features such as rule-based application initialization and recovery. These mechanisms are supported with an Application Manager on an application, domain, and/or node basis. By defining the start-up order of any application at any point in time, and then utilizing heartbeat messaging to allow the exchange of state information, it is ensured that every application will start or restart at the point it was interrupted, thus guaranteeing service availability in the event of software failure.

Error Log

Distributed7 supports mechanisms for hierarchical logging of call-return values on disk. This simplifies problem analysis by allowing users to immediately view triggering events and sequential cause-and-effect relationships of conditions that cause software errors.

Dynamic Trace

Distributed7 supports mechanisms for manipulation of trace categories, allowing applications to dynamically select a category and direct any information to a trace buffer in memory. This data can be selectively analyzed off-line based on trace categories, and then saved on permanent storage media. This feature allows on-demand, non-intrusive monitoring of the status of any application.

2.3.3.3 Distributed Shared Memory Management

Applications running on multiple hosts can share user-space data across Distributed Shared Memory (DSM) segments. Implementation of the DSM framework is a pure software implementation of the widely-known DSM paradigm, and requires no special software or hardware arrangements other than the Distributed7 product itself. Using the DSM API Library, applications can create DSM segments across a network of hosts and exchange information in a transparent manner with applications running on other hosts.

2.3.3.4 Distributed Kernel Memory Management

In the Distributed7 implementation of the SS7 protocol stack, MTP, SCCP, and TCAP protocol layers are embedded mostly in the kernel-level code. When operating under a distributed environment, all of these protocol layers are required to maintain replicated copies of a significant part of their kernel-space data across multiple hosts in a consistent manner. The Distributed Kernel Memory (DKM) and Distributed Record Access (DRA) support kernel-space data distribution across a distributed Distributed7 product:

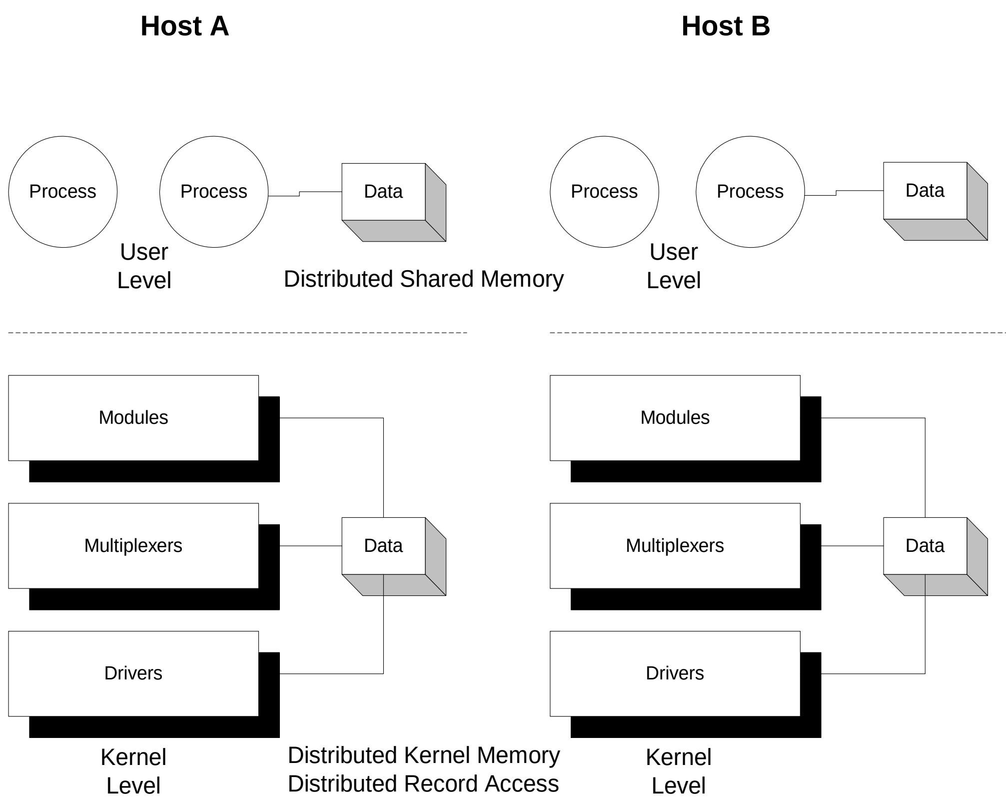


Figure 2-2: Distributed Memory Management

- The DKM framework constitutes the primary means of maintaining replicated copies of kernel-resident data that are in the form of DKM segments.
- The DRA framework fulfills the needs of database-oriented kernel-resident Distributed7 applications. It is with the DRA framework that a kernel application can view its kernel-resident data in the form of a distributed database, and operate on it.

2.3.3.5 Distributed Node/Configuration Management

Distributed7 supports node management based on a *managed object* paradigm. A managed object is an external representation of the functional and physical domain of a system with a set of attributes and operations.

The Object Server provides generic mechanisms to create and manage data, resources, and operations of managed objects, and enables the implementation of multiple, generic user-presentation interfaces such as the Man-Machine Language (MML), the Graphical User Interface (GUI), and the Simple Network Management Protocol (SNMP) Agent.

The Object Server acts as a name server, and provides access to the appropriate managed object handler, depending on the issued request type. Using the CNFG API Programming Guide, application developers can define and dynamically add new managed objects and operations, and customize presentation applications, interfaces, and/or agents.

Man Machine Language

The Distributed7 platform supports MML — a CCITT Z.100 recommended syntax command processing language — for provisioning and monitoring node characteristics. MML supports local or remote execution by the operator from the command line or standard UNIX shell scripts.

Graphical User Interface

An object-oriented GUI is available for node management using X/Motif. The GUI provides a hierarchical view of all the managed objects in the Distributed7 platform. Since the information model is dynamically extensible, any changes occurring in the information tree are dynamically accessible with the GUI.

Actions can be invoked on the managed objects by selecting the object and a corresponding operation with easy-to-use pull-down/pop-up menus. Attributes of the managed object instances can also be displayed. Each action can be applied to certain managed object instance(s) identified by special attributes called keys.

Simple Network Management Protocol

The Distributed7 platform supports the SNMP application-level protocol. SNMP can be thought of as a query language on the Management Information Base (MIB) tree, and is intended to operate over the User Datagram Protocol (UDP). Each message exchange is a separate transaction.

The SNMP agent communicates with managed objects through the Object Server, which acts as a name server and provides access to the correct managed object, depending on the request type issued.

2.3.3.6 Distributed Alarm/Event Management

Each host comprising a Distributed7 environment is equipped with a local copy of the alarm daemon process. Alarm conditions encountered on each host are attended by the local alarm daemon, and are logged locally when necessary. This approach ensures reliable detection/logging of alarms under all circumstances and eliminates the need for duplicating alarm logs on multiple hosts.

Under Distributed7, an application can express interest in any number of alarm conditions that may occur on the local or remote hosts, and be informed about them when they occur. It is also possible for an application process running on a particular host to be notified about alarm conditions on a specified host — or on any host — across the network. Users can designate the alarm daemon process on a specified host to be the global alarm reporter, which monitors alarm events through the console of that host. A copy of all alarm conditions encountered on the other hosts is forwarded to the global instance of the alarm daemon. This function is only for viewing purposes because the global alarm handler does not log alarm events encountered on remote hosts.

2.3.3.7 Redundant LAN Support

Distributed7 supports dual-LAN configurations in which each host comprising a distributed environment is connected to other hosts by two physically separate LAN hardware systems. Dual-LAN configurations provide additional reliability when operating under a distributed environment, and prevents the LAN hardware from becoming a single point of failure. For increased reliability, Distributed7 provides an optional heartbeat mechanism across the kernel-level TCP/IP connections between individual hosts within a distributed environment. It is also instrumental in monitoring the health of individual connections on an on-going basis. In the case of dual-LAN configurations, one of the two TCP/IP connections associated with a particular host operates in active mode and the other connection operates in standby mode. Inter-host message traffic is carried across both active and standby connections at all times. Messages carried across standby connections provide redundancy and do not normally get used: When an active TCP/IP connection becomes unusable, e.g., it is removed or the heartbeat across the connection is lost, the standby connection starts handling the message traffic. This avoids message loss or duplication. These capabilities remain transparent to the users of the system.

Note: *To achieve dual-LAN support, the system must be configured so that all hosts have aliases. See* Section 9.7, System MML Commands*.*

2.3.3.8 Network Clock Synchronization

Distributed7 provides the option to synchronize system clocks on individual hosts within a network. When activated, this capability takes the system clock on the host with the largest IP address as the norm, and adjusts the system clocks on the other hosts to match that norm. Network clock synchronization is essential when operating under a distributed environment. Therefore, unless there is another means of synchronizing system clocks on the individual hosts, this capability must be enabled on all hosts comprising the environment.

2.3.3.9 Application Programming Guides

To help guide application programmers, Distributed7 comes with a set of customer documentation called Application Programming Guides. The following is a list of guides that make up the Application Development Manual.

- SPM API
- APM API
- DSM API
- TCAP API
- ISUP API
- ISUP AoC API
- DKM API

2.3.3.10 Backward Compatibility

Distributed7 API Libraries are largely backward compatible with earlier releases of the Distributed7 product. While most API Libraries contain a new set of function calls and require the use of new header files, extreme care was taken to maintain backward compatibility with AccessMANAGER release 3.x.y. Chapter 8: Compatibility Charts in the *Application Development Manual* contains a complete reference to the backward compatibility of the various API Library routines for Distributed7.

2.3.4 Intelligent Network Emulation (INE)

Distributed7 provides a solid platform to support the development of a broad array of applications. Distributed7 can function as both a Service Control Point (SCP) and a Service Switching Point (SSP) to enable true intelligent network emulation for testing newly designed services. In effect, Distributed7 puts the network design and deployment on the desktop. The same UNIX-based platform that is used to design and test the service can go into the network as the deployment platform.

2.4 Architecture

A high level Distributed7 platform architecture is shown in Figure 2-3.

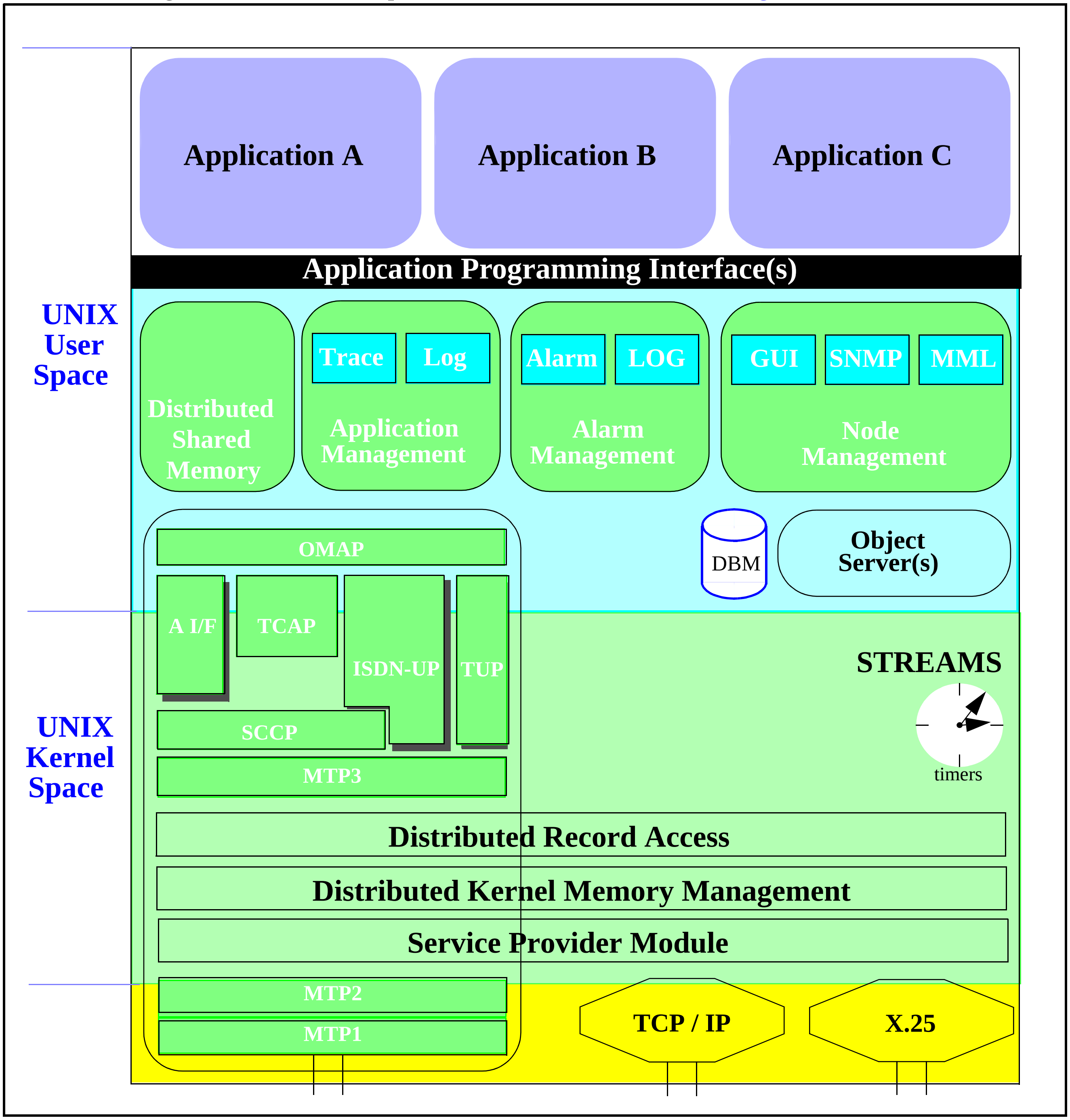


Figure 2-3: Distributed7 Software Architecture

The Distributed7 platform is implemented as a *soft-switch* based on the powerful STREAMS concept. The core building block of the platform is the Service Provider Module

(SPM). It contains extensions to the UNIX kernel, and supports registration and inter-process messaging. Message routing is handled within the soft-switch, which maintains all information regarding platform users. Standard timer facilities are provided to support event management.

Node Management

A generic Node Management capability provides access to the information model by administration personnel. The Object Server performs the bulk of the tasks by providing generic mechanisms to create and manage data, resources, and operations. MML and a GUI are supported for local and dial-up management. SNMP is also supported for remote management.

Process Management

The Process Management provides mechanisms to define rule-based application initialization and recovery strategies on an application, domain, and/or node basis. Additional capabilities allow for hierarchical trace and logging of call-return values, and on-demand, non-intrusive manipulation of trace categories.

Alarm Managememt

The Alarm Management provides a user controllable alarm management capability. Both raw and managed alarms are supported. Alarms contain severity, type, and description, with options for clear and recommended action. Multiple instances of alarms can be kept and separately identified.

SS7 Controller Card

For optimal performance, the SS7 subsystem is tightly integrated with the soft-switch. A bus-resident SS7 Controller card handles the electrical and mechanical characteristics of SS7 data transmission and reception required by MTP Level 1. The SS7 Controller also has its own processor and memory that run the MTP Level 2 signalling data link software.

MTP-L3

The MTP Level 3 is implemented as a STREAMS driver that performs signalling message handling and signalling network management functions. It provides access for its user parts — SCCP and ISUP. MTP Level 3 supports both SS7 and IPC messaging.

SCCP

The SCCP module is also embedded into STREAMS. Its primary functions are to provide the routing control, connectionless services, connection-oriented services, and management functions required by the SCCP protocol layer. The SCCP layer also supports IPC messaging, which allows SCCP users to exchange IPC messages with other objects under the Distributed7 environment. In order to perform protocol-related tasks, the SCCP layer maintains a database that is a collection of kernel-level data structures containing all SCCP subsystems and point codes.

TCAP

The primary function of the TCAP module is to perform dialog-related functions and to support IPC messaging for its users. The TCAP module consists of the component and the transaction sublayers. The transaction sublayer is built upon STREAMS and supports signalling procedures and state machines related to transaction handling. The component layer is implemented in the UNIX user space as an API. It provides message assembly, disassembly, signalling procedures, and state machines related to individual operations, i.e., invocations. An IS41-D API library is also available.

ISUP

The ISUP module is implemented as an application which directly interfaces with the MTP module. For increased performance, the ISUP layer features a kernel-resident module.

The platform provides a number of Application Programming Interfaces (APIs) that allow application developers to develop communication applications utilizing the SS7 and Object Server APIs.

2.4.1 SPM API Library

The SPM API Library provides library calls for registration, message sending, message receiving, and timer handling.

2.4.2 APM API Library

The APM API Library provides the application process management capabilities.

2.4.3 DSM API Library

The DSM API Library allows applications running under a distributed environment to share user-space data in an effective manner.

2.4.4 OAM API Library

The OA&M (Operations, Administration, and Maintenance) API Library supports the OMAP, and enables applications to retrieve measurement data and to manage the SS7 signalling point operation.

2.4.5 Alarm API Library

The Alarm API Library provides library calls for system and/or application software to trigger alarm conditions, specify interest in alarms that may occur anywhere in a cluster and detect them in an asynchronous fashion, and trap and relay selected alarms to network management entities.

2.4.6 MTP API Library

The MTP API Library provides the application with library calls to access the MTP module, and supports basic SS7 message handling library calls.

2.4.7 SCCP API Library

The SCCP API Library provides library calls to access the SCCP module, and supports connectionless and connection-oriented message handling and management library calls.

2.4.8 TCAP API Library

The TCAP API Library provides library calls to access the TCAP module, and supports dialog and component handling library calls. An extended TCAP API Library adds parameter handling capabilities.

2.4.9 Raw TCAP API Library

The raw TCAP API library allows application programs to take advantage of the registration, message distribution, and load-sharing capabilities that are available as part of the Distributed7 TCAP layer without getting involved in any of the transaction and/or component handling capabilities associated with the TCAP layer.

2.4.10 ISUP API Library

The ISUP API Library provides library calls to access the ISUP module from Call Control, and supports message and parameter handling library calls.

2.4.11 ISUP Advice of Charge (AoC) API Library

The ISUP Advice of Charge (AoC) API Library provides the Charging-Application Service Element (ASE) and Application Transport Mechanism-Application Service Element (ASE) Application Programming Interface (API) library calls.

2.4.12 Gateway API Library

This Gateway API Library provides library calls to exercise the gateway functionality available as part of the Distributed7 software products.

2.4.13 IS41-D API Library

The IS41-D API Library supports encoding and decoding Mobile Application Part (MAP) messages.

2.4.14 GSM MAP API Library

The GSM MAP API Library supports encoding and decoding MAP messages.

2.4.15 GSM A-Interface API Library

The GSM A-Interface library defines the necessary signaling protocols to support cellular call processing between any manufacturer's Mobile Switching Center (MSC) and any manufacturer's Base Station Subsystem (BSS).

2.4.16 JAIN TCAP API Library

The Distributed7 JAIN TCAP API provides the main interfaces required to represent TCAP protocol stacks, TCAP applications, as well as the Classes and Exceptions needed to send and receive JAIN TCAP Primitives.

2.4.17 JAIN ISUP API Library

The Distributed7 JAIN ISUP API provides an API to the signaling functions that are needed to support switched voice and data applications. Using the Distributed7 JAIN ISUP API, call control applications can exchange ISUP control messages with Distributed7 ISUP protocol stack in form of Java Event objects.

2.4.18 Passive Monitoring API Library

The Distributed7 Passive Monitoring (PM) API library calls are used to create and manage passive monitoring links under a distributed computing environment..

2.5 Distributed7 System Applications

The Distributed7 software environment provides all the signalling control and transaction handling functions to immediately build the following:

- Intelligent Network-based network nodes with ISUP, and TCAP capabilities that comply with global standards
- Network-based SCPs, including HLRs, VLRs, EIRs, AuCs, and Short Messaging Service Centers (SMSCs) for the wireless networks
- Premises-based Customer Routing Points (CRPs)
- Network Signalling Interfaces to media servers providing voice, fax, and video services
- Protocol converters and gateways

2.5.1 Media Server Network Signalling Interface

A clear trend exists for wireless service providers to integrate network-based voice messaging platforms into their networks rather than rely on stand-alone voice messaging platforms. As cellular networks are built on international standards such as GSM and IS-41, SS7-based voice messaging platforms justify the investment in intelligent networks.

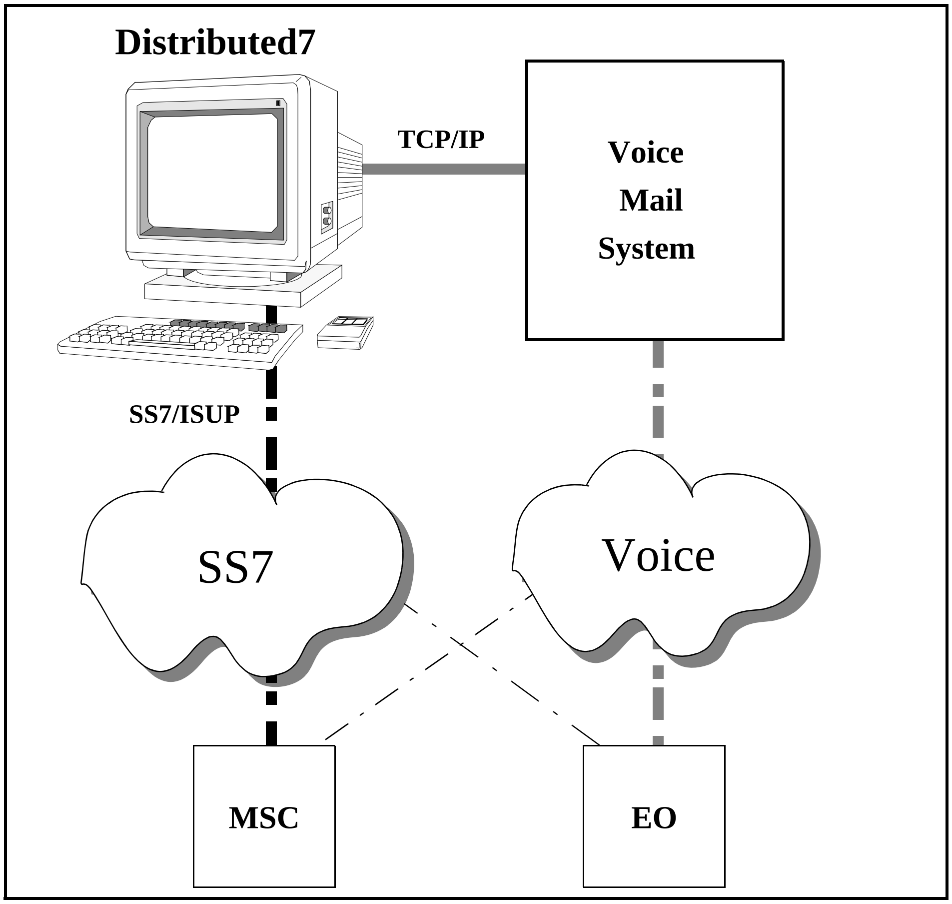


Figure 2-4: Media Server Network Interface

The Distributed7 platform can be used to build Call Control Adjuncts (CCAs), which support standard network and line interfaces with sophisticated call processing. The adjunct operating on an open-architecture UNIX platform can be linked to the Voice Mail System to control the media resources over standard UNIX interfaces such as TCP/IP or X.25, as shown in Figure 2-4.

2.5.2 Customer Routing Point

The Customer Routing Point (CRP) is an advanced 800 feature that supports call processing between the network and a customer premise database. The CRP allows customers to exploit the intelligent network while retaining control of their own information and routing design.

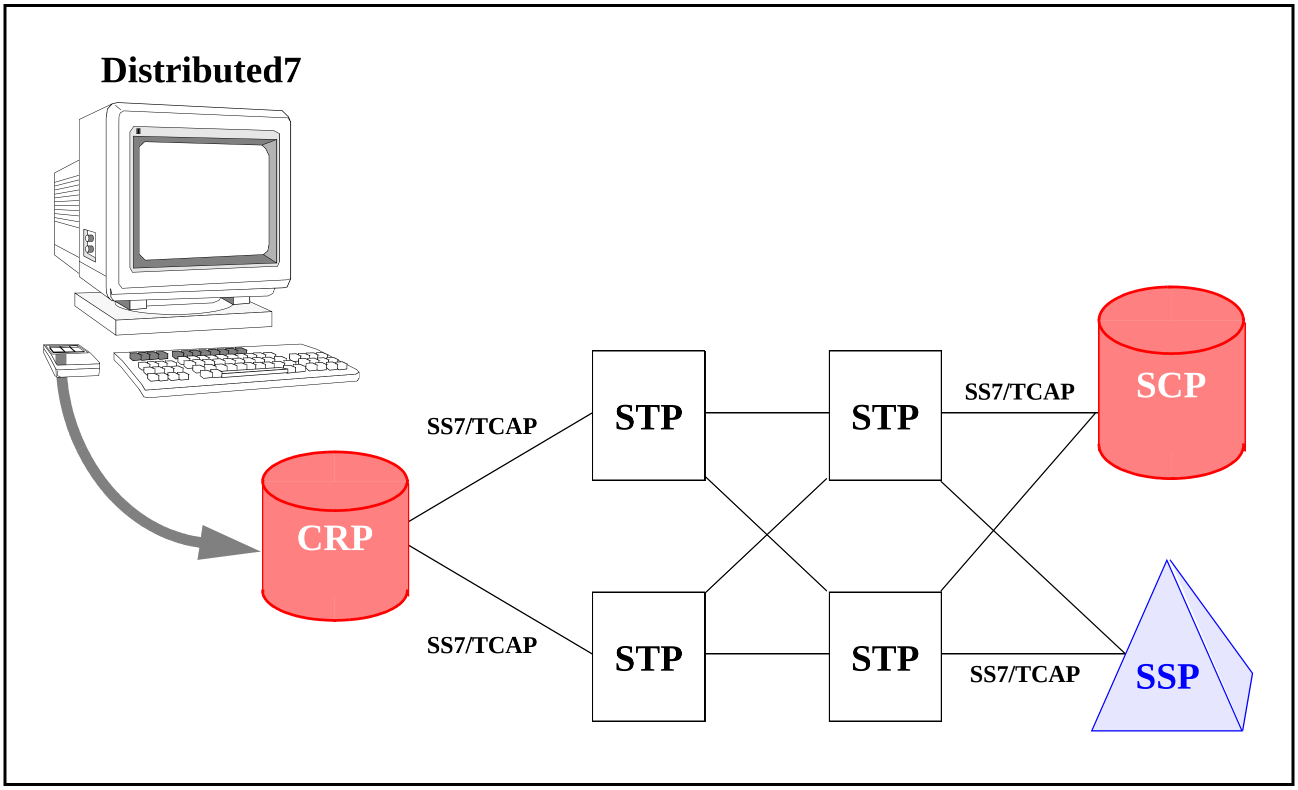


Figure 2-5: Distributed7 as a Customer Routing Point

Customers can use the Distributed7 platform to build a CRP, as shown in Figure 2-5. The CRP contains the logic and data to decide how to appropriately distribute calls to any number of call centers distributed around a geographical area. By tapping into the network ability to provide information such as the dialed 800 number, the calling party number, or caller-entered digits, the CRP enables the customer to make flexible, highly sophisticated routing decisions.

2.5.3 Short Message Server

SMserver is a robust, flexible, open architecture short messaging platform ready to deploy with value added short messaging services.

SMserver manages the transmission of alphanumeric messages between mobile subscribers and external systems such as paging, electronic mail and voice mail systems. Built around a client server architecture, it supports connectivity to external systems via dedicated client modules. It accepts, stores and manages alphanumeric messages to be delivered to mobile subscribers.

SMserver manages all network interactions and provides sophisticated redelivery mechanisms to ensure reliable delivery of short messages. It supports performance monitoring and full billing capabilities. The open Short Message Client Interface facilitates prototyping and deployment of value added services.

Figure 2-6 depicts the high-level software architecture of SMS.

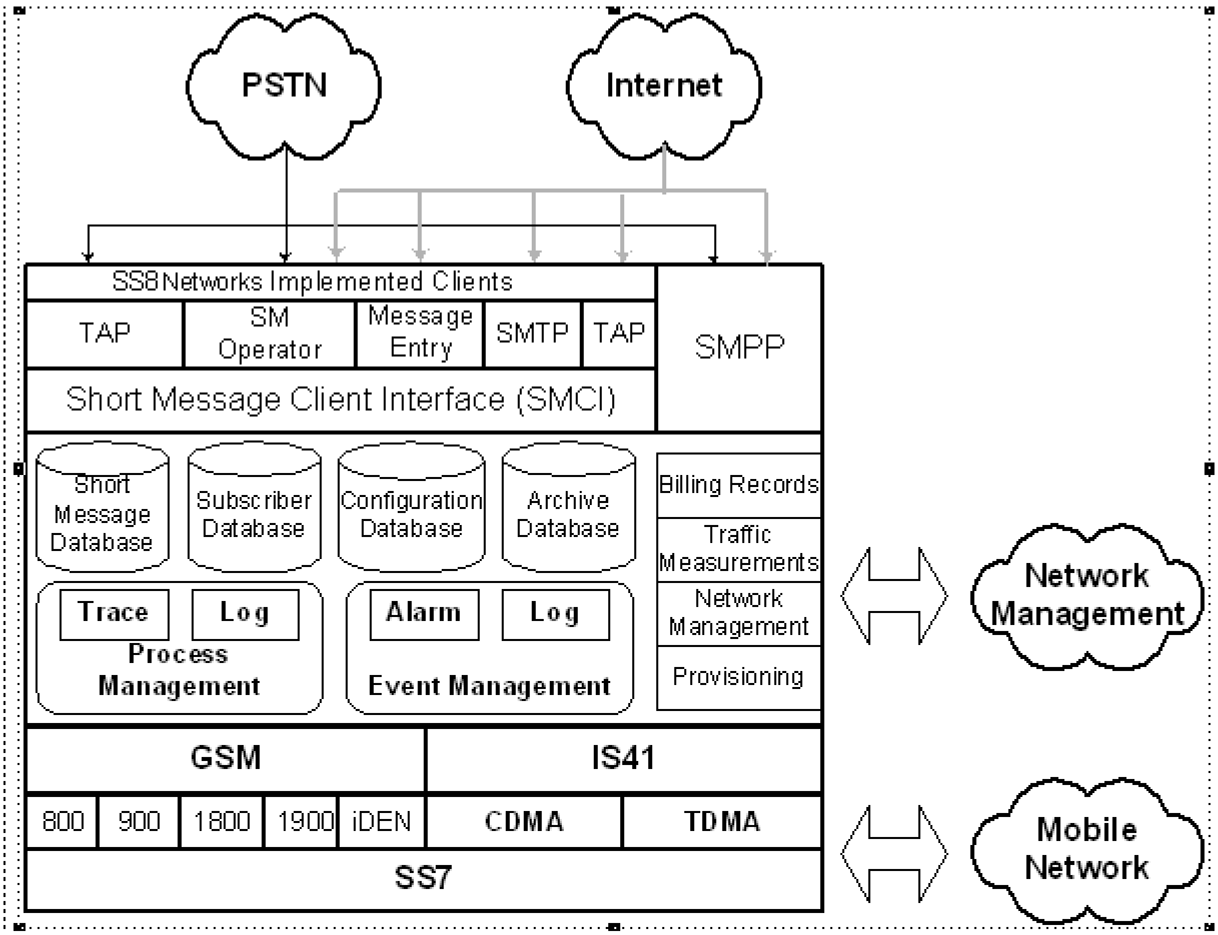


Figure 2-6: SMServer Software Architecture

SMserver is designed to take advantage of symmetric multi-processing and performance scalability offered by the UNIX operating system. The software design is based on object-oriented methodologies, and combines the advent of a high-performance, kernel-resident messaging soft-switch with the traditional client/server methodology.

The soft-switch, implemented in UNIX STREAMS, allows the software to run as part of the operating system kernel, and provides advanced inter-process communication and SS7 messaging. Message routing is handled within the soft-switch, which maintains all the information regarding the different processes that compose the SMSC. This kernel-resident approach results in optimal use of the raw power and resources of the underlying computing platform.

While the messaging soft-switch runs concurrently with the operating system, the client/server architecture eliminates performance bottlenecks by offering the flexibility to run the short message service applications out-of-the box on remote processors.

Management of short message data, subscriber profiles, and service classes is provided through a database manager. The database manager makes use of an Informix C-ISAM database engine that enables disk-based and memory-based data transactions through a common interface. This simple design incorporating a single database access eliminates the need to implement complex lock mechanisms required in multi-user data access environments, and thus provides reliability. Furthermore, the memory-based transaction support minimizes disk I/O and improves performance.

2.5.4 Home Location Register

Home Location Register (HLR) provides a central database of Personal Communications Service (PCS) subscriber information within an IS-41 signaling network. The HLR maintains a permanent entry for each PCS user as well as other information concerning the user's location and status. When network elements other than the HLR require information about a user, such as the Mobile Identification Number/Electronic Serial Number (MIN/ESN), feature data, i.e., call forwarding numbers, or current location, the information is obtained by querying the HLR.

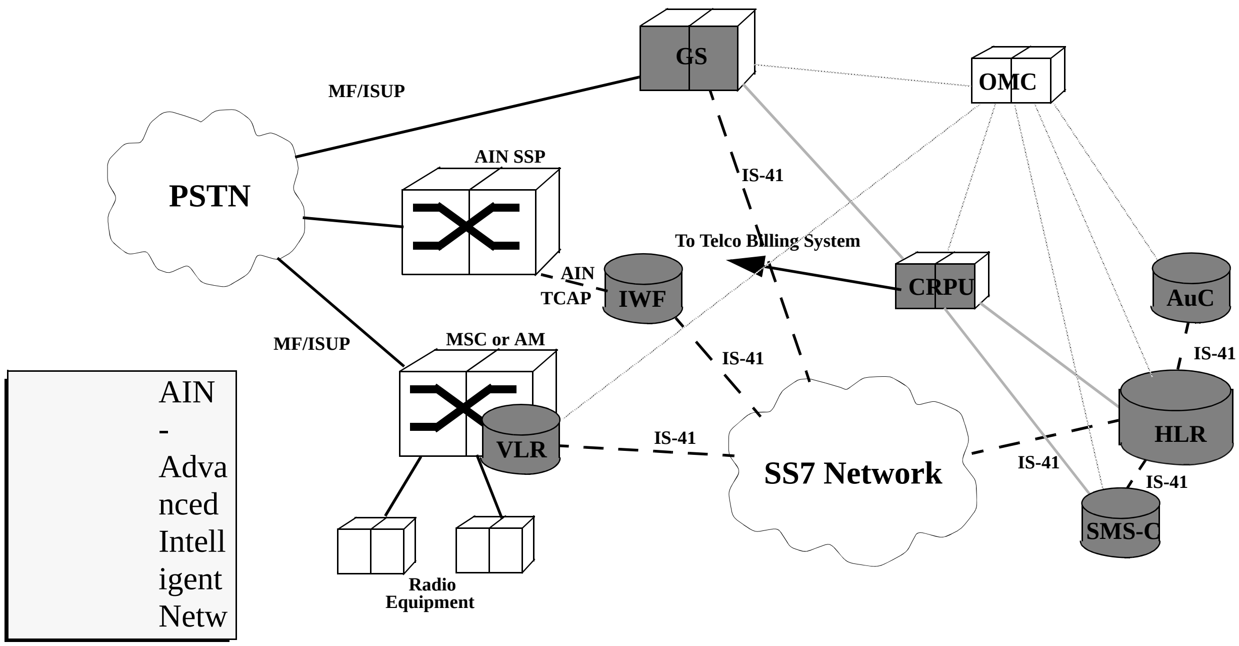


Figure 2-7: HLR in the Wireless Intelligent Network

When a user initially accesses the system, the serving Visitor Location Register (VLR) queries the HLR to validate the user's identity and download necessary feature information. At this time, a user's current VLR location is recorded in the HLR database.

During call termination, i.e., calls to a user, the HLR is queried to determine the user's location — the serving VLR. This information is used to route the call to the Radio Port Control Unit (RPCU) serving the user.

In addition, the HLR is queried to provide feature processing during call originations and terminations.

In general, the HLR provides the following features and functions:

- Centralized storage of persistent subscriber data
- Per-call subscriber validation
- Revenue generating call features
- Support for authentication through co-resident or remote Authentication Centers (AuCs)
- Scalable hardware and software architecture
- Support for multiple service providers or resellers

2.5.5 Visitor Location Register (VLR)

Visitor Location Register (VLR) is a non-persistent database that temporarily holds an entry for each subscriber currently registered within the VLR's area. Generally, a VLR is paired one-to-one with Distributed7 or Mobile Switching Center (MSC), and provides that component with fast access to subscriber data. This data contains a subscriber's active features, location data, and service status information.

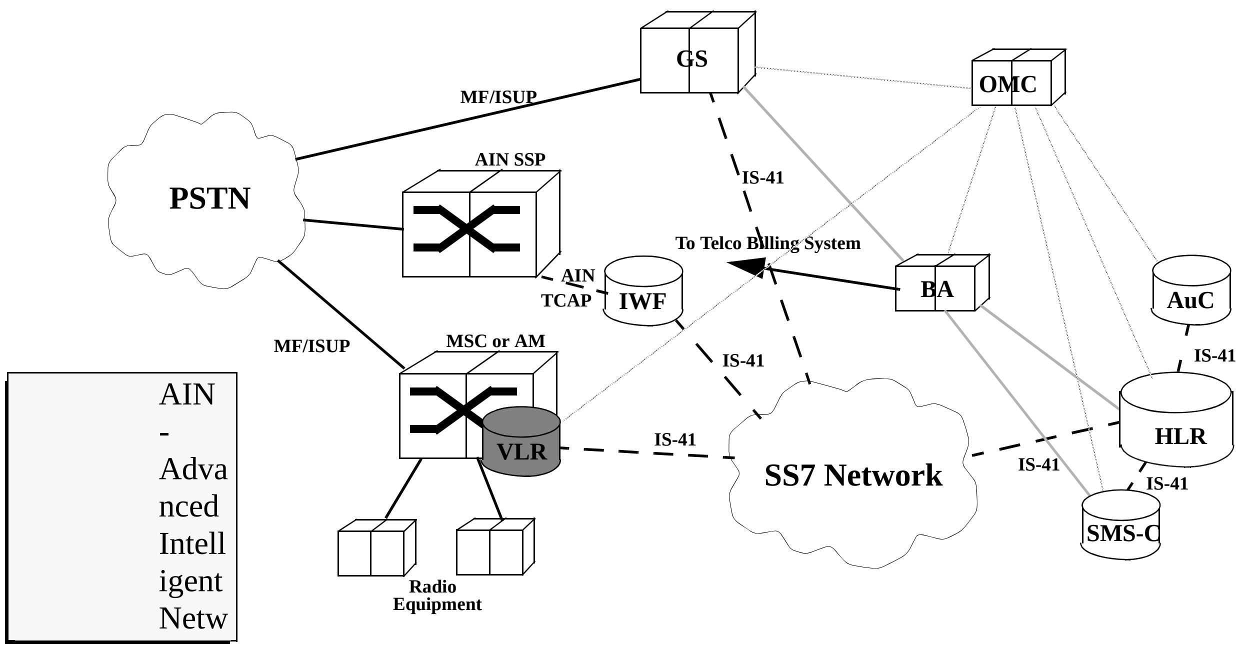


Figure 2-8: VLR in the Wireless Intelligent Network

When network elements other than the VLR require information about a user, e.g., MIN/ ESN, feature data, e.g., call forwarding numbers, or current location, the information is obtained by querying the VLR.

When a user initially accesses the system, the serving AM/MSC registers the subscriber unit (SU) in the VLR. During this process, the AM/MSC sends a query to the VLR, which in turn queries the HLR to validate the user's identity and download necessary feature information into the VLR's volatile database. The VLR's local data is then used during subsequent call originations, terminations, and features invocations.

2.6 Major Standards Compliance

The Distributed7 software platform layers conform to the respective ANSI, ITU/CCITT, and Telecommunications Technology Committee (TTC) standards, as listed below. Over twenty country adaptations to the standards are also available..

Table 2-1: Distributed7 Standards Compliance

SS7 Layer	MTP-2	MTP-3	SCCP	ISUP	TCAP
ANSI	ANSI T1.111.3, 1992	ANSI T1.111.4, 1992, 1996	ANSI T1.112.x, 1992, 1996	ANSI T1.113.x, 1992, 1995	ANSI T1.114, 1992, 1996
ITU	ITU Q.701-Q.703, 1993	ITU Q.704-Q.707, 1993, 1997	ITU Q.711-Q.714, 1993, 1997	ITU Q.761-Q.764, 1993, 1997	ITU Q.771-Q.775, 1993, 1997
TTC	**JT-Q.701-Q.703**	**JT-Q.704-Q.707**	**JT-Q.711-Q.714**	**JT-Q.761-Q.764**	**JT-Q.771-Q.775**

.

Table 2-2: Distributed7 Standards Compliance

SS7 Layer	ANSI	ITU	TTC
MTP-2	**ANSI T1.111.3, 1992**	**ITU Q.701-Q.703, 1993**	**JT-Q.701-Q.703**
MTP-3	**ANSI T1.111.4, 1992**	**ITU Q.704-Q.707, 1993**	**JT-Q.704-Q.707**
SCCP	**ANSI T1.112.x, 1992**	**ITU Q.711-Q.714, 1993**	**JT-Q.711-Q.714**
ISUP	**ANSI T1.113.x, 1992**	**ITU Q.761-Q.764, 1993**	**JT-Q.761-Q.764**
TCAP	**ANSI T1.114, 1992**	**ITU Q.771-Q.775, 1993**	**JT-Q.771-Q.775**

Table 2-3: Distributed7 Wireless Standards Compliance

Interface	IS-41-D	GSM MAP	GSM A
Document Number	TIA/EIA-41-D	ETSI 09.02 version 7.3.0	ETSI GSM 04.01 - 04.08

.

Table 2-4: Distributed7 Standards Compliance

SS7 Layer	ANSI	ITU	TTC
MTP-2	**ANSI T1.111.3, 1992**	**ITU Q.701-Q.703, 1993**	**JT-Q.701-Q.703**
MTP-3	**ANSI T1.111.4, 1992**	**ITU Q.704-Q.707, 1993**	**JT-Q.704-Q.707**
SCCP	**ANSI T1.112.x, 1992**	**ITU Q.711-Q.714, 1993**	**JT-Q.711-Q.714**
ISUP	**ANSI T1.113.x, 1992**	**ITU Q.761-Q.764, 1993**	**JT-Q.761-Q.764**
TCAP	**ANSI T1.114, 1992**	**ITU Q.771-Q.775, 1993**	**JT-Q.771-Q.775**

Table 2-5: Distributed7 Wireless Standards Compliance

Interface	Document Number
IS-41-C	EIA/TIA/PN_2991
GSM MAP	ETSI 09.02 version 4.11

2.7 Capacity and Configuration Options

2.7.1 SS7 Database Capacity

Table 2-3 shows the default configurations for the MTP, SCCP, ISUP, and TCAP layers, and for the INE. These parameters can be modified according to the customer's needs.

Table 2-6: Standard SS7 Database Capacity

	Description	ANSI	ITU
MTP	destination point codes (SS7 + capability)	2048	2048
	destinations behind link sets	2048	2048
	links	511	511
	link sets	64	64
	routes per destination	16	16
SCCP	destination point codes	8192	8192
	subsystems per destination	256	256
	concerned point codes per subsystem	8192	8192
	global title types	16	16
	translation types per global title type	256	256
	simultaneous open SCCP connections	16384	16384
	simultaneous reassembly processes per system	16	16
ISUP	destination point codes	2048	2048
	circuit groups per destination	3040	3040
	total trunk groups for all destinations	8192	8192
	circuits per circuit group	32	32
TCAP	simultaneously open dialogues	262144	262144
	local instances of the same subsystem	63	63
INE	signalling points	8	8

2.7.2 Host Platform Options

Distributed7 1.6.0 is available on the following hardware platforms:

Table 2-7: Host Platform Options

Make	Model	Processor	Operating System	Bus	Board PCI-X	Board PCIe	Development Environment
Sun	Netra T2xx series	UltraSPARC T2	Solaris 10	PCI-X	PCI334[*] PCI334a PCI370PQ PCI370APQ PCI372PQ PCI372APQ ARTIC2000 PMC4539F PMC8260	Not applicable	Sun Studio 11 (SC5.8) Sun Workshop 5.0 GNU C 3.3 GNU C 3.1
	Netra Vxxx series Fire Vxxx series		Solaris 8 Solaris 9 Solaris 10				
	Netra X4200 M2	AMD Opteron	Solaris 10 (64-bit kernel)				
	Netra T5xx series	UltraSPARC T2	Solaris 10	PCI-X PCIe		HDCII-LPe HDC3-LPe	
	Enterprise Txxx series Fire Txxx series	UltraSPARC T1, T2		PCIe	Not applicable		
	Fire X2xxx series Fire X4xxx series	AMD Opteron	Solairs 10 X86 (64-bit kernel)				
	Netra/Fire X4250 Netra/Fire X4450	Intel Xeon					
	In case your server type is not listed above, please visit www.sun.com/servers for more information on the PCI bus type.						
IBM	All servers with PCI-X or PCIe bus type and supporting Solaris 10, X86. Please visit http://www.sun.com. Please visit: http://www.sun.com/bigadmin/hcl/data/sol/systems/views/all_servers_oem.page1.html.						
Dell	All servers with PCI-X or PCIe bus type and supporting Solaris 10, X86. Please visit: http://www.sun.com/bigadmin/hcl/data/sol/systems/views/all_servers_oem.page1.html.						
HP	All servers with PCI-X or PCIe bus type and supporting Solaris 10, X86. Please visit: http://h71028.www7.hp.com/enterprise/cache/492635-0-0-0-121.html.						

[*] For Sun SPARC machines that have the Sun HSIP package installed, the PCI334 card is incompatible. To use the PCI334 card on a SPARC with an existing Sun HSIP package, the HSIP driver must be removed.

***Note**: D7 supports Solaris 8, Solaris 9, and Solaris 10 in 32-bit mode and 64-bit mode.*

Distributed7 1.6.0 is guaranteed on:

- Sun Solaris 8 with kernel patch level 108528-11
- Sun Solaris 9 with kernel patch level 118558-11
- Solaris 10 with kernel patch level 118833-17
- On X86 systems, Solaris 10 with kernel patch level 118855-19

Consult TAC for updated patch levels for all operating systems.

2.7.3 SS7 Controller Options

Table 2-8: Available SS7 Controller Options

Bus Architecture	Physical Interface	Ports per Controller for Links up to 64 kbps	Ports per Controller for High Speed Links	Numberof Controllers per System[1]
PCIbus	RS-449	Up to 4		8
	V.35	Up to 4		
	T1	Up to 4[2] Up to 24[3,8] Up to 64[9]	Up to 4[10]	4
	E1	Up to 4[4] Up to 24[5,8] Up to 64[9]		
CompactPCI bus	T1	Up to 24[6,8] Up to 64[9]		8
	E1	Up to 24[7,8] Up to 64[9]		
PCIe	T1	92[11]	Up to 4	1
	E1	124[11]		

1. *Also limited by the number of available slots on the bus.*
2. *Available with PCI370 board.*
3. *Available with PCI370PQ and PCI370APQ boards.*
4. *Available with PCI372 board.*
5. *Available with PCI372PQ and PCI372APQ boards.*
6. *Available with CPC370PQ board.*
7. *Available with CPC372PQ board.*
8. *Although PCI3xPQ, PCI3xAPQ and CPC3xPQ boards allow configuration of up to 24 links, use of more than 16 with the PCI3xPQ card is not recommended for systems requiring full bandwidth on all configured links.*
9. *Available with PMC8260 and ARTIC1000/2000 boards.*
10. *Available with the PMC4539F board.*
11. *Available with HDCII-LPe boards.*

2.8 External Dependencies

All the executable modules listed in Table 2-6 implicitly require the following shared libraries that are provided by the operating system vendor:

- Standard C library (***libc***)
- Network Service Library (***libnsl***)
- Socket Library (***libsocket***)
- Dynamic Linking Library (***libdl***)
- Internationalization Library (***libintl***)

- Wide Character Library (***libw***)

In addition, the following environment variables must be set:

- ***EBSHOME***: to point to Distributed7 installation directory
- ***LD_LIBRARY_PATH***: to include the external shared library installation directories.

Table 2-9: Executable External Dependencies

Module	Shared Libraries	Environment Variables
AccessAlarm	C++ library (provided by NewNet Communication Technologies, LLC from SunSoft) (libC)	
AccessISUP	C++ library (provided by NewNet Communication Technologies, LLC from SunSoft) (libC)	
AccessMOB	C++ library (provided by NewNet Communication Technologies, LLC from SunSoft) (libC) Motif library (libXm) (not provided by NewNet Communication Technologies, LLC), X Windowing System Libraries (libX11, libXt, libXext) (provided by o/s vendor)	MOTIFHOME OPENWINHOME DISPLAY
AccessOMAP	C++ library (provided by NewNet Communication Technologies, LLC from SunSoft) (libC)	
AccessSNMP	C++ library (provided by NewNet Communication Technologies, LLC from SunSoft) (libC)	
AccessStatus	C++ library (provided by NewNet Communication Technologies, LLC from SunSoft) (libC) Motif library (libXm) (not provided by NewNet Communication Technologies, LLC), X Windowing System Libraries (libX11, libXt, libXext) (provided by o/s vendor)	TC_LIBRARY TK_LIBRARY
AccessMonitor	Similar to AccessStatus entry	
upmd	C++ library (provided by NewNet Communication Technologies, LLC from SunSoft) (libC)	
scmd	C++ library (provided by NewNet Communication Technologies, LLC from SunSoft) (libC)	
MML	C++ library (provided by NewNet Communication Technologies, LLC from SunSoft) (libC)	

The NewNet Communication Technologies, LLC libraries listed in Table 2-7 require the associated shared library. These libraries are used with the OA&M and/or Object Server APIs.

Table 2-10: Library External Dependencies

Library	Shared Libraries
liboam	C++ library (provided by NewNet Communication Technologies, LLC from SunSoft) (libC)
libapm	C++ library (provided by NewNet Communication Technologies, LLC from SunSoft) (libC)
libcnfg	C++ library (provided by NewNet Communication Technologies, LLC from SunSoft) (libC)

Chapter 3: # Concepts

3.1 Chapter Overview

This chapter provides a general overview of SS7, the Distributed7 managed object concept, and the Product Specifications.

3.2 SS7 Overview

SS7 is the protocol used to transfer signaling information between entities in SS7-based common channel signaling networks. Along with the SS7 protocol, a specific network architecture is also defined. This section provides a brief overview of both the architecture and the protocol layers.

3.2.1 SS7 Protocol

The SS7 protocol consists of seven layers that map to the OSI model. Each layer has a specific function:

- The first three layers of the protocol are the *Message Transfer Part (MTP).* MTP is responsible for reliable transfer and delivery of messages across the signaling network. All other layers use the MTP layer directly or indirectly.
- The *Signaling Connection Control Part (SCCP)* layer extends the addressing capability of MTP.
- The *Integrated Services Digital Network User Part (ISUP)* layer provides control of circuit-switched network connections, e.g., call set-up.
- The *Transaction Capabilities Application Part (TCAP)* interfaces with SCCP to provide information exchange independent of circuits, such as database queries.

Figure 3-1 compares the SS7 protocol to the OSI model.

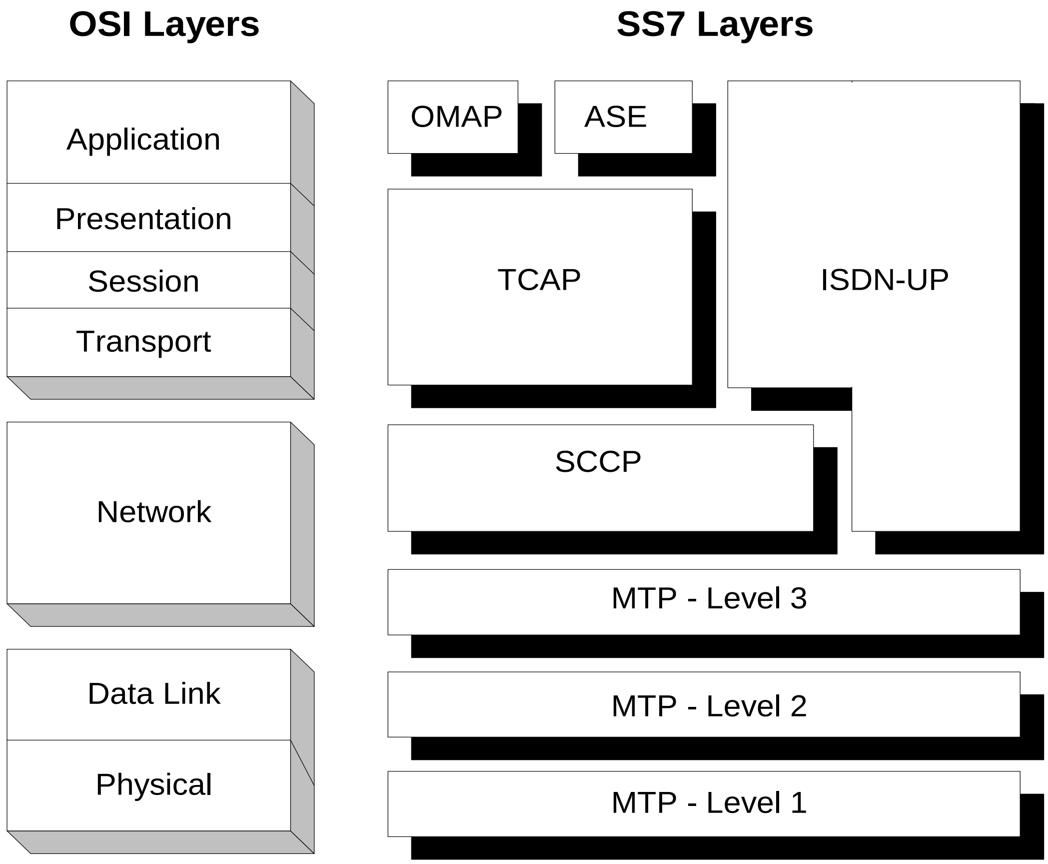


Figure 3-1: SS7 Protocol

3.2.2 SS7 Architecture

The SS7 network architecture consists of the physical elements in Figure 3-2.

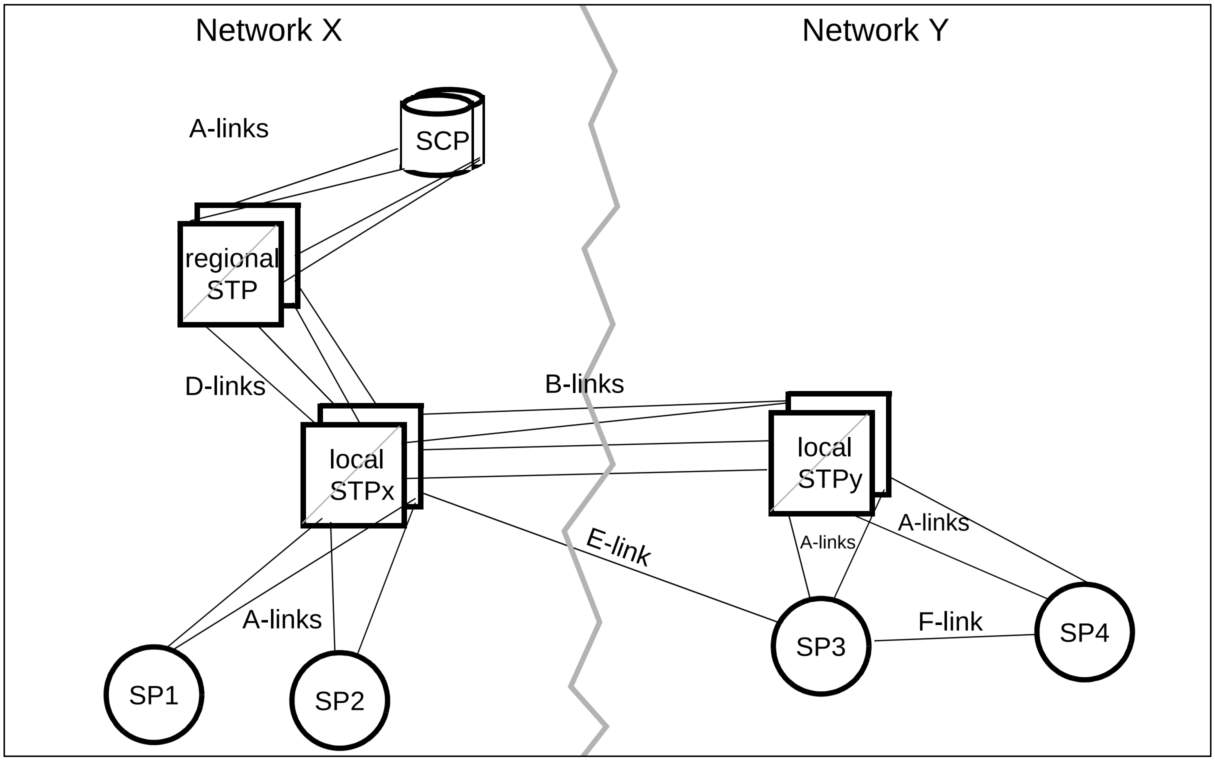


Figure 3-2: SS7 Network Architecture

The SPs and SCPs are particular kinds of Signaling End Points (SEPs). SEPs are nodes where signals terminate. Signaling Points (SPs) in this diagram represent entities such as the switches that set-up and connect calls. The Service Control Point (SCP) is a special element that processes queries from other nodes using data in its database. Examples of SCP applications are 800 number translation and calling card validation. Other types of SEPs also exist.

The other type of node is a Signal Transfer Point (STP). STPs transfer message packets between SEPs. They function as routers in the SS7 network. STPs are always deployed in mated pairs for redundancy.

Links connect the nodes together. Different link names are associated with different node connections, as follows:

- A-links: Between any type of SEP and its normal home STP pair - minimum of one to each mate of the pair
- B-links: Between STP pairs of the same hierarchy, e.g., connection of STP pair to STP pair of another network; each mate must have at least one link to each mate of the other pair
- C-links: Between the mates of an STP pair (not shown in the diagram)

- D-links: Between an STP pair and its regional STP pair - each mate must have at least one link to each mate of the other pair
- E-links: Between an SEP and a foreign STP
- F-links: Between two SEPs (bypassing or as an alternate to the STP)

Normally, Distributed7 applications are SEPs that use A-links or F-links. However, Distributed7 can be used as an STP as well.

Other components of the SS7 network architecture are link sets, routes, and route sets. These elements are more abstract.

- A *link set* is a group of links that originate from the same node and physically go to the same destination.
- A *route* is associated with a final destination, such as another SEP, that does not necessarily have a direct physical connection. It is the path the message should take to get to its final destination. The link sets from an SEP to its STP pair may be associated with each other in the form of equal priority routes. Equal priority routes allow message traffic to go on both link sets since the STPs of the pair are essentially the same destination. For example, for SP1 to send a message to SP4, the message would go on a link to SP1's STP pair. The STP would take care of the rest of the routing, so that first link is the only part of the route that must be specified.
- A r*oute set* is the group of routes to a specific destination.

Once the Distributed7 software is installed, the system must be configured and provisioned for the SS7 network. MTP configuration is required prior to activating a signaling point on the network. When the SCCP and ISUP options are included, these must also be configured before activating the signaling point. A Man-Machine Language Terminal Handler and a Graphical User Interface (GUI) are available for MTP, SCCP, and ISUP configuration.

3.2.2.1 SEP-STP Recommended Link Configuration

The SS7 network architecture standards state that A-link sets from an SEP to a mated STP pair should be on fully, physically diverse paths. Using one SS7 controller board for both A-link sets does not meet that diversity requirement. SS7 controller boards support from 4 to 64 links, as shown in Section 2-8, Available SS7 Controller Options. However, in order to meet the standards, a minimum of one link from each SS7 controller board should be used for an A-link connection to a mated STP pair. Figure 3-3 illustrates this connection.

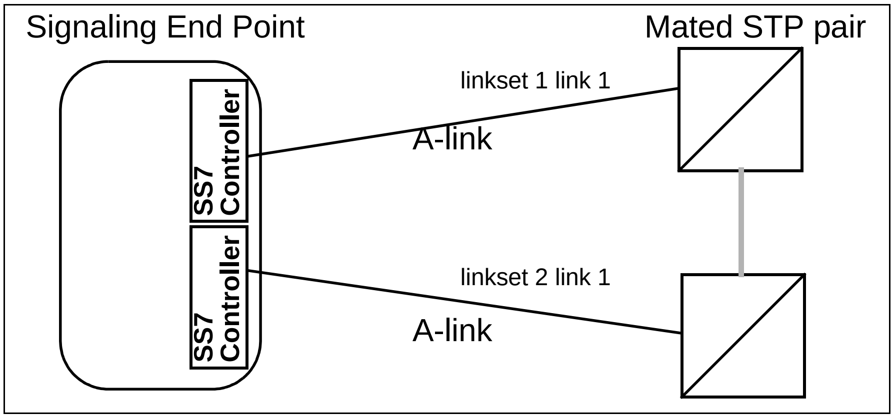


Figure 3-3: A-Link Configuration (2 links)

Additional links can be added to the diverse link sets in various ways using the link capacity of the SS7 Controller boards. The next link for each link set can be on the opposite SS7 Controller board of the first link in that set, as shown in Figure 3-4.

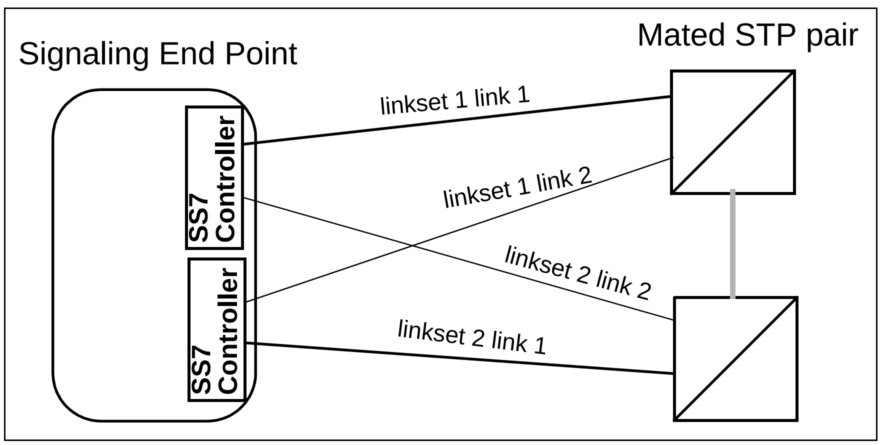


Figure 3-4: A-Link Configuration (greater than 2 links)

For a fault tolerant system, one SS7 controller board should be on each bus, and one link set should come from each board, similar to Figure 3-3 .

3.2.3 Distributed7 SS7 Node

For the Distributed7 system to be an SEP in the SS7 network, it must be configured to identify itself as an SS7 node and to define its physical connections and routing options. Different levels of configuration are needed, depending on the type of SEP the node is. Configuration levels include:

- MTP database
  All Distributed7 nodes must configure the MTP database. This database defines the node, its links, and the destination nodes with which it communicates. The information is used by the MTP protocol layer to process incoming and outgoing SS7 messages.
- SCCP database
  The SCCP database must be configured if the Distributed7 application is an SCCP or TCAP application. The SCCP database identifies additional routing information used by the SCCP layer to process messages from these types of applications.
- ISUP database
  The ISUP database must be configured if the Distributed7 application is a Call Control type of application that interfaces with the ISUP layer. This database defines the nodes and circuits that are involved in call processing with ISUP.

These databases are described in the following subsections. More information about configuring a particular database may be found in *Chapter 9: Man-Machine Language Commands*.

3.2.3.1 MTP Database

The MTP database is configured using MML commands or the AccessMOB GUI. It must be configured in the order given below.

1. Signaling System and MTP information

Each node in an SS7 network uses a specific protocol and has specific MTP parameters that identify it. All parameters, including the protocol of the node, can be modified. The current parameters of the MTP part are the ***Protocol*** of the system, i.e., ***ANSI_92, ANSI_96, ITU_93, ITU_97, Variant***, ***Point Code Size*** (14 bit, 16 bit, and 24 bit), ***Multiple Congestion and Priority Support*** in SS7 messaging, ***Signaling Link Test Message*** support, and some implementation related parameters.

2. Own Signaling Point Information

Each SS7 node has a unique identity called the Signaling Point Code (SPC). The SPC is used in the SS7 protocol for addressing messages to other nodes. It may also be referred to as an Origination Point Code or a Destination Point Code.

The Network Indicator (NI) identifies the type of network the node is in. NI types must match in order to communicate. The type of node must also be identified as an SEP, an STP, or an SEP with routing capability, as depicted in Figure 3-2.

The Distributed7 node can also be given a name of up to 20 alphanumeric characters. This is used for administrative purposes and has no implication on the protocol handling. In addition, it is possible to specify that a signaling link test message be sent periodically, or if a restart procedure needs to be invoked.

3. Linksets and Links

A link set must be defined before the links it contains can be configured. It identifies the destination of the links and the number of links. After the link set is defined, each link must be configured.

Each link in the link set has a unique identity known as the Signaling Link Code (SLC), which must be agreed upon by the management at both termination points of the link.

In MTPL2, ***basic*** and ***preventive cyclic retransmission*** are the two types of error correction methods for SS7 links. If the signaling links are established via satellite, then the preventive cyclic retransmission method should be used. For links where one-way propagation delay is less than 15 ms, basic method should be used.

Links also have a ***hostname***, a ***board type***, a ***slot number***, and a relative ***port number*** in the board. The port number identifies the physical connection.

The maximum number of links in a link set is 128. The total number of links in the link set is determined by the number of links provisioned to that link set. However, the administrator can specify the number of active links to be maintained, as well as the number of links to be loaded. MTP attempts to keep the specified number of links active. The decision of which links to activate or deactivate depends on the priority of the links. Of the active links, only the loaded links carry traffic.

4. Route Sets and Routes

A route set must be created before the routes it contains can be added. Each route set has a destination point code and at least one route that can reach the destination point. Alternate routes can be added to the route set. Each route has a priority, which determines the order of the routes carrying traffic. The maximum number of routes in a route set is 16. The top priority route has a priority value of Ø. The maximum number of routes that can have the same priority in a route set is 2. These are load-sharing routes. MTP attempts to use the highest priority route available.

5. Timers

Timers values specified by the protocol are provisional; they are set to default values that are within the ranges specified in the specification documents for each particular protocol variant. In general, the default time-out values for an expected response from external sources are the maximum of the range. For periodic tests, the default value is the minimum of the range. These values may be changed, but unless there are special requirements, they do not need to be.

L3 timers are global for all Signaling Network Management (SNM) activities, and L2 timers can be configured for each signaling link.

6. Thresholds

The thresholds for congestion are set to values optimized for average 100-byte MSUs. For applications with special traffic or size requirements, these values can be changed. Congestion thresholds are configured for each signaling link.

3.2.3.2 SCCP Database

SCCP supplements the MTP to provide OSI-defined Network Services to its users. SCCP can support subsystems numbered from 2 to 255 (number 1 is reserved) on its own signaling point, and an unlimited number of subsystems on remote nodes. SCCP coordinates the correct communications among these subsystems or with subsystem backups.

SCCP does this with the help of two addresses: (1) *Called Party Address (CLD)*, which is the destination address, and (2) *Calling Party Address (CLG)*, which is the origination address. These two addresses together are referred to as SCCP Routing Information. These addresses can be encoded in two forms:

- Destination Point Code (DPC) + Subsystem Number (SSN) combination: DPC is a designated address uniquely identifying a network entity. SSN is local addressing information identifying each of the SCCP users, i.e., applications.
- Global Title (GT): A GT is an address like a dialed digit. A translation capability is provided by the SCCP to convert GTs to a DPC+SSN combination. This translation can be performed at the originating point or at the destination point. To use global title routing, the global title tables must be provisioned.

To configure the SCCP database, all signaling point codes with which a node communicates must be entered into that SCCP database. SCCP tracks the status of these signaling points. The SCCP network database must be configured after the MTP database. Only SPCs entered in the MTP database can be entered into the SCCP database.

Next, all subsystem numbers that are accessed by a node and that belong to specified remote signaling points must also be added to the database.

Optionally, for each subsystem on a node, a series of Concerned Point Codes (CPC) can be identified. A CPC is the SPC of an SS7 node that already exists in the SCCP database, and that needs to be notified when the status of the local subsystem changes.

Signaling points and subsystems can also be mated to each other for backup purposes. This information isd also specified in the SCCP database.

3.2.3.3 ISUP Database

The ISUP database identifies the nodes to be included in a circuit-switched SS7 network over which basic voice, data, and supplementary services are provided. The nodes must already be defined in the MTP database, i.e., generally by route sets. The database also identifies which circuit groups and circuits to the destination are used for SS7. Certain office settings and ISUP timers may also be modified.

3.3 Managed Objects and the Object Server

The resources of the Distributed7 system are modeled in terms of *Managed Objects* (MOs). An MO is defined as an external view of the functional and physical domain of the system with a set of parameters and operations that may be performed upon it. MOs are resources that define the system, such as subsystems or link sets.

Each MO belongs to an MO Server. If the MO Server is not active, then none of the operations for the MOs beloning to that server can be executed or viewed with the help tool. For example, the ***isupd*** process must be running to use ISUP-related MML commands or to see ISUP MOs in the GUI.

This section describes MOs, MO Servers, and their use.

3.3.1 Object Server

The Object Server is a module containing all the Distributed7 MO Servers and an application programming interface (API). It deals with configuration, fault, security, and performance management of the hardware and software components. It also enables the implementation of multiple management interfaces, e.g., MMI, GUI or SNMP, or user-defined MO Servers. These interfaces can be developed using the CNFG API Library. The management interfaces must interact with the system according to the managed object hierarchy described in *Section 3.3.1.1 on page 3-11*.

With the CNFG API Library, application developers can add new MOs to create their own custom presentation applications and agents. The Object Server consists of the CNFG library, an object database, and several Managed Object Servers, as shown in Figure 3-5.

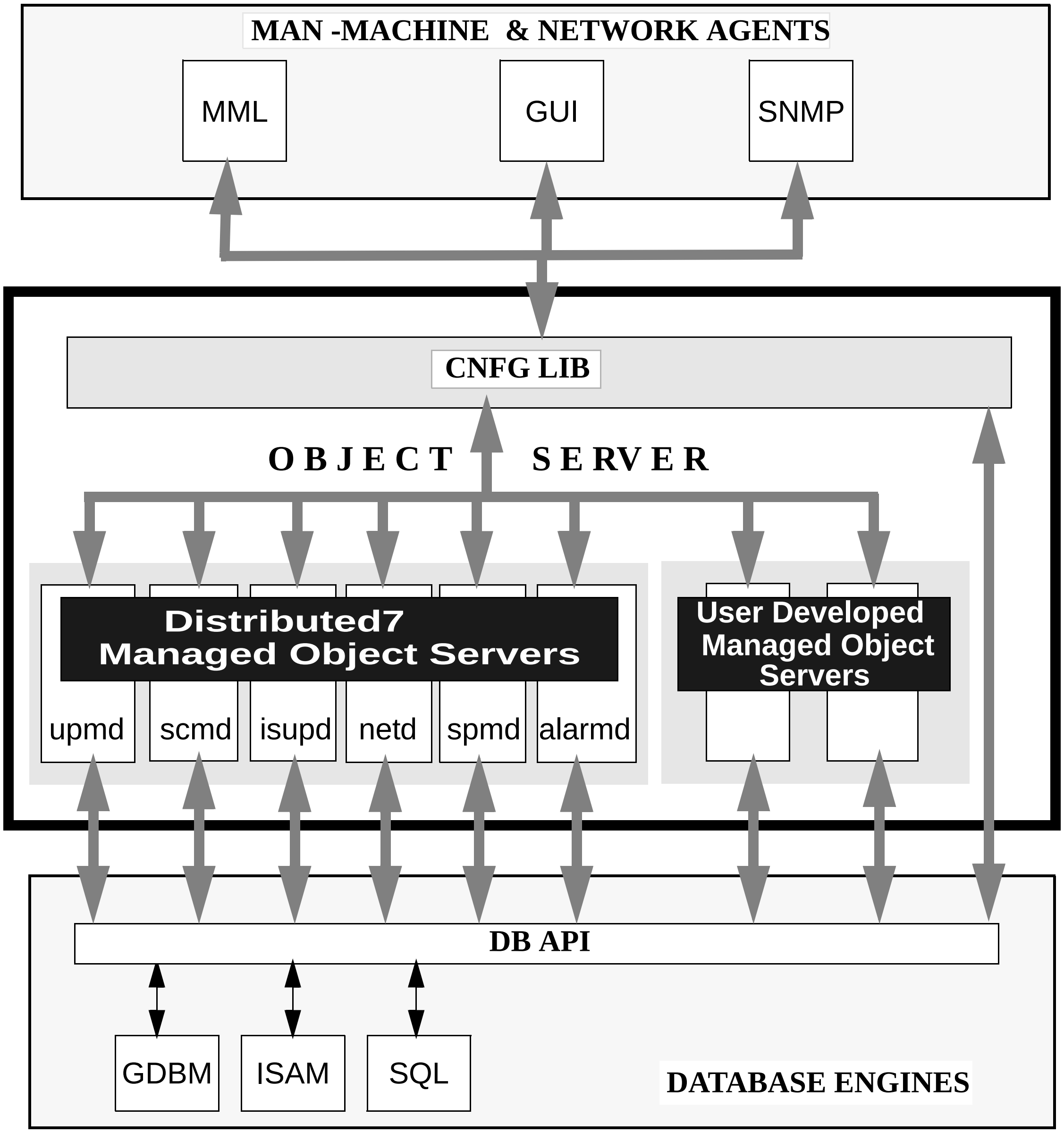


Figure 3-5: Object Server Internal Architecture

3.3.1.1 MO Groupings

Figure 3-6 and Figure 3-7 show the MO *parent-child* hierarchy in the Distributed7 product. The dynamic nature of the Object Server allows this hierarchy to be changed easily.

The six distinct object groups and their respective MO Servers are described below.

Table 3-1: Object Groups and MO Servers

Object Group	MO server
spm	spmd
alarm	alarmd
network	netd
mtp	upmd
sccp	scmd
isup	isupd

When the MO server processes, i.e., daemon processes, start up, they create their MOs and define the operations for them.

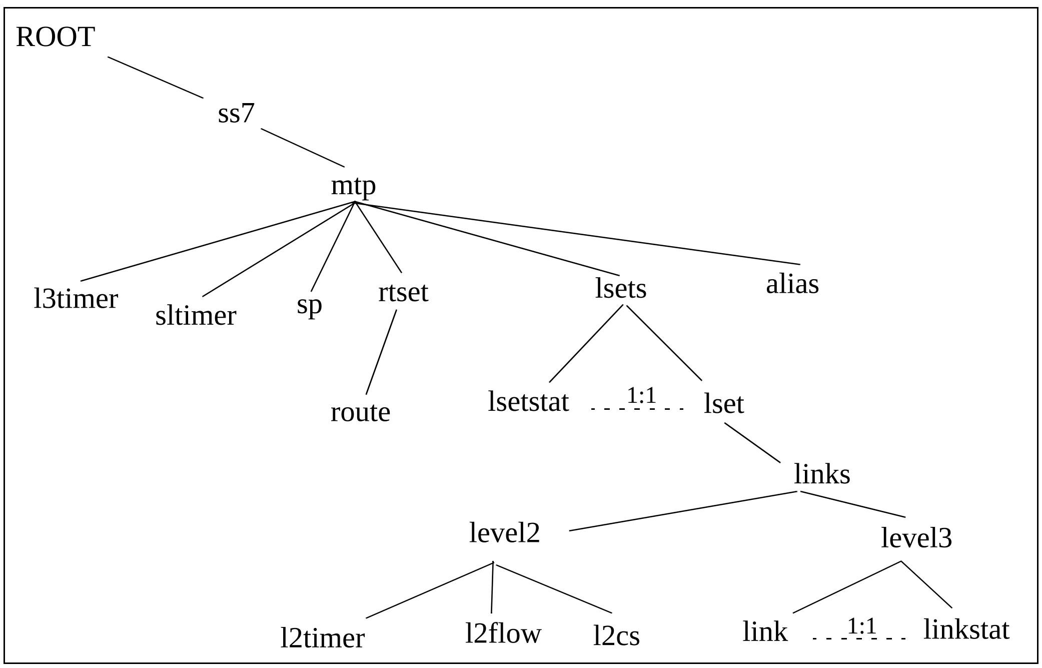


Figure 3-6: MTP Managed Object Containment Structure

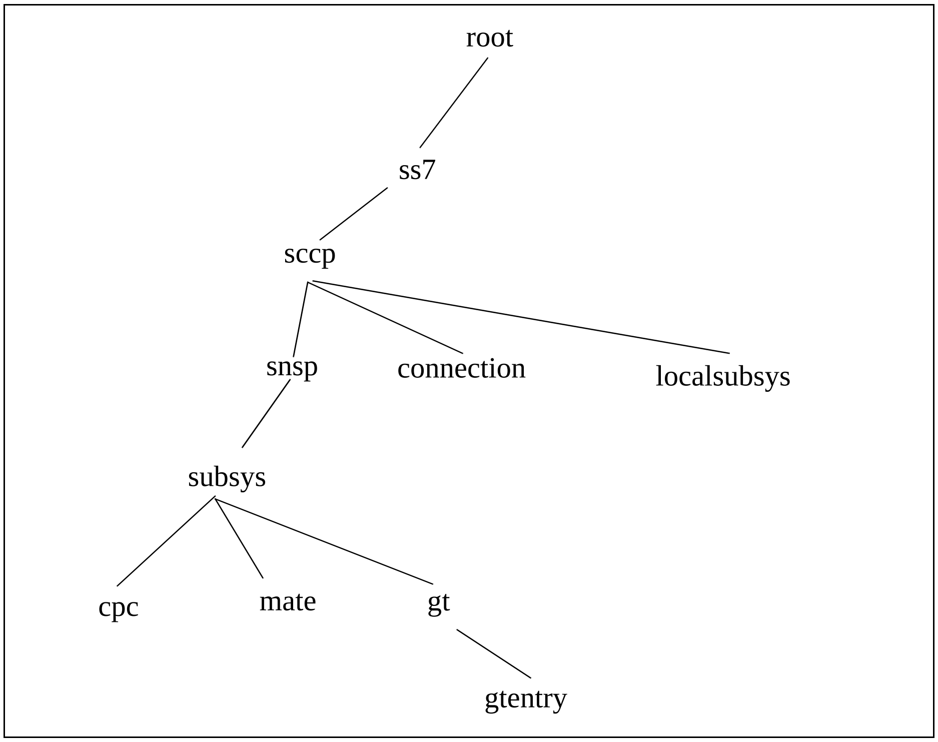


Figure 3-7: SCCP Managed Object Containment Structure

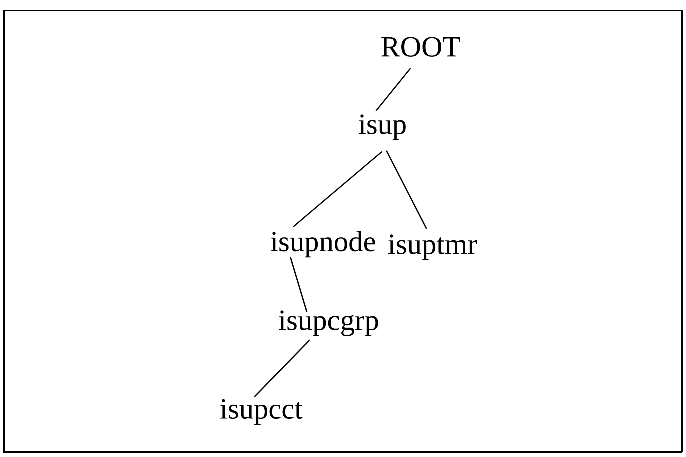


Figure 3-8: ISUP Managed Object Containment Structure

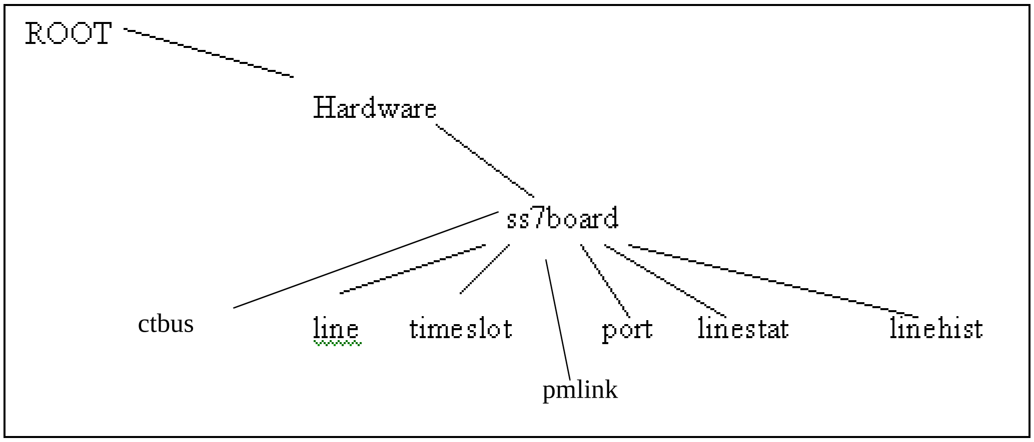


Figure 3-9: SPM Managed Object Containment Structure

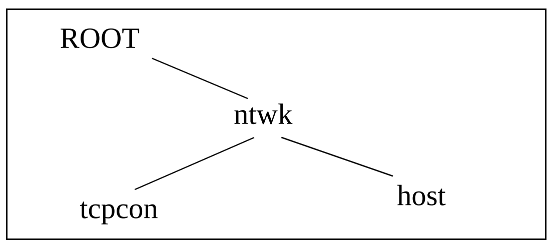


Figure 3-10: Network Managed Object Containment Structure

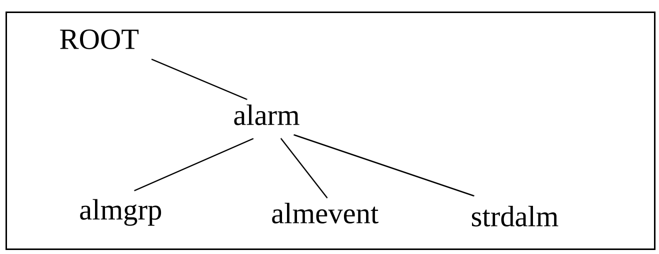


Figure 3-11: Alarm Managed Object Containment Structure

3.3.2 Managed Objects

Each MO has a set of operations and specific parameters defined for it by its MO Server. MOs also have a hierarchy in relation to each other.

A configurable MO has one or more operations:

- Add
- Modify
- Delete
- Display/view

An MML command name is made up of the operation name and the MO name, as in OPERATION-MO. An example is ***ADD-LSET***.

Each box in the main window of the Distributed7 GUI (AccessMOB) is an MO, and the operation is defined by the mode.

An individual instance of an MO, such as a specific link set, is defined by its parameters. Parameters have certain properties and restrictions, and provide the MO with a unique identity.

3.3.2.1 MO Parameters

Operations change the state of the provide the MO with a unique identity through its parameters. Parameters are realized as the parameters of an MML command, or as the fields in the dialog box of the AccessMOB GUI. Table 3-5 and Table 3-6 list the parameters and the operations for the Distributed7 object groups. Some of the MOs have no operations defined. These MOs serve as abstract objects, and were included to model the system better.

For each of the MO parameters, there are predefined *types*. Parameters can be **String**, **Integer, Set**, or **PointCode**. The **PointCode** parameter type is a special case in which input and output formats are String, but the internal representation is Integer. For instance, the *PC="10-20-30"* ANSI point code is converted to its integer representation, and vice versa. The **Set** parameter type is another special case in which input and output formats are a set of strings that correspond to an integer value. For instance, the input set for the NI parameter of the SP MO is INTERNATIONAL, SPARE, NATIONAL, and RESERVED, which correspond to 0, 1, 2, and 3.

The **Access** column in Table 3-2 identifies whether parameters can be read, modified, or both.

- A **read-write** parameter can be both read and modified
- A **read-only** parameter cannot be modified
- A **write-only** parameter is not displayed to the human operator
- The **read-create** parameter cannot be modified; it is used as a key

Table 3-2: SPM Branch Managed Object Descriptions

Managed Object	Parameters	Types	Access	Set Values	Operations
hardware	N/A	N/A	N/A	N/A	N/A
ss7board	hostname	String	read-create	-	ADD DELETE DISPLAY MODIFY
	boardnm	Set	read-create	sbs334/pci334/vbrd/pci3xpq/ pci3xapq/cpc3xpq/pmc8260/ artic8260/pmc4539/adaxm	
	inst	Integer	read-create	-	
	conf	Set	read-write	ON/OFF/SUSPEND/RESUME	
	pm	Set	read-create	ON/OFF	
	modules	String	read-write	-	
	state	Set	read-only	DETACHED/ATTACHED/ CDWNLOADED/READY	
	class	Set	read-only	I/II/III/IV	
	ports	Integer	read-create	-	
	lines	Integer	read-only	-	
	clockmode	Set	read-write	LINE/INTERNAL/ EXTERNAL/REMOTE/ NOTUSED	
	clockspan	Set	read-write	1/2/3/4/5/6/7/8	
	spmlinkno	Integer	read-only	-	
line	hostname	String	read-create	-	DISPLAY MODIFY
	boardnm	Set	read-create	sbs334/pci334/vbrd/pci3xpq/ pci3xapq/cpc3xpq/pmc8260/ artic8260/pmc4539/adaxm	
	inst	Integer	read-create	-	
	span	Set	read-create	1/2/3/4/5/6/7/8	
	class	Set	read-only	I/II/III/IV	
	line_typ	Set	read-write	E1/T1/J1 E1HSL/T1HSL/J1HSL	
	line_frmmod	Set	read-write	T1ESF/T1ZBTSI/T1SLC96/ T1SFRM/T1SF4/E1FEBE/ E1CRC4/E1BASIC	
	line_cod	Set	read-write	T1B8ZS/T1B7ZS/E1HDB3/ AMI	
	line_len	Set	read-write	L133/L266/L399/L533/L655/ L110/L220/L330/L440/L550/ L660/LB000/LB075/LB150/ LB225	
	line_imp	Set	read-write	I75/I100/I120	
	line_lpbk	Set	read-write	NONE/LOCAL/REMOTE	
	line_accs	Set	read-write	FRONT/REAR	

Table 3-2: SPM Branch Managed Object Descriptions (Continued)

Managed Object	Parameters	Types	Access	Set Values	Operations
port	hostname	String	read-create	-	DISPLAY MODIFY
	boardnm	Set	read-create	sbs334/pci334/vbrd/pci3xpq/ pci3xapq/cpc3xpq/pmc8260/ artic8260/pmc4539/adaxm	
	inst	Integer	read-create	-	
	portnum	Integer	read-create	-	
	class	Set	read-only	I/II/III/IV	
	type	Set	read-write	DTE/DCE/NOTUSED	
	baud	Set	read-write	600/1200/2400/4800/7200/ 9600/16000/19200/32000/ 38400/48000/56000/64000/ 1544000/2048000	
	lpbkmode	Set	read-write	NONE/LOCAL/REMOTE	
	idledetect	Set	read-write	OFF/ON	
timeslot	hostname	String	read-create	-	ADD DELETE DISPLAY MODIFY
	boardnm	Set	read-create	sbs334/pci334/vbrd/pci3xpq/ pci3xapq/cpc3xpq/pmc8260/ artic8260/pmc4539/adaxm	
	inst	Integer	read-create	-	
	desttype	Set	read-create	LINE/HDLC/CTBUS	
	destspan	Integer	read-create	-	
	destslot	Integer	read-create	-	
	class	Set	read-only	I/II/III/IV	
	origtype	Set	read-write	LINE/HDLC/NOCONNECT/ CTBUS	
	origspan	Integer	read-write	-	
	origslot	Integer	read-write	-	
linestat	hostname	String	read-create	-	DISPLAY MODIFY
	boardnm	Set	read-create	pci3xpq/pci3xapq/cpc3xpq/ pmc8260/artic8260/pmc4539/ adaxm	
	inst	Integer	read-create	-	
	span	Set	read-create	1/2/3/4/5/6/7/8	
	errevents	Integer	read-write	-	
	curstatus	Set	read-only	SIG-AV/SIG-UNAV	
	curtimer	Integer	read-only	-	
	cur-ES	Integer	read-only	-	
	cur-UAS	Integer	read-only	-	
	24h-ES	Integer	read-only	-	
	24h-UAS	Integer	read-only	-	
	vldinttotal	Integer	read-only	-	

Table 3-2: SPM Branch Managed Object Descriptions (Continued)

Managed Object	Parameters	Types	Access	Set Values	Operations
linehist	hostname	String	read-create	-	DISPLAY MODIFY
	boardnm	Set	read-create	pci3xpq/pci3xapq/cpc3xpq/pmc8260/artic8260/pmc4539/adaxm	
	inst	Integer	read-create	-	
	span	Set	read-create	1/2/3/4/5/6/7/8	
	interval	Integer	read-only	-	
	reset	Set	write-only	NO/YES	
	ES	Integer	read-only	-	
	UAS	Integer	read-only	-	
ctbus	hostname	String	read-create	-	DISPLAYMODIFY
	boardnm	Set	read-create	pmc8260/artic8260	
	inst	Integer	read-create	-	
	refclk	Set	read-write	C8A/C8B/NETREF1/NETREF2/SCSA2/SCSA4/SCSA8/MVIP/HMVIP	
	refinv	Set	read-write	OFF/ON	
	fbmode	Set	read-write	C8A/C8B/NETREF1/NETREF2/INTERNAL/LINE	
	fbspan	Set	read-write	1/2/3/4/5/6/7/8	
	fb	Set	read-only	OFF/ON	
	comp	Set	read-write	OFF/ON	
	c8a	Set	read-write	OFF/ON	
	c8b	Set	read-write	OFF/ON	
	nrmode	Set	read-write	NETREF1/NETREF2/INTERNAL/LINE	
	nrspan	Set	read-write	1/2/3/4/5/6/7/8	
	nr8khz	Set	read-write	OFF/ON	
	nrinv	Set	read-write	OFF/ON	
	nract	Set	read-only	OFF/ON	
	nr1	Set	read-write	OFF/ON	
	nr2	Set	read-write	OFF/ON	
	grp_a	Set	read-write	OFF/2048/4096/8192	
	grp_b	Set	read-write	OFF/2048/4096/8192	
	grp_c	Set	read-write	OFF/2048/4096/8192	
	grp_d	Set	read-write	OFF/2048/4096/8192	
	grp_e	Set	read-write	OFF/2048/4096/8192	
	grp_f	Set	read-write	OFF/2048/4096/8192	
	grp_g	Set	read-write	OFF/2048/4096/8192	
	grp_h	Set	read-write	OFF/2048/4096/8192	

Table 3-2: SPM Branch Managed Object Descriptions (Continued)

Managed Object	Parameters	Types	Access	Set Values	Operations
pmlink	hostname	String	read-create	-	ADD DELETE DISPLAY MODIFY
	boardnm	Set	read-create	pci3xpq/pci3xapq/	
	pmc8260/ artic8260				
	inst	Integer	read-create	-	
	port	Integer	read-create	-	
	adminstat	Set	read-write	ACTIVATE/DEACTIVATE	
	operstat	Set	read-only	SHUTOFF/INACTIVE/IDLE/ OOS/ALIGNING/ INSERVICE/PROC-OUT	
	linkf	Integer	read-only	-	
	rxframes	Integer	read-only	-	
	rxoctets	Integer	read-only	-	
	rsu_e	Integer	read-only	-	
	d_rxl	Integer	read-only	-	

Table 3-3: NETWORK (NTWK) Branch Managed Object Descriptions

Managed Object	Parameters	Types	Access	Set Values	Operations
ntwk	hostname	String	read-create	-	DISPLAY MODIFY
	mode	Set	read-write	STNDLN/DSTRBTD	
	clocksync	Set	read-write	ON/OFF	
	frequency	Integer	read-write	-	
	dualhost	String	read-write	-	
	netmask1	String	read-write	-	
	netmask2	String	read-write	-	
host	hostname	String	read-create	-	ADD DELETE DISPLAY MODIFY
	rmthost	String	read-create	-	
	alias	String	read-create	-	
	conf	Set	read-write	ON/OFF	
	rmthosttyp	Set	read-write	AMGR/OTHER	

Table 3-3: NETWORK (NTWK) Branch Managed Object Descriptions (Continued)

Managed Object	Parameters	Types	Access	Set Values	Operations
tcpcon	hostname	String	read-create	-	DISPLAY MODIFY
	rmthost	String	read-create	-	
	mode	Set	read-write	AUTO/MASTER/SLAVE	
	service	Set	read-write	NETDBASE	
	proto	Set	read-write	TCP	
	modules	String	read-write	NIMOD/TCMOD	
	hbeat	Set	read-write	ON/OFF	
	frequ	Integer	read-write	-	
	maxtries	Integer	read-write	-	
	act_est	Set	read-write	IGNORE/INFORM	
	act_rmv	Set	read-write	IGNORE/INFORM	
	hb_loss	Set	read-write	NOACTION/SYNCDATA	
	state	Set	read-only	IDLE/PENDING/REQ2SYNC/ ESTBLSHD	

Table 3-4: ALARM Branch Managed Object Descriptions

Managed Object	Parameters	Types	Access	Set Values	Operations
alarm	hostname	String	read-write	-	DISPLAY MODIFY
	display	Set	read-write	OFF/ON	
	cons_thrs	Set	read-write	NONE/INFO/MINOR/ MAJOR/CRITICAL/FATAL	
	user_thrs	Set	read-write	NONE/INFO/MINOR/ MAJOR/CRITICAL/FATAL	
	repeat	Integer	read-write	-	
	update	Set	read-write	OFF/ON	
	global	Set	read-write	OFF/ON	
	log_file_num	Integer	read-only	-	
almgrp	hostname	String	read-write	-	DISPLAY MODIFY
	group	String	read-write	-	
	cons_thrs	Set	read-write	NONE/INFO/MINOR/ MAJOR/CRITICAL/FATAL	
	user_thrs	Set	read-write	NONE/INFO/MINOR/ MAJOR/CRITICAL/FATAL	
	num_of_alms	Integer	read-only	-	

Table 3-4: ALARM Branch Managed Object Descriptions (Continued)

Managed Object	Parameters	Types	Access	Set Values	Operations
strdalm	hostname	String	read-create	-	DISPLAY DELETE
	group	String	read-create	-	
	module	Integer	read-create	-	
	type	Integer	read-create	-	
	parameters	String	read-create	-	
	severity	Set	read-only	NONE/INFO/MINOR/ MAJOR/CRITICAL/FATAL	
	first_occ	String	read-only	-	
	last_occ	String	read-only	-	
	num_of_occur	Integer	read-only	-	
	alm_text	String	read-only	-	
almevent	hostname	String	read-create	-	ADD DISPLAY DELETE
	req_hostname	String	read-create	-	
	groupno	Integer	read-create	-	
	module	Integer	read-create	-	
	type	Integer	read-create	-	
	threshold	Set	read-create	NONE/INFO/MINOR/ MAJOR/CRITICAL/FATAL	

Table 3-5: MTP Managed Object Descriptions

Managed Object	Parameter	Types	Access	Set Values	Operations
ss7	N/A	N/A	N/A	N/A	N/A

Table 3-5: MTP Managed Object Descriptions (Continued)

Managed Object	Parameter	Types	Access	Set Values	Operations
mtp	spno	Integer	read-create	-	ADD DELETE DISPLAY MODIFY
	protocol	Set	read-create	ITU_93/ITU_97/ANSI_92/ANSI_96	
	variant	Set	read-write	GENERIC/NEW_ZEL/AT&T/GTE/ETSI97/BELL	
	pcsize	Set	read-create	14_BIT/16_BIT/24_BIT	
	mcong	Set	read-write	OFF/ON	
	mprio	Set	read-write	OFF/ON	
	sltc	Set	read-write	OFF/ON	
	restart	Set	write-only	ON	
	mtp_state	Set	read-only	CREATED/ISOLATED/RESTARTING/RESTARTED	
	rtrc	Set	read-write	OFF/ON	
	rpo2lpo	Set	read-write	OFF/ON	
	nicheck	Set	read-write	OFF/ON	
	dpccheck	Set	read-write	OFF/ON	
sp	spno	Integer	read-create	-	DISPLAY MODIFY
	name	String	read-write	-	
	spc	PointCode	read-write	-	
	ni	Set	read-write	-	
	type	Set	read-write	-	
alias	apc	PointCode	read-write	-	ADD DELETE DISPLAY MODIFY
	ogpc	Set	read-write	OFF/ON	
	infltr	Set	read-write	OFF/SPC/APC	
	fltract	Set	read-write	ALARM/UPU	
l3Timer	timer	Integer	read-create	-	DISPLAY MODIFY
	value	Integer	read-write	-	
	minval	Integer	read-only	-	
	maxval	Integer	read-only	-	
sltimer	timer	Integer	read-create	-	DISPLAY MODIFY
	value	Integer	read-write	-	
	minval	Integer	read-only	-	
	maxval	Integer	read-only	-	
lsets	N/A	N/A	N/A		N/A

Table 3-5: MTP Managed Object Descriptions (Continued)

Managed Object	Parameter	Types	Access	Set Values	Operations
lset	lset	String	read-create	-	ADD DELETE DISPLAY MODIFY
	dpc	PointCode	read-create	-	
	type	Set	read-write	ALINK/BLINK/CLINK/ DLINK/ELINK/FLINK	
	loaded	Integer	read-write	-	
	active	Integer	read-write	-	
	abbit	Set	read-create	A/B	
	emergency	Set	read-write	OFF/ON	
lsetstat	lset	String	read-create	-	DISPLAY MODIFY
	dpc	PointCode	read-only	-	
	status	Set	write-only	CLR_ACT/SET_ACT	
	act	Set	read-only	OFF/ON	
	avl	Set	read-only	OFF/ON	
links	N/A	N/A	N/A	N/A	N/A
level2	N/A	N/A	N/A	N/A	N/A
l2Timer	link	String	read-create	-	DISPLAY MODIFY
	timer	Integer	read-create	-	
	value	Integer	read-write	-	
	minval	Integer	read-only	-	
	maxval	Integer	read-only	-	
l2flow	link	String	read-create	-	DISPLAY MODIFY
	fclevel	Integer	read-create	-	
	congonval	Integer	read-write	-	
	congabvla	Integer	read-write	-	
	disconval	Integer	read-write	-	
	discabval	Integer	read-write	-	

Table 3-5: MTP Managed Object Descriptions (Continued)

Managed Object	Parameter	Types	Access	Set Values	Operations
l2cs	link	String	read-create	-	DISPLAY
	stat	Set	read-only	OFF/OOS/IA/AR/ANR/IS/PO/H_NA	
	tminsrv	Integer	read-only	-	
	suerm	Integer	read-only	-	
	algnf	Integer	read-only	-	
	linkf	Integer	read-only	-	
	rsu_e	Integer	read-only	-	
	d_rxl	Integer	read-only	-	
	d_txl	Integer	read-only	-	
	d_bo	Integer	read-only	-	
	txframes	Integer	read-only	-	
	rxframes	Integer	read-only	-	
	txoctets	Integer	read-only	-	
	rsoctets	Integer	read-only	-	
level3	N/A	N/A	N/A	N/A	N/A
link	link	String	read-create	-	ADD DELETE MODIFY DISPLAY
	lset	String	read-create	-	
	slc	Integer	read-create	-	
	priority	Integer	read-write	-	
	l2ecm	Set	read-write	BASIC/PCR	
	pcrN1	Integer	read-write	-	
	pcrN2	Integer	read-write	-	
	hostname	String	read-create	-	
	hoststatus	Set	read-only	UNAVAILABLE/AVAILABLE/CONFLICT	
	boardnm	Set	read-create	sbs334/pci334/vbrd/pci3xpq/pci3xapq/cpc3xpq/pmc8260/artic8260/pmc4539	
	inst	Integer	read-create	-	
	port	Integer	read-create	-	

Table 3-5: MTP Managed Object Descriptions (Continued)

Managed Object	Parameter	Types	Access	Set Values	Operations
linkstat	link	String	read-create	-	DISPLAY MODIFY
	lset	String	read-only	-	
	slc	Integer	read-only	-	
	status	Set	write-only	CLR_ACT/SET_ACT/ CLR_ECO/SET_ECO/ CLR_EMR/SET_EMR/ CLR_LPO/SET_LPO/ CLR_INH/SET_INH/ TEST_SLTM	
	act	Set	read-only	OFF/ON	
	emr	Set	read-only	OFF/ON	
	eco	Set	read-only	OFF/ON	
	loaded	Set	read-only	OFF/ON	
	avl	Set	read-only	OFF/ON	
	lin	Set	read-only	OFF/ON	
	rin	Set	read-only	OFF/ON	
	lpo	Set	read-only	OFF/ON	
	rpo	Set	read-only	OFF/ON	
rtset	rtset	String	read-create	-	ADD DELETE DISPLAY
	dpc	PointCode	read-create	-	
	rtype	Set	read-create	MEMBER/CLUSTER/ NETWORK	
	capability	Set	read-only	OFF/ON	
	state	Set	read-write	INACC/ACC/REST	
	cong	Set	read-only	OFF/ON	
route	rtset	String	read-create	-	ADD DELETE DISPLAY
	lset	String	read-create	-	
	priority	Integer	read-create	-	
	state	Set	read-only	NI/RS/PR	
	lsstate	Set	read-only	UA/AV	
	current	Set	read-only	OFF/ON	
	rtcong	Set	read-only	OFF/ON	
	lscong	Set	read-only	OFF/ON	

Table 3-6: SCCP Branch Managed Object Descriptions

Managed Object	Parameters	Types	Access	Set Values	Operations
sccp	spno	Integer	read-create	-	ADD, DELETE, MODIFY, DISPLAY
	protocol	Settype	read-write	ANSI_92/ANSI_96/ITU_93/ ITU_97	
	variant	Settype		NONE/ATT/APLUS/SNET	
	pcind	Settype	read-write	YES/NO	
	t_conn_est	Integer	read-write	-	
	t_ias	Integer	read-write	-	
	pcind	Integer	read-write	-	
	t_rel	Integer	read-write	-	
	t_guard	Integer	read-write	-	
	t_reset	Integer	read-write	-	
	t_segment	Integer	read-write	-	
snsp	spc	PointCode	read-create	-	ADD, DELETE, DISPLAY
	status	Settype	read-only	-	
	xlate	Settype	read-only	-	
	concerned	Settype	read-only	-	
	has_ssn	Settype	read-only	-	
subsys	spc	PointCode	read-create	-	ADD, DELETE, DISPLAY
	ssn	Integer	read-create	-	
	mspc	PointCode	read-only	-	
	mssn	Integer	read-only	-	
	ssnStatus	Settype	read-only	-	
	xlate	Settype	read-only	-	
	has_cpc	Settype	read-only	-	
localsubsys	ssn	Integer	read-only	-	DISPLAY
	mssn	Integer	read-only	-	
	mspc	Pointcode	read-only	-	
	ssn_status	Settype	read-only	-	
	xlate	Settype	read-only	-	
	has_cpc	Settype	read-only	-	
cpc	spc	PointCode	read-create	-	ADD, DELETE, DISPLAY
	ssn	Integer	read-create	-	
	cpc	PointCode	read-create	-	

Table 3-6: SCCP Branch Managed Object Descriptions (Continued)

Managed Object	Parameters	Types	Access	Set Values	Operations
mate	spc	PointCode	read-create	-	ADD, DELETE, DISPLAY
	ssn	Integer	read-create	-	
	mspc	PointCode	read-only	-	
	mssn	Integer	read-only	-	
	waiting_for_grant	Settype	read-only	-	
	ignore_sst	Integer	read-only	-	
gtEntry	io	Settype	read-create	-	ADD, DELETE, DISPLAY, MODIFY
	gt	String	read-create	-	
	spc	Pointcode	read-write	-	
	ssn	Integer	read-write	-	
	newgt	String	read-write	-	
	xlate_id	String	read-create	-	
	entrytype	Settype	read-create	PRIMARY/SECONDARY	
gt	gt	String	read-create	-	ADD, DELETE, DISPLAY, MODIFY
	gtie	Integer	read-create	-	
	natofaddr	Integer	read-create	-	
	trtype	Integer	read-create	-	
	addrinfo	String	read-create	-	
	loadshare	Settype	read-write	YES/NO	
connection	id	Integer	read-only	-	DISPLAY
	state	String	read-only	-	

Table 3-7: ISUP Branch Managed Object Descriptions

Managed Object	Parameters	Types	Access	Operations
Isup	cfgname	String	read-create	MODIFY DISPLAY
	variant	String	read-write	
	mntcind	Settype	read-write	
	congestion	Settype	read-write *(not accessible in ANSI)*	
	xlate	Settype	read-write	
	recmode	Integer	read-write	

Table 3-7: ISUP Branch Managed Object Descriptions (Continued)

Managed Object	Parameters	Types	Access	Operations
isupnode	pcno	Integer	read-create	ADD DELETE MODIFY DISPLAY
	dpc	PointCode	read-write	
	congestion	Integer	read-only	
	access	String	read-only	
	ANMOFF	Integer	read-write	
	ACMOFF	Integer	read-write	
	CRGOFF	Integer	read-write	
	ciccontrol	Settype	read-write	
isupcgrp	pcno	Integer	read-create	ADD DELETE DISPLAY MODIFY
	dpc	PointCode	read-only	
	grpId	Integer	read-create	
	cctNum	Integer	read-write	
	trnkGrpId	Integer	read-write	
	scgaind	Settype	read-write *(not accessible in CCITT)*	
	ccname	String	read-only	
	mntcname	String	read-only	
isupcct	pcno	Integer	read-create	ADD DELETE DISPLAY MODIFY
	dpc	PointCode	read-only	
	grpId	Integer	read-create	
	cctNum	Integer	read-create	
	range	Integer	write-only	
	status	Integer	read-only	
	mtcstatus	String	read-only	
	hwdstatus	String	read-only	
	susstatus	String	read-only	
	opestate	Settype	write-only	
isuptmr	timerid	Integer	read-create	DISPLAY MODIFY
	value	Integer	read-write	

3.4 Distributed System Characteristics

The key characteristics that are commonly used in evaluating the overall usefulness of a distributed system solution are:

- Resource Sharing on page 3-28
- Reliability on page 3-29

- Scalability on page 3-32
- Transparency on page 3-39
- Performance Considerations on page 3-45

The following sections provide a brief description of the Distributed system characteristics and explain the commonly used methods to achieve them.

3.4.1 Resource Sharing

In a distributed system, hardware and/or software entities that can be shared by users across the network must be identified. Also, the means of accessing these shared resources in a reliable and consistent manner — especially when concurrent access to such resources is permitted — must be determined.

3.4.2 Reliability

The two aspects of reliability are *High Availability* and *Fault Tolerance*.

3.4.2.1 High Availability

High Availability refers to the fraction of time during which the system is usable. The two ways availability can be enhanced are:

- Design
  It is advantageous to design a system that does not require simultaneous functioning of many components.
- Redundancy
  Replicating key components of system hardware and/or software ensures that the system continues to function properly even if some components fail.

High availability is necessary but not sufficient. Ideally, data stored on a distributed system must not be lost and/or garbled. Furthermore, if the data is stored on multiple hosts, individual copies of the data must be consistent throughout the constellation. Generally, the more copies of the data there are, the better the availability, but the greater the chance that the data can become inconsistent. This is especially true of data environments in which there are frequent updates. Therefore, a second aspect of availability, discussed below, is fault tolerance.

3.4.2.2 Fault Tolerance

In general terms, fault tolerant systems are designed to mask failures, i.e., hide them from the users. When failure is unavoidable, fault tolerant systems fail in predictable ways. A description of all the predictable ways in which a fault tolerant system may fail is called the *predictable failure semantics* of the system.

The failure semantics of a system describe two conditions under which the system operates:

- Fail-safe mode – The failure is transparent to the users and the system contains built-in redundancy to recover.
- Fail-faulted mode – The failure is not transparent to the users, but the system can confine the failure and continue to operate in a somewhat degraded mode.

The design of fault-tolerant systems is based on two main approaches: *hardware redundancy*, i.e., the use of redundant components, and *software recovery*, i.e., the design of programs that can recover from faults.

Building systems that are tolerant to hardware failures can be costly because it involves the replication of critical system components. The software recovery approach, however, involves designing the software so that the state of the permanent system data can be recovered when a fault is detected.

Computations performed by a program are incomplete when a fault occurs, and the permanent data that the program updates may not be in a consistent state. Thus, the software recovery methods employed should restore the permanent data to the state it was in before the failed program started its execution.

An additional requirement on software fault-tolerance is the real-time needs of the users of a system. Generally, user needs prohibit time-consuming recovery methods. Implementing fault tolerant systems with real-time requirements in software therefore involves deploying tightly-coupled servers. In this way, users do not notice the loss of a partial number of servers beyond some performance degradation. Of course, this solution calls for cooperation among multiple servers so that substantial overhead to the system is not added in the normal circumstances, i.e., when everything is functioning correctly.

3.4.3 Scalability

Scalability means that the system/application software should not need to change when the scale of the system increases. The demand for scalability in distributed systems can be addressed effectively by a design philosophy in which no single hardware and/or software resource is assumed to be in restricted supply. Rather, as the demand for a resource grows, it should be possible to extend the system to meet it. One guiding principle in building scalable distributed systems is to avoid centralized components, tables, and algorithms.

3.4.4 Transparency

Transparency means concealing the separate components of a distributed system from the user and the application programmer. This way, the system is perceived as a whole rather than as a collection of independent components. Transparency has a number of characteristics, some of which are listed below.

- *access transparency* - enables both local and remote objects to be accessed using identical operations.
- *location transparency* - enables objects to be accessed without knowledge of their location, i.e., the host on which they are located.
- *concurrence transparency* - enables several processes to operate concurrently using shared objects without interference between them.
- *replication transparency* - enables multiple instances of objects to be used — increasing reliability and performance — without knowledge of the replicas by users or application programs.
- *failure transparency* - enables the concealment of faults, allowing users and application programs to complete their tasks in the event of hardware and/or software component failure.
- *scaling transparency* - allows the system and application programs to expand in scale without change to the system structure or the application algorithms.
- *growth/retrofit transparency* - allows parts of the system hardware and/or software to be replaced without requiring the overall system to be taken out of service.

Note: Transparency hides from users and renders anonymous those resources that are not of direct relevance and/or importance to the task-at-hand.

3.4.5 Performance

As a general index, when running an application program under a distributed system configuration, performance should not be appreciably worse than running the same application in a stand-alone configuration. The various metrics of performance for Distributed7are:

- Response times
- Throughput
- Utilization of system resources
- Amount of network capacity consumed

Most Distributed7 applications are real-time applications and require responses within specified time intervals. A response that is too late, e.g., due to an overloaded system, is viewed as a performance failure that should be avoided. A widely-used method of improving the overall performance in a distributed system is *concurrence*, in which multiple instances of system/application programs may execute at the same time.

3.5 Core Product Specifications

3.5.1 Resource Sharing

Distributed7 supports interworking of hosts that are of homogeneous architecture, i.e., hosts that feature the same type of byte ordering. This release of the product supports Solaris 2.5.1 and later OS releases.

Note: Distributed7 does not currently support distributed system configurations comprising hosts that are of heterogeneous architecture.

Configuration management tools that are available as part of Distributed7 enable users to configure their operational environment easily by allowing them to specify the names of hosts to be included as part of the system. Once a system is configured, users can add new hosts to a distributed environment and/or delete existing hosts from the environment at a later time without halting the system software on other hosts in the network.

Distributed7 provides users with a wide range of flexibility when it comes to distributed SS7 protocol implementation.

Distributed7 does not require:

- Each host comprising a distributed Distributed7 environment to be equipped with identical layers of SS7 protocol software.
- Applications that are interested in accessing the SS7 network to be executing on hosts that are locally equipped with the signaling link hardware.

Users of Distributed7 can specify the SS7 protocol layers that should be running on each host within a network by following a well-defined set of procedures. This approach not only allows the CPU power of each host operating under a distributed environment to be utilized effectively, but also makes it possible to implement custom product configurations featuring high availability, fault tolerance, and enhanced performance.

3.5.2 Reliability

The following subsections describe how the core Distributed7 product deals with the two most important reliability issues:

- High availability
- Fault tolerance

A list of Distributed7 failure semantics is also provided.

3.5.2.1 Reliability through High Availability

Replicated Data

Whenever possible, Distributed7 relies on the use of replicated data. This is a key to the effectiveness of the Distributed7 product in a distributed environment. Replicated data ensures high availability, fault tolerance, and enhanced performance. All Distributed7 data distribution frameworks, i.e., Distributed Shared Memory (DSM), Distributed Kernel Memory (DKM), and Distributed Record Access (DRA) frameworks, are designed around this.

Each host under a distributed Distributed7 environment maintains local copies of all appropriate pieces of user/kernel-space data in the form of DSM, DKM, or DRA segments. When an attempt is made to retrieve a certain piece of user/kernel-space data, instead of contacting one or more remote hosts, Distributed7 consults with the replicated copy of the data available on the local host to retrieve the information requested. This approach drastically improves the response time of the overall system and results in an architecture that can survive individual host failures.

A key concern when dealing with replicated pieces of data is that of consistency, especially when manipulating the contents of replicated user/kernel-space data. To address this issue, all Distributed7 data distribution frameworks, i.e., DSM, DKM, and DRA, use customized *atomic commit protocols* — either all the machines are updated, or none of them is — when updating local copies of the data on multiple hosts. For more information about these frameworks, refer to *Chapter 5: User/Kernel-space Data Distribution Methods* in this manual.

Centralized Algorithms

Another design issue that effects availability in a direct manner is the use of centralized algorithms: Extensive use of such algorithms under a distributed environment is likely to result in poor performance and diminish overall system availability. Distributed7 avoids the use of centralized algorithms as much as possible. There are instances, however, when the use of centralized algorithms is unavoidable, e.g., during processing of a global registration request, or when acquiring an exclusive read-write lock across a DSM or DKM segment. The use of centralized algorithms within this product is limited to the absolute minimum, and used during selected operations that are not as time-sensitive.

3.5.2.2 Reliability through Fault Tolerance

Distributed7 incorporates a variety of a hardware redundancy and software recovery techniques to achieve fault tolerance in the distributed mode of operation. Highlights of these techniques are as follows:

- To achieve high reliability, Distributed7 relies on the use of connection-oriented TCP/IP protocol when exchanging information between individual host machines comprising a distributed environment.
- For those user applications that cannot tolerate a single-point-of-failure during inter-machine communications, i.e., LAN failures, Distributed7 supports redundant LAN configurations.
- Distributed7 features an optional heartbeat mechanism across TCP/IP connections established between the individual hosts. This mechanism continually monitors the health and usability of the TCP/IP connections. Capabilities are provided to take a pre-planned course of action if and when all TCP/IP connections to a specified host become unreliable and/or fail.
- Distributed7 allows multiple instantiations of the system software components both in user-space and kernel-space to run concurrent, multiple instances on a particular host or on different hosts. Thus, the failure of a particular instance does not pose a serious threat from the user's perspective because the other instances were designed to take over its responsibilities.

3.5.2.3 Failure Semantics

For inter-host communication, Distributed7 relies on availability of kernel-level TCP/IP connections between individual hosts comprising a distributed environment. Failures in communication over the LAN, therefore, have direct impact on the availability of the product as a whole. To help improve overall system availability, redundant LAN configurations are recommended.

Distributed7 was designed to survive individual host crashes. The DSM, DKM, and DRA frameworks that are available as part of the Distributed7 product were designed to cope with such conditions by aborting all active and/or pending read/write/lock operations initiated by applications running on the crashed host, and restoring the contents of the individual DSM/DKM/DRA segments into a consistent and usable state. Issues such as the introduction of a new host to an existing network, and late start-up of a host while the other hosts are in operation, are also addressed. Individual layers of the SS7 protocol stack, e.g., SCCP, TCAP, ISUP, offer enhanced services that allow the set of resources associated with application programs on crashed hosts to be adopted by other application programs, i.e., applications running on the surviving hosts.

3.5.2.4 LAN Failures

Partial or complete LAN isolations may result from unintentional cable disconnects, or from running UNIX network interface configuration/maintenance commands, i.e., ***ifconfig.*** Distributed7 handles them as follows:

- Single-LAN product configuration:
  LAN isolation on a particular host results in a forced system shutdown on that host and an appropriate set of ***etmod*** alarms. Any SS7 protocol-related recovery under LAN isolation of a particular host is performed by the hosts that remain intact.
- Dual-LAN product configuration, partial isolation of a host:
  Disabling one of the LAN interfaces results in generation of an ***etmod*** alarm, and is fully transparent to upper layers of Distributed7 software. If the LAN interface that was disabled was the active connection, then a switchover takes place and the standby interface, i.e., the interface that is not tampered with, becomes the active interface. When the disabled interface is re-enabled and the TCP/IP connection is re-established, the interface is marked as a standby connection. LAN switchovers can be identified by ***etmod*** alarms.
- Dual-LAN product configuration, complete isolation of a host:
  Disabling both LAN interfaces results in a forced shutdown on that host, and a set of ***etmod*** alarms. SS7 protocol-related recovery under complete LAN isolation of a particular host is performed by the hosts that remain intact.

3.5.3 Scalability

CCA OVERVIEW supports up to a maximum of eight (8) host machines that can be interconnected to form a distributed processing environment. It allows up to a maximum of eight (8) signaling points that can be realized on one or more hosts. The total number of UNIX processes that can register with the CCA OVERVIEW environment in the "global" sense is limited to 1024. Each application running under CCA OVERVIEW is can have a maximum of 64 instances.

Distributed7 makes a conscious effort to reduce the amount of static memory allocations within the kernel-resident system software, and exploits dynamic memory allocation mechanisms whenever possible, i.e., the DKM and DRA frameworks. This factor is a direct contributor to the scalability of the product.

3.5.4 Transparency

The transparency aspects of the core product are described below. Transparency aspects of Distributed7 with respect to the individual layers of the SS7 protocol are described later in this section.

3.5.4.1 Access Transparency

Distributed7 allows users to access and manipulate the operational parameters associated with the platform. Also, the individual protocol layers can be accessed through any configured host in an access transparent manner. Distributed7 allows users to control the system software operations through any host in the network. The only requirement is that all host machines running under a distributed Distributed7 environment are equipped, minimumally, with the basic Service Provider Module (SPM) and the associated Network Interface Modules. Most obvious examples of access transparency can be seen when one attempts to retrieve the list of processes executing under a distributed environment or start/stop the system software on a specified host within the network.

3.5.4.2 Location Transparency

Distributed7 allows application programs executing under a distributed environment to be addressed in a location transparent manner. For example, an application process can send messages to another process without specifying the host on which the destination process is executing. It is up to the Distributed7 system software to find the target host and deliver messages to the destination process on that host. Location transparent addressing methods are available for both SS7 objects — users of the SS7 protocol — and named objects, i.e. plain UNIX processes.

IPC Key

Under normal circumstances, location transparency is a desired feature when it comes to addressing objects under a distributed environment. There is one exception involving multiple instances of a particular process that may be running concurrently. To deal with this, Distributed7 provides a more direct form of addressing, i.e., IPC key based, that enables processes running under a distributed environment to be uniquely identified.

Another aspect of location transparency involves distributed process management. Distributed7 provides the capabilities to start/stop user-specified layers of system and/or application software under a distributed environment in a location transparent manner. Capabilities are also provided to start/stop software on selected hosts. In most cases, however, users need not know which hosts are executing specified pieces of software.

3.5.4.3 Concurrence Transparency

All service provider modules, i.e., STREAMS multiplexers, comprising the Distributed7 infrastructure are designed to support user concurrence as a built-in feature. Therefore, system and/or application processes running under the Distributed7 environment need not be concerned about the integrity of the overall system during their execution, provided that their interface to the system is through the Distributed7 API libraries.

3.5.4.4 Replication Transparency

Transparency involves data replication. To achieve high availability, Distributed7 makes use of replicated data within both user space and kernel space. The fact that a certain piece of data is replicated on several hosts is completely hidden from user applications, be they within the DSM, DKM, or DRA frameworks.

Another aspect of replication transparency involves software. Distributed7 permits multiple instantiations of system and/or application software to run concurrently. The purpose of multiple instantiations is to create system architectures that are fault tolerant. Fault tolerance comes through replication of critical system and/or application programs on different hosts, and through allowing programs to load-share, i.e., replicating them on the same host or on different hosts. The existence of multiple instances of system software is an internal issue, and is, therefore, transparent to user applications most of the time. Note however that users interested in building fault-tolerant system architectures may be required to instantiate more than one instance of selected pieces of system software on different host machines. The existence of multiple instances of application software must be of direct concern to the application itself and is likely to require some coordination among the individual instances, e.g., how to process messages received by the individual instances.

3.5.4.5 Failure Transparency

Distributed7 delivers failure transparency both with hardware redundancy and software recovery techniques.

An example of the Distributed7 hardware redundancy approach is support of redundant LAN configurations. The presence of a redundant LAN configuration, i.e., dual LAN, is completely transparent to user applications executing under the Distributed7 environment. This is because administration and maintenance of multiple TCP/IP connections between the individual hosts are performed by special-purpose daemon processes. It is only when all TCP/IP connections to a particular destination fail that user applications are informed about the unavailability of the corresponding destination.

To achieve failure transparency with software recovery techniques, Distributed7 requires multiple instantiations of the critical system components to run concurrently i.e., each instance running on a different host, when providing services to user applications. For an example of how that is achieved, refer to *Failure Transparency on page 3-44*.

3.5.4.6 Growth/Retrofit Transparency

Distributed7 allows installation of a new release of the product while the system is in use. This is useful because it allows customers to test/verify the contents of a new release, i.e., patch, on a limited number of hosts before upgrading all hosts in the network. For more information about live software upgrade, refer to Live Upgrade of Distributed7 in the *Installation and Maintence Manual*.

3.6 Product Specifications

This release provides a *reliable* and *scalable* distributed computing environment that allows the individual layers of the SS7 protocol stack to be distributed across multiple hosts in a *flexible* manner.

Distributed7 supports distributed operations of the following layers of the SS7 protocol stack:

- Message Transfer Part (MTP), Level 2 and Level 3
- Signaling Connection Control Part (SCCP)
- Transaction Capabilities Application Part (TCAP)
- ISDN User Part (ISUP)

3.6.1 MTP Layer Product Specifications

3.6.1.1 Resource Sharing

Flexibility of the Distributed7 MTP protocol layers involve the following:

- Allows signaling links for a particular link set to be distributed across a multitude of hosts, and so yields flexible system configurations.
- Allows MTP L2 and L3 protocols to be separated from each other: L2 protocol is needed only on hosts that are equipped with the signaling link hardware, and L3 protocol is needed on all hosts where MTP user parts, e.g., SCCP and ISUP, exist.
- Allows operational parameters associated with the MTP L2 and L3 protocols to be accessed and/or manipulated through any host in the network, and so provides flexibility in system administration/maintenance.
- Provides run-time support for ANSI and ITU variants of the MTP L2 and L3 protocols.

3.6.1.2 Reliability

Implementation of the MTP L2/L3 protocols reliability addresses high availability and fault tolerance. The conditions under which the MTP layer software may fail to function properly are listed below.

High Availability

To achieve high availability, the Distributed7 implementation of the MTP protocol layers relies on information about the following entities to be replicated on all host machines that are locally equipped with the MTP L3 software:

- signaling points
- signaling links
- signaling link sets
- signaling routes
- signaling route sets

The replication task involves synchronization of both user-space and kernel-space data items between the individual hosts. Because all critical pieces of MTP protocol data are maintained in the kernel-space, this task falls under the responsibility of the newly introduced DKM and DRA frameworks, and is coordinated by the MTP protocol layers.

Whenever possible, the functions of MTP L3 on a particular host utilize the local copy of the data available on that host to perform their tasks, e.g., routing decisions made by the MTP L3 Signaling Message Handling (SMH) functions. Failures on remote hosts therefore do not interfere with the operations of the MTP L3 software on the local host, and the availability of the system as a whole is increased. The use of local data — as opposed to using centralized copies of the data — improves the overall system performance and is essential for implementing a fault-tolerant system architecture.

The only centralized component in a distributed system configuration involves the MTP L3 SNM functionality, which is essential to guarantee orderly processing of messages received from the SS7 network (or initiated by the user) regarding the MTP network management functionality. More information about the Distributed7 network management functionality is provided in the following subsection.

Fault Tolerance

The Distributed7 implementation of the MTP L3 SNM functions contain built-in recovery procedures. This provides continued network management functionality in the case of partial failures, e.g., individual host crashes. The software recovery procedures employed by the MTP-L3 are based on the primary/backup server model. This means that the MTP functions in a distributed environment are provided through multiple hosts, i.e., through multiple instances of the MTP-L3 software.

Under normal circumstances, all network management functions are performed by the primary instance of the MTP-L3, with the backup instances having sufficient information to take over if and when the primary instance fails.When the primary instance of the MTP-L3 fails, one of its backup instances takes over as the primary.

Failure Semantics

While operating under Distributed7, messages submitted by MTP user parts may not be transmitted to the SS7 network under the following circumstances:

- A fatal error is encountered in the SS7 signaling link hardware while there are messages in the transmit buffer of the board. These errors may result from failures of individual hardware components on the SS7 board, e.g., CPU, on-board memory, ports, and/or failures of Distributed7 device drivers or board-resident software such as the board operating system or MTP L2 software.
- A fatal error is experienced on the local host while messages are in transition. These errors may result from hard/soft system resets, failure of critical hardware components, e.g., CPU, on-board memory, disk drivers, or failure of operating system and/or Distributed7 kernel-resident software.

- Messages submitted need to be transmitted over signaling links that are located on remote hosts, and a fatal error is experienced on the local host and/or one of the remote hosts while messages are in transition.
- Messages submitted need to be transmitted over signaling links that are located on remote hosts, and a single/dual-LAN failure is encountered while messages are in transition. LAN failures may result from hardware/software failures in the LAN interface cards, hardware/software failures in LAN components such as hubs and bridges, or hardware failures in LAN cables and/or LAN interfaces.

Messages received from the SS7 network may not be delivered to their final destination if one of the following events occurs:

- A fatal error is encountered in the SS7 signaling link hardware while there are messages in the receive buffer of the board.
- A fatal error is experienced on the local host while messages are in transition.
- Messages are destined to a user on a remote host, and a fatal error is experienced on the local host and/or remote host while messages are in transition.
- Messages are destined to a user on a remote host, and a LAN failure is encountered while messages are in transition.

The Distributed7 product is capable of detecting the LAN hardware/software failures within a time interval that is at most five times longer than the programmable TCP/IP heartbeat interval. The longer this time interval is set to, the higher the total number will be of lost inter-host messages when a fatal error is encountered in the LAN hardware/software.

3.6.1.3 Scalability

For scalability reasons, Distributed7 dynamically allocates all internal resources associated with the individual MTP user parts for a specified signaling point, i.e., resources are not allocated until the corresponding signaling point and/or the user part is instantiated.

3.6.1.4 Transparency

Transparency aspects of the MTP protocol layers are:

- Access transparency
- Location transparency
- Concurrence transparency
- Replication transparency
- Failure transparency

Access Transparency

Distributed7 provides full access transparency when retrieving/manipulating information regarding operational parameters associated with the MTP protocol layers as long as access to this information is with the officially supported Distributed7 management interfaces, e.g., MML and AccessMOB.

Location Transparency

Under Distributed7, all MTP L2/L3 parameters except for those associated with signaling link hardware can be accessed and manipulated in a location transparent manner. To access and/or manipulate parameters associated with the signaling link hardware, users need to specify the name of the host machine on which the signaling link hardware is installed.

Another aspect of location transparency involves the start-up/termination of the MTP related software on individual hosts. Under Distributed7, users can start-up/stop the MTP L3 functions across many host machines in a location transparent manner. Alternatively, they can start-up/stop selected portions of the MTP L3 software on specified hosts. MTP L2 software can also be started or stopped either in a location transparent manner or by specifying the host name and board number information.

From a users point of view, distribution of the MTP functionality across a network of hosts is to a great extent location transparent. While Distributed7 requires that all hosts on which MTP users exist be equipped with the MTP L3 software, it does not require all such hosts to feature local signaling link hardware. This approach enables MTP user parts to exchange messages with the SS7 network without knowing which signaling link hardware is being used to transmit and receive these messages.

Concurrence Transparency

Distributed7 allows multiple MTP users, e.g., SCCP, TCAP, ISUP, as well as multiple instantiations of a particular MTP user, to exchange messages through the SS7 network in a concurrent manner. All critical pieces of data used by the MTP L2/L3 software are internally protected, and users of the MTP protocol can therefore interact with the network concurrently without any interference among them.

Replication Transparency

Under a distributed environment, all critical pieces of MTP L3 information are replicated on all hosts equipped with the MTP L3 software through the use of DKM and DRA infrastructures. This includes information about signaling points, signaling links and link sets, signaling routes and route sets, and signaling network management function states. Replicated MTP data contains both configuration related information, e.g., the number of signaling links within a link set, and information of a more dynamic nature, e.g., changes in the state of a signaling link.

The Distributed7 implementation of the MTP protocol layers features built-in intelligence to keep the replicated copies of DKM and DRA data appropriately synchronized. This mechanism is completely transparent to the users of the MTP.

Failure Transparency

From a user perspective, Distributed7 supports failure transparency by allowing multiple instantiations of the MTP Signaling Network Management (SNM) functions to execute concurrently in the primary/backup server mode.

3.6.1.5 Performance Considerations

The highlights of the Distributed7 implementation of the MTP protocol layers in regard to performance are as follows:

- Distributed7 makes use of kernel-level TCP/IP connections to convey messages between hosts; therefore, it is capable of meeting the transfer times specified in Recommendation Q.706 when sending messages across signaling links located on remote hosts.
- MTP L3 changeover/changeback scenarios that involve signaling links located on different hosts meet the time limits specified in Recommendation Q.706.
- To achieve increased throughput, Distributed7 allows multiple users — executing on a particular host or different hosts — to submit/receive MTP L3 messages through the SS7 network in a concurrent manner.
- To achieve increased throughput, Distributed7 allows identical copies of the MTP L3 SMH functions to execute concurrently on different hosts.
- The implementation of MTP L3 software recovery procedures is designed so that backup instances remain mostly idle. This allows the CPU power on the hosts where backup instances run to be used for other computational needs.

3.6.2 SCCP Layer Product Specifications

3.6.2.1 Resource Sharing

Flexibility aspects of the Distributed7 SCCP protocol layer allow:

- SCCP subsystems/applications executing on a host that is not locally equipped with the signaling link hardware to exchange messages with the SS7 network using signaling links located on remote hosts
- Operational parameters associated with the SCCP protocol layer to be accessed and/or manipulated through any host in the network, i.e., flexibility in system administration/ maintenance
- Run-time support for ANSI and ITU variants of the SCCP protocol

3.6.2.2 Reliability

The following subsections describe how the implementation of the SCCP protocol addresses the reliability issues of high availability, fault tolerance, and failure semantics.

High Availability

To achieve high availability, the implementation of the SCCP protocol layer relies on critical pieces of information about all SCCP subsystems and/or applications to be replicated on all host machines that are locally equipped with the SCCP layer software. The replication task is performed by the DKM and DRA frameworks on behalf of the kernel-resident SCCP module. Whenever possible, the SCCP functions on a particular host use the local copy of data that is available on that host to perform its tasks, e.g., routing decisions made by the SCCP protocol layer. Thus, failures in remote host machines do not impact the operations of the SCCP protocol software on the local host, increasing the availability of the system as a whole.

The only centralized component under a distributed system configuration involves the SCCP management (SCMG) functionality. This is essential to guarantee an orderly evaluation and processing of SCCP management events. At any particular point in time, all management tasks are performed by a particular instance of the SCCP software on one host. The selection of this instance is fully dynamic, i.e., if the host goes out of service, then the management tasks are performed by a particular instance of the SCCP software on another host in the distributed system configuration.

Fault Tolerance

This implementation of the SCMG functionality features built-in recovery procedures. This is essential to provide continued management functionality in the case of partial failures, e.g., individual host crashes. The software recovery procedures employed by the SCMG layer are based on the primary/backup server model described in Fault Tolerance on page 3-42. For fault-tolerance reasons, the SCMG functions under a distributed Distributed7 environment are provided through multiple hosts with multiple instantiations of the SCCP software. Under normal circumstances, all SCMG functions are performed by the primary instance of the SCCP software, with the backup instances having sufficient information to take over if and when the primary instance fails.

This implementation of the SCCP protocol features built-in recovery procedures that allow application programs to recover from individual host crashes without losing the SCCP connections set up through the crashed hosts. More information on these procedures is provided in Failure Transparency on page 3-48.

Fault-tolerant Distributed7 system architectures require a minimum of two instances of the SCCP software (per signaling point) to be running, i.e., in the primary/backup mode, on two different hosts. System configurations that comprise a single instance of the SCCP software are not fault-tolerant.

Failure Semantics

Being an MTP user part, the SCCP protocol layer is directly effected by the failure semantics described in Failure Semantics on page 3-42.

3.6.2.3 Scalability

For scalability reasons, Distributed7 allocates all internal resources associated with the individual SCCP subsystems/applications in a dynamic manner, i.e., resources are not allocated until the corresponding subsystem is instantiated.

3.6.2.4 Transparency

The following subsections describe the transparency aspects of the SCCP protocol layer.

Access Transparency

Provided that access to the information is through the officially supported Distributed7 management interfaces, i.e., MML and AccessMOB, Distributed7 provides full access transparency when retrieving/manipulating information regarding operational parameters associated with the SCCP protocol.

Location Transparency

Under Distributed7, operational parameters associated with the SCCP protocol layer can be accessed and manipulated in a location transparent manner, i.e., users need not know which hosts are equipped with and/or running the SCCP layer software.

Another aspect of location transparency involves the start-up/termination of the SCCP layer software on the individual hosts. Under Distributed7, users can start-up/stop the SCCP software across a multitude of host machines in a location transparent manner. For more controlled operations, Distributed7 also provides the means to start-up/stop the SCCP software on a specified host or set of hosts.

Concurrence Transparency

Distributed7 allows multiple SCCP subsystems as well as multiple instantiations of a particular subsystem to exchange messages through the SS7 network in a concurrent manner. All critical pieces of data used by the SCCP software are internally protected; thus, users of the SCCP protocol can interact with the network concurrently without any interference among them.

Replication Transparency

Under a distributed environment, all critical pieces of SCCP protocol information are replicated on all host machines that are equipped with the SCCP layer software, using the DKM and DRA frameworks. This includes information about subsystems as well as information about the class of SCCP service being provided, i.e., in the case of connection-oriented services.

This implementation of the SCCP protocol layers features built-in intelligence to keep the replicated copies of DKM and DRA data in sync at all appropriate times. This mechanism is completely transparent to the SCCP applications.

Failure Transparency

From a users perspective, Distributed7 supports the concept of failure transparency by allowing multiple instantiations of the SCCP software to run concurrently. Specifically, the failure transparency characteristics of the SCCP layer are as follows:

- All critical pieces of information about the SCMG functions are replicated on all hosts equipped with the SCCP layer.
- All critical pieces of information about the connection-oriented SCCP services are replicated on all hosts equipped with the SCCP layer. This ensures that in the case of an individual host crash, the connections that are established through SCCP applications running on that host can be assigned to other applications running on the surviving hosts.
- When operating under a distributed Distributed7 environment, information about the connectionless SCCP services is not replicated on the individual hosts. Thus, it is not possible to provide fault-tolerant connectionless SCCP services using the Distributed7 product. This should be viewed as an acceptable mode of operation however, due to the unreliable nature of the “connectionless” communication protocols in general.

3.6.3 TCAP Product Specifications

3.6.3.1 Flexibility

Flexibility aspects of the TCAP (TC) layer are:

- Allows TC applications running on a host that is not locally equipped with the signaling link hardware to exchange messages with the SS7 network using signaling links located on remote hosts.
- Supports front-end/back-end system configurations when the MTP and SCCP layers are run on one or more front-end hosts, and the TCAP layer as well as the TC application are run on one or more back-end hosts.
- Allows TC applications to specify the underlying transport service provider to be used for TCAP message transportation. Users can currently select between the SCCP and TCP/IP transport protocols.
- Supports TCP/IP connectivity to 3rd-party hosts that are not equipped with the Distributed7 software for exchanging TCAP messages with them.
- Provides run-time support for ANSI and ITU variants of the TCAP protocol.
- Allows hybrid stacks to be built in which the TCAP protocol may differ from that of the MTP and SCCP layers.
- Allows a particular TC application to exchange TCAP messages with other applications through multiple service endpoints. In this case the user application must specify the underlying transport service provider as well as the variant of TCAP protocol to be used for each service endpoint.

3.6.3.2 Reliability

The following sections describe how the implementation of the TCAP protocol addresses reliability issues such as high availability and fault tolerance.

High Availability

This implementation of the TCAP protocol layer does not rely on the use of any centralized data. Information about all active transactions initiated by a TC application is replicated on all hosts that are equipped with the TCAP software and feature another instance of that TC application. Thus, failures in remote host machines do not impact the operations of the TC applications on the local host, increasing the availability of the system as a whole.

All kernel-space data used by the TCAP transaction layer is implemented in the form of DKM segments and, under normal circumstances, the scope of this data is limited to the local host. It is only in the case of a failure that the need arises to access the replicated copies of the data manipulated through a remote host. This latter feature requires almost no extra time due to the off-line synchronization capacity of the DKM framework.

Fault Tolerance

This implementation of the TCAP protocol features built-in recovery procedures that allow application programs to recover from individual host crashes without losing the active transactions on the crashed hosts. More information on these procedures is provided in Failure Transparency on page 3-51.

Note: *Fault-tolerant Distributed7 system architectures require a minimum of two instances of each TC application to be executing (in load-shared mode) on two different hosts. System configurations that comprise a single instance of a TC application, or multiple instances of a TC application all executing on the same host, are not fault-tolerant.*

Failure Semantics

When the transport services provided by the SCCP protocol are in use, the TCAP protocol layer is directly effected by the failure semantics described in Failure Semantics on page 3-47. When the transport services provided by the TCP/IP protocol are in use, failures in communication over the LAN have direct impact on the correct operations of the TC applications involved.

Important*: The TCAP implementation features no built-in software recovery intelligence at the TCAP component-handling layer. If such functionality is desired, then it needs to be implemented by the TC application itself. TC applications may take advantage of the powerful user/kernel-space data distribution frameworks that are available as part of Distributed7 to implement customized recovery strategies.*

3.6.3.3 Scalability

Distributed7 allows up to a maximum of 16 different TC applications to be executing on a particular host, with each TC application having up to 63 instances executing concurrently (in load-shared mode). Each TC application may carry out up to 65536 active transactions at a given time.

Note: *The maximum number of instances for a particular TC application permitted across a network is limited to 63 * n, where n is the number of hosts in the network.*

3.6.3.4 Transparency

The following sections describe the transparency aspects of the TCAP protocol layer.

Access Transparency

The TCAP implementation features a small yet powerful set of command-line utilities that can be used to retrieve various pieces of information regarding the TC applications running under a distributed Distributed7 environment, and/or configure various parameters used by the TCAP transaction layer.

Location Transparency

Under Distributed7, users can start-up/stop the TCAP layer software across a multitude of hosts in a location transparent manner. For more controlled operations, Distributed7 also provides the means to start-up/stop the TCAP layer software on a specified host or set of hosts.

Concurrence Transparency

Distributed7 allows multiple TC applications to execute in a concurrent manner. All critical pieces of data used by the TCAP software are internally protected; thus, TC applications can exchange messages with other applications concurrently without any interference among them.

Replication Transparency

Under a distributed environment, all critical pieces of TCAP protocol information are replicated on all host machines that are equipped with the TCAP layer software and have active TC applications running on them, using the DKM framework.

This implementation of the TCAP transaction layer features built-in intelligence to keep the replicated copies of all DKM data synchronized at all appropriate times. This mechanism is completely transparent to the TC applications.

Failure Transparency

From a user's perspective, Distributed7 supports the concept of failure transparency by allowing multiple instantiations of a TC application to run concurrently. Specifically, the failure transparency characteristics of the TCAP layer are as follows:

All critical pieces of information about the active TCAP transactions is replicated on all hosts equipped with the TCAP layer. This ensures that in the case of an individual host crash, the transactions that are set up with TC applications running on that host can be assigned to other instances of that application running on the surviving hosts.

3.6.4 ISUP Product Specifications

3.6.4.1 Flexibility

Flexibility aspects of the ISUP protocol layer are:

- Allows users of the ISUP protocol layer to execute on a host that is not equipped with the signaling link hardware
- Supports front-end/back-end system configurations in which the MTP and ISUP layers are run on one or more front-end hosts and the users of the ISUP layer are run on one or more back-end hosts
- Allows operational parameters associated with the ISUP protocol layer to be accessed and/or manipulated through any host in the network; thus, provides flexibility in system administration/maintenance
- Provides run-time support for ANSI and ITU variants of the ISUP protocol
- Supports instantiation of multiple call control instances on the same and/or multiple hosts

3.6.4.2 Reliability

The following sections describe how the Distributed7 implementation of the ISDN User Part (ISUP) protocol addresses reliability issues such as high availability and fault tolerance.

High Availability

To achieve high availability, the Distributed7 implementation of the ISUP protocol layer relies on critical pieces of information about ISUP circuit groups and circuits within each circuit group to be replicated on all hosts that are locally equipped with the ISUP layer software. Whenever possible, ISUP functions on a specified host use the local copy of data available on that host to perform their tasks. Thus, failures in remote hosts do not impact the operations of the ISUP software on the local host, increasing the availability of the system as a whole. Distributed7 ISUP implementation relies on the DSM and DKM frameworks for user/kernel-space data distribution.

Fault Tolerance

This implementation of the ISUP protocol features built-in recovery procedures that allow application programs to recover from individual host crashes without losing the stable calls set up through crashed hosts. More information on these procedures is provided in Failure Transparency on page 3-54.

Note: *Fault-tolerant system architectures require a minimum of two instances of the ISUP software per signaling point (in load-shared mode) on at least two different host machines. System configurations that comprise a single instance of the ISUP software are not fault-tolerant.*

Failure Semantics

Being an MTP user part, the ISUP protocol layer is directly effected by the failure semantics described in Failure Semantics on page 3-42.

3.6.4.3 Scalability

Distributed7 supports a maximum of 2048 ISUP destinations. The total number of trunks that may be configured across a network of hosts is also limited to 8192. There are no other inherent limitations with respect to the number of trunks that may be connected to a particular ISUP destination.

For scalability reasons, CCA OVERVIEW allocates all internal resources associated with the ISUP protocol data in a dynamic manner. User-space data maintained by the ISUP daemon process is stored in the form of DSM segments. These DSM segments are dynamically created when the first instance of the ISUP daemon process for a specified signaling point is started, and destroyed when all instances of the ISUP daemon process for a specified signaling point are terminated. ISUP implementation relies on the DKM framework to create and store its kernel-resident data.

3.6.4.4 Transparency

The following sections describe the transparency aspects of the ISUP protocol layer.

Access Transparency

CCA OVERVIEW provides full access transparency when retrieving/manipulating information regarding operational parameters associated with the ISUP protocol layer as long as access to this information is through the officially supported CCA OVERVIEW management interfaces, e.g., MML and AccessMOB.

Location Transparency

Under CCA OVERVIEW, operational parameters associated with the ISUP protocol layer can be accessed and manipulated in a location transparent manner, i.e., users need not know which hosts are equipped with and/or running the ISUP layer software.

Another aspect of location transparency involves the start-up/termination of the ISUP layer software on the individual hosts. Under CCA OVERVIEW, users can start-up/stop the ISUP software across a multitude of host machines in a location transparent manner. For more controlled operations, CCA OVERVIEW also provides the means to start-up/stop the ISUP software on a specified host or set of hosts.

Concurrence Transparency

CCA OVERVIEW allows multiple instantiations of the ISUP layer software, for a specified signaling point, to execute concurrently on multiple hosts within a distributed CCA OVERVIEW environment. This capacity for multiple instance instantiation, with software redundancy, supports fault-tolerant system configurations.

Circuit supervision events that occur while operating under a distributed environment are always processed by the primary instance of the ISUP software — in case the ISUP layer for the corresponding signaling point features multiple instances. The selection of the primary ISUP instance is fully dynamic.

Replication Transparency

Under a distributed environment, all critical pieces of ISUP protocol information are replicated on all host machines that are equipped with the ISUP layer software, using the DSM and DKM frameworks. This includes information about individual circuits groups, circuits within each circuit group, and protocol timers.

Implementation of the ISUP protocol layer features built-in intelligence to keep the replicated copies of all DSM and DKM data synchronized at all appropriate times. This mechanism is completely transparent to the ISUP users.

Failure Transparency

From a user's perspective, CCA OVERVIEW supports the concept of failure transparency by allowing multiple instantiations of the ISUP protocol software to execute concurrently. All critical pieces of circuit information are replicated on all hosts equipped with the ISUP layer, meaning that in the case of an individual host crash, the ownership of the circuits allocated by application programs running on that host can be assigned to other applications running on the surviving hosts.

Chapter 4: Distributed System Operations

4.1 Introduction

This chapter describes the distributed operations of the following protocol layers of Distributed7:

- Message Transfer Part (MTP)
- Signaling Connection Control Part (SCCP)
- Transaction Capabilities Application Part (TCAP)
- ISDN User Part (ISUP)

4.2 Message Transfer Part (MTP)

This section describes the distributed operations of the Message Transfer Part (MTP) protocol layer of Distributed7.

In the Distributed7 stack, a signaling point can be configured as one or more hosts; some having MTP/L3 function, some having MTP/L2 function or both. From the network point of view, all hosts constructing the distributed environment are considered as a single signaling point.

4.2.1 Multiple Instance Support

MTP/L2 software runs in the SS7 hardware and has a message-based interaction with MTP/L3 software. MTP/L2 software manages the SS7 links that are physically connected to the machine. SS7 links do not have multiple instances either on the same host or on a different host. This is to say that, any failure (software/hardware) on MTP/L2 causes the SS7 link to fail.

MTP/L3 software, on the other hand, supports multiple instances. Each hosts can run only one instance of MTP/L3 for a specific signaling point. Out of the MTP/L3 instances on the network, only one of them performs the Signaling Network Management (SNM) function. This instance is called the primary MTP/L3 instance and runs on the host where the primary MTP/L3 daemon (***upmd***) is running. The Signaling Message Handling (SMH) function of MTP/L3 protocol, on the other hand, runs on the all the hosts where MTP/L3 software is started.

MTP/L2 software can run on any host that has SS7 hardware and does not need MTP/L3 software running on the same host. However, there must be at least one instance of MTP/L3 software running on the network.

User parts (SCCP, ISUP) need MTP/L3 software running on the same hosts. This is to say that before starting a user part on a host, MTP/L3 software must be started on the same host.

Figure 4-1 illustrates multiple instance support of MTP/L2 and MTP/L3 software in a Distributed7 environment.

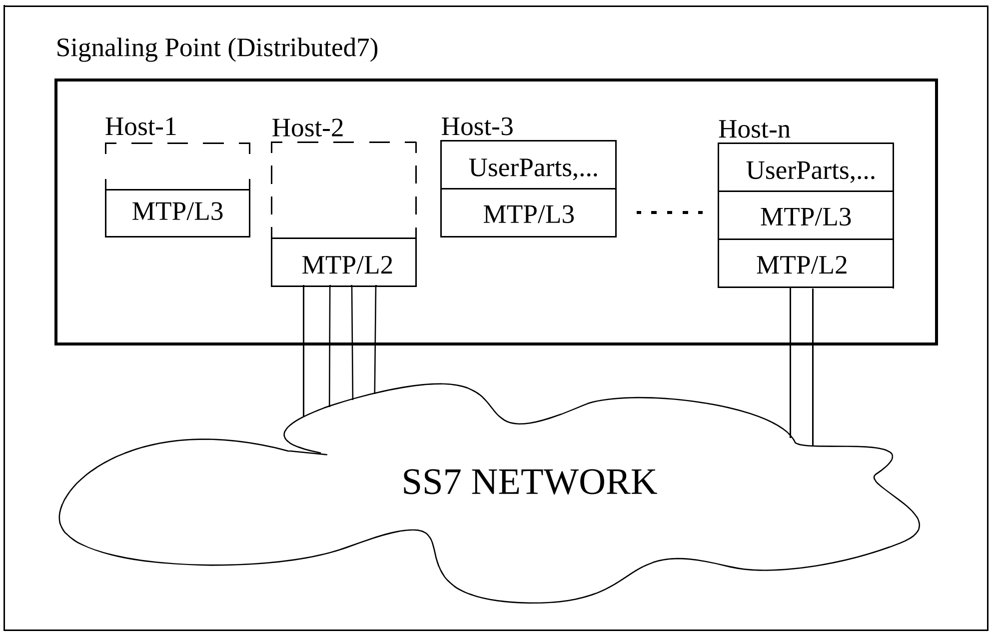


Figure 4-1: Multiple instance support of MTP software

4.2.2 Message Distribution and Routing

The three different "message routing" mechanisms in the MTP layer are:

- Routing of Configuration Messages
- Routing of Incoming SS7 Messages (from SS7 network)
- Routing of Outgoing SS7 Messages (to SS7 network)

4.2.2.1 Routing of Configuration Messages

All configuration related messages are destined for the primary instance of the ***upmd*** process. This instance handles the message and replies to the originator. If the message is an update message (add, delete, modify) then the primary instance broadcasts the message to the other ***upmd*** processes on the network.

4.2.2.2 Routing of Incoming SS7 Messages

All incoming messages coming from MTP/L2 software are destined for the local MTP/L3 software, if it exists. If the local host does not have MTP/L3 software, then the next host that has the MTP/L3 software running is found and the message is sent to that host.

Upon receiving an incoming message, MTP/L3 software checks the service indicator of the message. If the message is an SNM message, then it is forwarded to the host where the primary MTP/L3 is running. If the service indicator is for a user part, then the distribution method for the user parts, which is set upon binding to MTP/L3, is applied:

- LOADSHARE – Loadshare incoming messages between the active users
- ROUTE TO CLOSEST – Send the incoming message to the user on the local host (if any); otherwise send the message to the next host in the host chain, i.e., each host has a unique number in the DISTRIBUTED7 network, and the host chain is ... first ... last ... first ...
- ROUTE TO MASTER – Send messages to the master instance of the user. The selection of the master is done by MTP/L3: The first bound user becomes the master. If the master user terminates, then the next in the chain becomes the master.

4.2.2.3 Routing of Outgoing SS7 Messages

Any message that is sent by a user part of MTP/L3 is routed to the SS7 network, and depends on the local routing data. The physical SS7 link selection is determined by the destination and the SLS value of the message. If the SS7 link hardware is not on the local host where the message is sent, then the message is forwarded to that host.

As a result, at a certain time, all messages sent to MTP/L3 on different hosts go through the same physical SS7 link if they have the same destination and SLS values. This provides a network-wide SS7 hardware loadshare.

4.2.3 Protocol Specific Issues

4.2.3.1 MTP Capacity and Protocols

MTP supports STP functionality which was not supported in the previous releases. It also supports the following protocols:

- ANSI 96
- ANSI 92
- ITU White Book (ITU-93)
- ANSI Bellcore (1991)
- ITU 1997
- ETSI 1997

The database of current MTP has the following limitations:

- Maximum number of linksets per sp: 64
- Maximum total number of links per sp: 256

***Note**: Only 255 links are available for use. One link is reserved for internal use for gateway functionality.*

- Maximum number of links per linkset: 128
- Maximum number of destinations (route set) per sp: 2048
- Maximum number of destinations behind a linkset: 2048
- Maximum number of routes per route set: 16
- Maximum number of load sharing linksets per destination: 2

4.2.3.2 SNM Procedures that Interact with Remote SMHs

The four SNM procedures in which SMH instances on all hosts are involved are:

- changeover
- changeback
- forced re-routing
- controlled re-routing

The routing data in each procedure changes. Before changing the routing data, traffic on the affected element(s) must be buffered until the new routing data is prepared. The routing operations are performed in the SMH portion of the UPM drivers, and the new routing information is prepared in the SNM portion of the UPM driver.

Whenever one of the four SNM procedures starts, the corresponding SNM task broadcasts an *M_buffer_message* to all *hmrt* tasks, i.e., routing tasks in UPM driver. The SNM task starts an internal timer (collect *hmrt* messages) and waits for the responses from *hmrt* tasks. Upon receiving the message, all *hmrt* tasks start buffering messages for this affected component. Later, the SNM task prepares the new routing information and broadcasts it to *hmrt* tasks with a *M_send_buffered_message*. *Hmrt* tasks then send the messages they had buffered and update the routing data.

In addition, changeover includes a message retrieval procedure from MTP/L2. In this procedure, the SNM task (changeover task running on the primary SNM host) sends a *M_retrieve_message* to MTP/L2, and MTP/L2 sends all messages in its transmit buffer to the *hmrt* task on the primary SNM host.

After completion of sending buffered messages, all *hmrt* tasks respond to the corresponding task on the primary host with a message indicating the termination of the process. The SNM task checks if all responses arrived and continues the normal operation. If the timer expires, this means that one of the *hmrt*'s could not complete the operation on time, then, the SNM task continues normal operation.

4.2.4 Data Distribution Methods

4.2.4.1 Data Model

MTP/L3 makes use of the DRA framework to maintain its kernel level data. All data structures are kept as hashed or sorted DRA segments. The segments of MTP/L3 data are organized as normalized databases and are shown in Figure 4-2 on page 6.

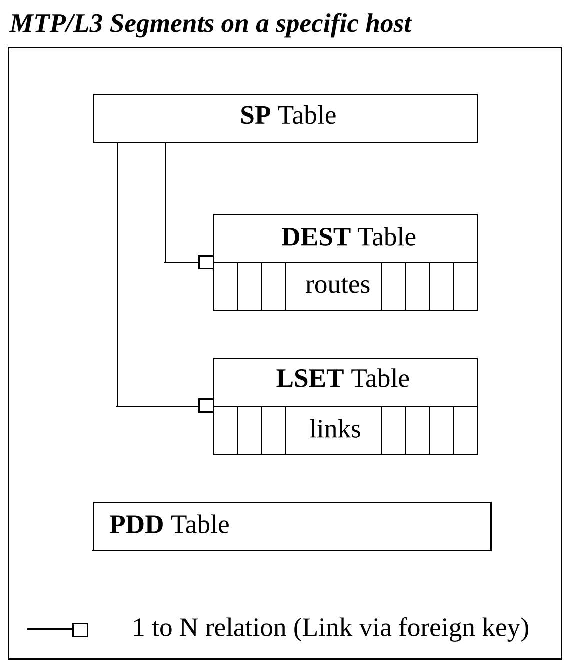


Figure 4-2: MTP/L3 Data Segments

PDD Table: Private Device Driver table, includes device-specific data for UPM multiplexer. Hashed DRA segment. Key is the clone device number.

SP Table: Signaling Point table, includes all SP related flags, timers and state information. Hashed DRA segment. Key is the signaling point number.

LSET Table: Link set table, includes all link Set related flags, timers, state information, and link array. Hashed DRA segment. Primary key is link set ID; secondary key is destination point code.

DEST Table: Destination table, includes destination (route set) related flags, timers, state information, and route array. Sorted DRA segment. Primary key is the network+cluster+member array; secondary key is destination point code.

4.2.5 MTP/L3 Recovery Procedures

There are recovery actions only when the primary MTP/L3 instance terminates, i.e., host crash or shutdown. In such cases, the two major recovery activities that must be performed are:

- Recovery of SNM task activities, if the crashed host was running the primary MTP/L3
- Handling of the SS7 links on the crashed host, if there are any.

The new MTP/L3 instance that takes over after a crash first performs the recovery of SNM tasks, and then immediately performs a changeover procedure for the links on the crashed hosts. If there is no ongoing procedure in the primary SNM, i.e., changeover, changeback, forced re-routing, and controlled re-routing, then there will be no recovery action in the new MTP/L3 instance that takes over.

4.2.5.1 Recovery Definition of SNM tasks

Since all SNM activities are management activities, it is concluded that the SNM tasks do not have a lot of work on a real network. Thus, the SNM activities are performed in a synchronized manner. This means that all modifications on MTP/L3 distributed data are synchronized immediately with the other MTP/L3 instances. This may lead to some performance decrease on SNM procedures. However, the DRA/DKM framework functions so fast that MTP/L3 can comply with the performance requirements of MTP standards (Recommendation Q.706 Message Transfer Part Signaling Performance).

Although all data is synchronized across the network, there are some other recovery related actions needed for SNM tasks.

4.2.5.2 Recovery of MTP instances

MTP/L3 software constructs its data in three different tables: *sp*, *lset* and *dest*. Links are constructed as array in the link set table. Operations on link instances indirectly affect the other components. Thus, recovering link data must be adequate enough to recover all MTP data. However, there are some network messages, e.g., transfer prohibited, transfer restricted, etc., that directly affect the *dest* instances. These kinds of messages are handled separately.

The general method to recover SNM tasks on individual MTP components is as follows;

1. Check the state of the component.
2. If the component is idle, then confirm the state by exchanging a message with the network or SS7 hardware.
3. If there is a long lasting timer that was started for this component, then restart the timer.
4. For short lasting timers, assume that the timer expired and use this information as an input to the recovery task.
5. Perform the recovery action by tracing the state of the component.

4.2.5.3 Detection of MTP/L2 failures

MTP/L3 software catches the M_DEV_REMOTE and M_DEV_LOCAL events to decide if an SS7 board failed or is connected to the environment:

If either event indicates a...	Then...
device attach,	MTP/L3 tries to start the links configured on that board, if any.
device detach,	MTP/L3 immediately deactivates the existing links on that board to start changeover procedures of SNM function.

4.2.6 Application Programming Interface

The MTP API library is unified to support different protocols, e.g., ANSI, ITU, from the same object code. ***libcmtp.a*** and ***libamtp.a*** libraries are merged into one object file called ***libmtp.a***.

Previous release MTP API functions are supported in this release. However, the structures of the function are different. This is because the old structures had members (pointcode) that depended on the protocol. In this release, there is one unique structure for all protocols. Users are able to parse this pointcode with the other newly introduced MTP API functions. The new MTP API library consists of the following set of functions:

mtp_init_api()
Initialize the MTP library for the given signaling point (sp).

Note: *This function* ***must*** *be called prior to using other MTP API functions.*

mtp_decode_pc()
Converts a string pointcode (str) into an integer value (pointcode). The format and protocol compatibility check is done within the API while using the internal MTP data of given signaling point (sp).

mtp_encode_pc()
Converts an integer pointcode (pointcode) into a string format (str). The output pointcode format will be as <X-Y-Z>.

mtp_flow_ind()
Extracts flow data from an incoming message having an MTP flow primitive. This function replaces the old MTP API calls: MTP_PauseInd, MTP_ResumeInd and MTP_StatusInd.

mtp_info_ind()
Extracts MTP level-3 specific data from an incoming message that carries the MTP_INFO primitive.

mtp_xfer_ind()
Parses an MSU and extracts MTP specific data into a user buffer. This function replaces the old MTP API call MTP_XferInd.

mtp_xfer_req()
Sends an SS7 message through a service endpoint, identified by *fd*, under the Distributed7 environment.

mtp_spc()
Retrieves the own signaling point code of the MTP layer-3, in the form of an integer.

mtp_ni()
Retrieves the network indicator of the MTP layer-3 of the signaling point.

mtp_up_offset()
Retrieves the offset of the user part message in an MSU.

mtp_protocol()
Retrieves the protocol type of the MTP layer-3 of the signaling point.

mtp_variant()
Retrieves the variant type of the MTP layer-3 for the signaling point indicated by the (sp).

mtp_pc_bytes()
Retrieves the pointcode size, in bytes, of the MTP layer-3 for the signaling point indicated by the (sp).

mtp_pc_size()
Retrieves the pointcode size type of the MTP layer-3, for the signaling point indicated by the (sp).

mtp_dest_stat()
Retrieves the status of the destination point code — indicated by the (dpc) at MTP Level 3 — in the signaling point indicated by the (spc).

4.2.6.1 MTP Primitives

The seven primitives exchanged between MTP/L3 and user parts are:

- MTP_TRANSFER
- MTP_STATUS
- MTP_PAUSE
- MTP_RESUME
- MTP_INFO
- MTP_RESTART_BEGINS
- MTP_RESTART_ENDS

All user primitive messages travel in a *ipcmsg_t* structure. The following sections explain each primitive.

MTP_TRANSFER

This primitive is used for regular MSU transfer between user parts and MTP/L3. Upon receiving a primitive of MTP_TRANSFER, the user must call the ***mtp_xfer_ind*** API function to decode the MSU. ***mtp_xfer_ind*** populates the *mtpxfer_t* structure. To send an MSU to network, the user must fill the *mtpxfer_t* structure and call the ***mtp_xfer_req*** API function.

Prior to this release, *MTP_Xfer_t* structure was used to send and receive messages, requiring the user to populate both the ‘originating point code’ and ‘destination point code’ fields. Now, only one pointcode definition is required. The pc_xxx members define the ‘destination point code’ for outgoing messages, and ‘originating point code’ for incoming messages. The MTP library populates the ‘originating point code’ field for outgoing messages, and hides the ‘destination point code’ field for incoming messages. Applications interested in populating the 'originating point code' field (i.e., in addition to 'destination point code' information) for outgoing messages can use the *mtp_xfer2_req* API call instead of *mtp_xfer_req*.

MTP_PAUSE, MTP_RESUME, and MTP_STATUS

These primitives are sent from MTP/L3 to user parts to indicate the status of a destination point code. Upon receiving these primitive, users must call the ***mtp_flow_ind*** API function to get the detailed information about the primitive. ***mtp_flow_ind*** populates the *mtpflow_t* structure depending on the primitive type. The “status” and “type” fields of the structure are populated only if the primitive is MTP_STATUS.

MTP_INFO

The MTP_INFO primitive is broadcast to all users when the PROTOCOL, SPC and NI parameters of one signaling point is modified. Upon receiving this primitive, users must call the ***mtp_info_ind*** API function to get the modified data.

MTP_RESTART_BEGINS and MTP_RESTART_ENDS

These primitives are used to indicate the beginning and the end of the MTP restart period. During this period, user parts should not send any message to the network. Similarly, no message is delivered to the user parts.

At the beginning of the restart period, the MTP_RESTART_BEGINS primitive is sent to all user parts. Users must stop sending messages to the network because all incoming and outgoing messages are discarded in MTP/L3 due to the recommendation.

At the end of the restart period, the MTP_RESTART_ENDS primitive is sent to all user parts. Users can resume message sending to the network, if the corresponding destination is accessible, i.e., if the MTP_RESUME indication is received for that destination.

4.3 Signaling Connection Control Part (SCCP)

4.3.1 Introduction

Distributed7 SCCP is the distributed version of the SCCP protocol layer. Here, distributed means that a Distributed7 SCCP Network Service Point is actually formed of a number of machines that host the Distributed7 SCCP software to provide features like load-sharing, management action recovery and connection recovery.

4.3.2 Multiple Instance Support

Distributed7 SCCP supports multiple instantiation of both the SCCP protocol layer and SCCP subsytems. But for a particular host, only one instance of each entity can exist.

Multiple instantiations of SCCP related entities are needed to provide a fault-tolerant SCCP Layer which also supports a load-shared mode of data delivery.

4.3.3 SCCP Message Routing

4.3.3.1 Routing of SCCP Management Messages

Every SCCP SP has a master instance to perform SCCP management actions related to that SP. The address of the master host is recorded in the SP data, and all of the management messages are routed to the master SCCP instance — for that SP. This routing policy is applied to both network received, SSN generated, and to internally generated, i.e., by any SCCP module, management messages.

4.3.3.2 Routing of Non-management SCCP Messages

Routing of User Received Messages

User-received, i.e., subsystem, non-management messages are directly handed over to the MTP Layer. No SCCP based routing is performed on such messages.

Routing of Network Received Messages

The SCCP layer provides different routing policies for its clients. The policy for a particular subsytem is declared during the registration of the first subsystem instance, and that policy is used to route most of the non-management SCCP messages destined towards that subsystem.

- *ROUTE2CLOSEST*: Route messages to the subsystem instance that resides on the same host with the routing SCCP instance. If no SSN instance exists on the same host, then the destination instance is determined by applying a selection algorithm. The algorithm depends on the address of the routing SCCP instance. This policy ensures that a particular SCCP instance delivers to a particular instance of a subsystem.
- *LOADSHARE*: Route messages by applying a round-robin algorithm to existing instances of the destination subsytem.
- *ROUTE2MASTER*: Route messages to a fixed instance of the destination subsytem. This policy ensures that only one instance of a subsytem receives messages from all the SCCP instances for a particular SP.

All non-management messages, with the exception of following three cases, are distributed according to the subsystem declared routing policy:

- *SEQUENCED MESSAGES*: If subsystem routing policy is *loadshar*e, all sequenced messages are delivered according to *route2closest* policy. This ensures that all sequenced messages received from a particular link are received by the same subsytem instance.
- *SEGMENTED MESSAGES*: If subsystem routing policy is *loadshar*e, an index is generated from the reference number of the incoming segmented message, and the destination instance is located utilizing the generated index.
- *CONNECTION ORIENTED MESSAGES*: SCCP connections are expected to be terminated by a single instance of a subsystem. Because of this expectation, *loadsharing* is applied only for Connection Request messages, i.e., received from the network, and the remaining set of messages referring to a particular connection is routed to that SSN instance that requested

the setup of the connection, or was chosen as the destination instance for a Connection Request message received from the network.

Routing of Configuration Messages

Configuration of the MO database is done by individual SCMD instances upon broadcast requests received from the global SCMD. Only the global SCMD instance is allowed to forward the configuration commands to its driver. This is because the DKM infrastructure is used for driver level data synchronization.

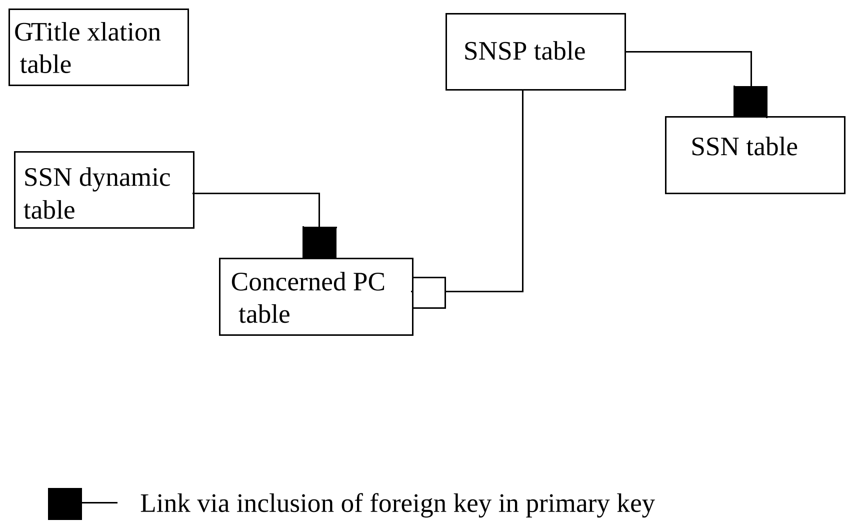
Routing of Local Broadcast Messages

Local broadcast messages for a particular SP are broadcasted to all instances of all subsytems that have registered to that SP.

4.3.4 Data Distribution Methods

4.3.4.1 Data Model

SCCP data for each SP is organized as Distributed Record Access (DRA) tables that form a normalized relational data base. Three additional DRA tables are used to store SP data, SP dynamic data, and private device data. Every SP record provides access to the related DRA tables on an SP basis.



- *SP Table*: SP specific SCCP data, protocol information, pointers to related DRA tables, etc.
- *SP Dynamic Table*: SP related dynamic data
- *PDD Table*: Private device data for the SCCP multiplexer
- *Gtitle Xlation Table*: Data used for incoming and outgoing global title translations
- *SNSP Table*: SCCP network signaling point data
- *SSN Table*: Common subsystem data
- *SSN Dynamic Table*: Local subsystem data
- *Concerned PC Table*: Concerned point code data
- *Connection Table*: Connection data for a local subsystem

In all of the tables above, record private parts are used to store data that is relevant for a particular host, i.e, action timers, buffer pointers, local round-robin marks, etc., whereas record distributed parts are used to store data that is meaningful at the SP level, i.e, global title information, subystem routing and recovery policies, connection references, etc.

4.3.5 Software Recovery Procedures

SCCP layer crash recovery utilizes the Internal Event Notification mechanism and state information kept for each recoverable entity (SNSP, SSN and Connection).

SCCP recovery means:

- Re-distributing the ownership of entities owned by the terminating instance
- Continuing with or gracefully terminating the actions that were started on these entities by the terminating instance

4.3.5.1 SP Level Recovery

When an SCCP multiplexer terminates, SCCP multiplexers on other hosts check whether the terminating multiplexer served as a management instance for any of the available SCCP SPs. For all the SPs that are hosted by the terminating instance, the corresponding host list is updated to remove the terminating multiplexer's address from the list.

For every SP that was managed by the terminating multiplexer, a new management multiplexer is selected, and SP level recovery is initiated.

The first step of an SP recovery is the finalization of MTP requested jobs, i.e., MTP Pause, MTP Resume, and MTP Status. The management SCCP instance performs an MTP audit to get the status of all SNSPs defined for the SP. If the MTP state is not in line with the SNSP state as it is recorded in the SCCP database, then the related state machine, i.e., SP Allowed Control, SP Prohibited Control, or SP Congested Control, is triggered.

The next step is the recovery of SSN state information, i.e., the finalization of SSN management actions. For all the subsytems of allowed SNSPs — decided after the first recovery step — a Subystem Test procedure is initiated.

4.3.5.2 SSN Level Recovery

When an SCCP subsystem (instance) terminates,

1. The SCCP multiplexer hosting the master instance for the subsytem performs connection recovery for the subsystem. If the terminating instance is the master itself, then a new master is selected.
2. The master SCCP multiplexer updates the corresponding SSN host list to remove the terminating instance's address from the list.
3. All connections handled by the terminating instance are recovered. The master instance performs the connection recovery depending on the three connection recovery policies available to the subsystem:
   - *RESUME* – Master instance assumes ownership of the connection
   - *ABORT* – Connection release procedure is initiated
   - *CLEAR* – Connection resources are released without further action

4.3.6 Protocol Specific Issues

4.3.6.1 Protocol Variants Supported

Distributed7 supports the following variants:

- ATT (Variant of ANSI)
- 5ESS (Variant of ANSI and ITU)
- INTERN (Variant of ANSI and ITU)

Variant is a parameter of the SCCP managed object, and the default value is set to NONE. To change the current variant of the SCCP layer, Variant parameter of SCCP MO must be modified. For mor information, see MODIFY-SCCP command in Section 9.5.5 on page 9-101 in this manual.

4.3.6.2 Single Object Code Support

SCCP configures itself at run-time to support ANSI92, ANSI96, ITU93, ITU97, or ETSI 97 stacks. Configuration information is retrieved from the MTP layer. The decision of the protocol type for the MTP layer also dictates the protocol type used for the SCCP layer.

4.3.6.3 Global Title Related Changes

SCCP supports wildcard global titles. A wildcard global title is indicated by setting the *wildcard* parameter of the global title to *true*. If no exact match is found for a global title, then the longest matching wildcarded title is used for the translation, if one can be found.

SCCP also takes the *numbering-plan* as a part of the global title. The value of the numbering plan depends on the global title indicator.

4.3.6.4 Handling of MTP Primitives

SCCP handles MTP primitives, i.e., pause, resume, or status, on MTP networks and clusters. All SNSPs accessed through an MTP network or cluster are managed according to the state of the network (cluster).

4.3.7 Dependencies on Other Distributed7 Components

SCCP data is built upon the DRA (hence DKM) framework, so apart from its dependency to SPM and UPM multiplexers, DKM multiplexer and DRAMOD module are needed for proper functioning of the SCCP layer.

4.3.8 Application Programming Interface

The subsystems registered through the old style APIs that use ***spm_open()*** and ***spm_bind()*** instead of ***sccp_reg()*** can only use the ROUTE2CLOSEST option for message routing, and the ABORT option for connection recovery.

4.4 Transaction Capabilities Application Part (TCAP)

This section describes the distributed operations of the Transaction Capabilities Application Part (TCAP) protocol under this Distributed7 release. This TCAP implementation is fully backward compatible with earlier releases of the product, and features a multitude of distributed related capabilities, e.g., message exchange over remote signaling links, support for front-end/back-end configurations, automatic failure detection and transaction-layer software recovery, stand-alone vs. distributed mode of operations, and additional control over TCAP load-sharing algorithms.

4.4.1 Multiple Instance Support

4.4.1.1 Need for Multiple Instances

Distributed7 supports multiple instantiations of a particular TCAP application on a single host or on multiple hosts. Multiple instantiations of TC applications offers two advantages:

- Implements fault-tolerance configurations in which premature termination of a particular instance of a TC application can be recovered by the system
- Increases the throughput of a TC application by allowing multiple instances of the application to participate in concurrent processing of messages received from the remote end

4.4.1.2 Concurrence Support and Restrictions

Distributed7 supports the following TCAP capacities:

- Up to 64 TC applications can run concurrently on each host machine comprising a distributed environment
- A distributed environment can have up to eight (8) hosts
- The maximum number of TC applications that can co-exist in a network is 512
- Up to 127 instances of a given TC application can execute concurrently on each host
- The maximum number of instances for a particular TC application that can co-exist under a Distributed7 environment is 127 * N, where N is the number of hosts configured

Note: *These limits apply to both TCAP over SCCP, and TCAP over TCP/IP applications.*

4.4.2 Routing of Outgoing Messages

4.4.2.1 Configurations Supported

Distributed Environment

When exercising the TCAP over SCCP functionality, the SS7 signaling links used for exchanging messages with the SS7 network do not need to be located on the same host on which the TC application is running. This is to say that in Distributed7, it is possible to run TC applications on a host that does not have direct signaling link connections to the SS7 network. This flexibility is a feature of the MTP layer, and is normally transparent to the users of the MTP layer 3 — in this case the SCCP layer. Communication between hosts in a distributed environment is through TCP/IP. The trade off for this flexibility is performance. See Section 4.4.6 for more information.

Front End/Back End Support

This release supports front-end/back-end system configuration in which the MTP and SCCP layers run on one or more front-end hosts, and the TCAP layer — as well as the TC application — run on one or more back-end hosts. Communication between the TCAP and SCCP layers in a front-end/back-end configuration is through a LAN interface.

For TCAP over TCP/IP, this release requires that each host that features TC applications also has local TCP/IP connections to all appropriate remote ends, i.e., the remote hosts that are of interest.

4.4.2.2 Message Routing Algorithms Used

Distributed Environment

In the case of TCAP over SCCP, when a TC application originates a TCAP message, this message is submitted to the SCCP layer on the local host for subsequent delivery to the MTP layer 3 again on the local host. From this point on, whether the message is sent to the SS7 network through signaling links on the local host or through links located on remote hosts depends on how MTP L3 has been configured, i.e., how the link sets have been formed. If the signaling link selected by the MTP L3 is a local one, then the message is sent to the SS7 network through that link on the local host. This is in effect no different from a stand-alone product configuration. If the signaling link selected by MTP L3 is located on a remote host, however, then the message is delivered to the remote host over the kernel-resident TCP/IP connection between the two hosts, and it is subsequently sent to the SS7 network from the signaling link located on that remote host.

Front End/Back End Support

In a front-end/back-end configuration in which the TC application is run on a back-end host and the SCCP and MTP layers are run on front-end hosts, messages originated by the application are first transported across the TCP/IP connections between the front-end and back-end hosts, and only then sent to the SCCP and MTP layers on the front-end host. The effect is the same as that of the application running on the front-end host.

When the TCAP over TCP/IP is in use, messages originated by a TC application are sent to the TCP/IP network directly through the local host.

Note: *All such messages first go through the TCMOD STREAMS module linked under the SPM multiplexer on the local host before they are sent to the TCP/IP network.*

4.4.3 Routing/Distribution of Incoming Messages

4.4.3.1 Configurations Supported

Due to the distributed nature of the MTP layer, when TCAP over SCCP is in use, incoming messages destined for a TC application can be received over SS7 signaling links located on the local host as well as on remote hosts. When operating under a front-end/back-end configuration, SS7 messages intended for a TC application are always received through the signaling links that are connected to the front-end machines. Routing of these messages to the TC application involves transporting them across the LAN interface that connects the front-end and back-end machines. For TCAP over TCP/IP however, incoming messages are always expected to be received across the TCP/IP connections that are in place between the local host and the corresponding remote host.

4.4.3.2 Message Routing/Distribution Algorithms Used

TCAP messages received from remote entities, i.e., through the SS7 network or TCP/IP connections to remote hosts, can be categorized as follows:

- Incoming dialogue requests, including Query With — or Without — Permission messages, Unidirectional messages, and Begin Conversation messages. In this case, there are likely multiple candidates to which the message can be delivered. These candidates may or may not be located on the same host as the one on which the message was received, especially when TCAP over SCCP is in use.
- Responses to on-going dialogues, including all other transactions. In this case, the message is associated with a particular instance of a TC application. It therefore needs to be delivered to that instance as long as that instance is up and running.

TCAP implementation relies on the 4-bit, logical host ID information. This information is embedded in all transaction IDs generated by the Distributed7 TCAP layer software, and identifies the actual destination for an incoming transaction that is associated with a particular TC application/instance. All other (unassigned) incoming transactions are loadshared among qualified instances of the target TC application.

Distribution of SS7 Layers and Location of Signaling Links

As described in Section 4.4.2, when TCAP over SCCP is in use and the actual TC application is running on a remote host, TCAP messages can always be received over signaling links. Such messages need to be delivered to the TCAP layer (on any host within the network) first before it is determined where exactly the target TC application is running.

- In the best case, i.e., the SS7 signaling links are located on the same host as the one on which the TC application is running, no intermediary nodes are involved, and the message is delivered to its destination directly through the MTP and SCCP layers.
- In the worst case, up to four intermediary nodes may be involved before the message is delivered to its final destination. The existence of each intermediary node introduces an additional delay in message delivery. This scenario should therefore be avoided by carefully designing which SS7 layers run on which hosts within a network, and where the SS7 signaling links are located.

In the current implementation of TCAP over TCP/IP, it is assumed that all TCAP messages received across a TCP/IP connection (to a remote host) are destined for the TC applications running on the local host. Therefore, message distribution involves which particular instance of a TC application should be receiving a TCAP message that is retrieved from the TCP/IP network. This situation is no different from a stand-alone product configuration, and the message distribution logic employed by the TCAP transaction-layer is, therefore, identical to that of stand-alone product configuration.

Load-Sharing Algorithms Used

All incoming TCAP messages that are not part of ongoing dialogues are loadshared between qualified instances of the target TC application. The load-sharing mechanism that was available in former releases of the Distributed7 product required all such messages to be delivered to the *least-busy* instance of the target TC application that expressed an *a priori* interest in loadsharing. In this release, an additional layer of control is introduced to the TCAP loadsharing mechanism. This new layer of control involves loadsharing all incoming (unassigned) messages between qualified hosts in a *round-robin* fashion first. It is only after the target host is identified that an incoming message can be delivered to the *least-busy* instance running on that host.

In a front-end/back-end system configuration, a third level of control exists on loadsharing of incoming TCAP message traffic at the front-end machines. This loadsharing is based on a distinction between *primary* and *secondary* types of connection modes by the TC application on the front-end hosts.

4.4.4 Data Distribution Methods

4.4.4.1 Centralized vs. Replicated Data

For high-availability and improved performance reasons, this TCAP implementation relies heavily on the use of replicated, as opposed to centralized, data whenever possible.

4.4.4.2 Component Layer Data

The user-space data maintained by the TCAP layer involves component-handling aspects of the TCAP protocol, and is not shared/distributed between individual instances of a TC application: The TCAP API library allocates and maintains user-space data for each TC application in an isolated fashion, and this data is neither shared nor distributed when operating in the distributed mode.

4.4.4.3 Transaction Layer Data

The kernel-space data maintained by the TCAP layer mostly involves transaction-handling aspects of the TCAP protocol, and is shared/distributed between multiple hosts using the DKM framework. The TCAP transaction layer stores all critical pieces of information about active transactions in a kernel-resident transaction table which, in essence, is a DKM segment. Thus, while the TCAP transaction-layer software, i.e., the TCAP multiplexer, on each host is primarily responsible for handling transactions and updating the corresponding pieces of data in the local transaction table, the DKM framework is responsible for maintaining replicated copies of this data on all remote hosts involved, i.e., hosts on which instances of the specified TC application is running, and keeping them synchronized at all appropriate times.

4.4.5 Software Recovery Procedures

4.4.5.1 Scope of Recovery

At the TCAP protocol layer, the only recoverable item is the on-going transaction. This TCAP implementation does not provide any built-in features to recover component-layer data just in case a TC application that is in the process of assembling or disassembling components were to terminate its execution prematurely. Therefore, if any component-layer software recovery is desired, then it must be provided by the TC application itself.

4.4.5.2 Failure Detection Mechanisms

Failure detection under a Distributed7 environment is as follows:

- When the endpoint associated with a particular instance of a TC user is closed, the system searches through the transaction table associated with that instance and initiates an appropriate set of recovery procedures for all unfinished transactions owned by that instance.
- When the TCAP multiplexer on the local host detects that the system software on a remote host is in the process of being shut down, it searches through the replicated copies of all transaction tables associated with TC applications running on that host and initiates an appropriate set of recovery procedures for all unfinished transactions owned by each application.
- When the TCAP multiplexer detects that a remote instance of it is terminated — either prematurely or as a result of a remote system software shutdown — it searchs through the replicated copy of the transaction table associated with that TC application and initiates an appropriate set of recovery procedures for all unfinished transactions owned by that application.

4.4.5.3 Recovery Methods Available

This implementation of TCAP protocol features a variety of transaction-layer recovery policies as follows:

Transaction Purge Policy

When a TC user terminates, the transaction records owned by that user (if any) are purged by the Distributed7 system software automatically, and the associated dialogue identifiers are returned to the pool for use by other instances of the same application.

Transaction Abort Policy

When a TC user terminates, all active transactions, whether incoming or outgoing, associated with the user (if any) are aborted. The Distributed7 system software automatically sends abort indicator messages to the remote end with an appropriate abort cause. The system purges any transactions in an initiated state (such as Init-Sent and Init-Rcvd state in ITU protocol) owned by the TC user at the time of termination.

Transaction Adopt Policy

When a TC user terminates, all active transactions owned by the user (if any) are assigned/re-distributed among surviving instances of that TC user so that transaction processing may continue. If no such instance is around, the transaction records owned by the terminating TC user are purged by the system.

The aforementioned recovery policies are available to TC applications running in the stand-alone mode as well as in the distributed mode. In the distributed mode, the adoption procedure can result in transactions owned by a particular TC user to be adopted by remote instances of that TC user running across the network.

Application control over these recovery policies is through a new set of function calls that are available as part of the TCAP API library.

***Important**: The TCAP implementation features no built-in software recovery intelligence at the TCAP component-handling layer. If such functionality is desired, then it needs to be implemented by the TC application itself. TC applications may take advantage of the powerful user/kernel-space data distribution frameworks that are available as part of Distributed7 to implement customized recovery strategies.*

4.4.6 Performance Considerations

4.4.6.1 Data Synchronization Methods Available

When operating under a Distributed7 environment, the transaction table contents on all hosts on which instances of a TC application are running must be synchronized for fault recovery reasons. This applies to all configurations except stand-alone or where the transaction purge policy is in use.

In essence, data synchronization takes place as soon as a transaction record on a host is updated. Under normal circumstances, a TC application waits for the data synchronization procedure to complete before processing its next transaction, i.e., the `DKM_SYNCFIRST` flag is used when releasing a DKM lock that spans a transaction record.

It is also possible that the `DKM_SYNCLATER` flag can be used during an unlock operation, in effect performing the actual data synchronization off-line. This approach is intended to boost the performance of a TC application while running under a distributed environment. However, it is likely to result in a less reliable configuration, i.e., if a fatal error occurs while the TC application on a specified host is handling transactions, then transaction records on the surviving hosts may not be up-to-date, and transaction recovery may therefore prove impossible.

4.4.6.2 Response Times

The message routing mechanisms described in Section 4.4.2.2 and Section 4.4.3.2 introduce additional flexibility in network configuration. However, routing inter-host messages through intermediary nodes results in additional delays — approximately 1 msec per message. Therefore, if performance is an issue, then each host on which there is a TC application using the SCCP transport services should also have local connectivity to the SS7 network.

4.4.7 Protocol Specific Issues

4.4.7.1 Protocol Variants Supported

Distributed7 TCAP supports the following variants:

- ANSI 92
- ANSI 88
- ANSI 96
- ITU White Book
- ITU Blue Book
- ITU AT&T variants
- ITU 97

4.4.7.2 Transport Service Providers Supported

Distributed7 TCAP supports the following transport service providers:

- Signaling Connection Control Part (SCCP)
- Transmission Control Protocol / Internet Protocol (TCP/IP)

4.4.7.3 Dependencies on Other Distributed7 Components

In Distributed7, the existence of the TCAP layer on a particular host requires the local existence of the associated service provider, i.e., SCCP or TCP/IP. This is to say, on each host where there is a TC user associated with a particular **SP/SSN** pair (when SCCP transport services are in use), the SCCP multiplexer for that **SP** needs to be instantiated. Similarly, for each Distributed7 host on which there is a TC user associated with a particular **HOST/PORT** pair (when TCP/IP transport services are in use), a kernel-level TCP/IP connection to that **HOST** needs to be set up, and the **TCMOD** module needs to be pushed across this connection.

4.4.7.4 Stand-alone vs. Distributed Mode of Operations

Distributed7 allows TC applications to control their mode of operation through a command-line utility. The operation mode of a TC application in a distributed environment is automatically adjusted by the TCAP layer, depending on whether the same application is running on multiple hosts or not. If the same application is running on multiple hosts, then its operation mode is set to *distributed*; otherwise, its mode is set to *stand-alone*.

The performance of a TC application that is running in distributed mode may be worse than that of the same TC application running in stand-alone mode, depending on the setting of the transaction recovery policy of that application. If the purge policy is in effect, for example, there are no differences in performance between distributed and stand-alone configuration. This is because no data synchronization overhead is involved when the purge policy is in use.

4.4.7.5 Dialogue ID Allocation

When operating under a Distributed7 environment, TC applications on each host acquire their dialogue IDs in a localized manner. To do this, the TCAP multiplexer on the local host uses the services provided by the TCAP API library. The maximum number of dialogue IDs that can be acquired by a particular TC application on a specified host is limited to 65536. Multiple instances of a TC application on a specified host share the dialogue ID pool. However, when multiple instances of a TC application run on different hosts, their dialogue ID pools are completely segregated.

4.4.7.6 Construction of Transaction Identifiers

In the TCAP layer software, transaction identifiers are constructed as 32-bit data types comprising a 16-bit dialogue identifier, a 4-bit logical host identifier, and an ever increasing number. That is to say, each TC application on a specified host can carry out up to a maximum of 65536 dialogues concurrently, and the chances of associating the same transaction identifier with a particular dialogue identifier are slim. This reduces the possibility of misinterpreting remote replies that are already late. The logical host identifier information within transaction IDs identify the host that a particular transaction is associated with, and is therefore instrumental in routing incoming TCAP messages to their final destination. See Section 4.4.3 for more information.

4.4.8 Application Programming Interface

4.4.8.1 Changes to Existing API Libraries

In Distributed7, compile-time versions of the TCAP API library that existed in former releases of the product are completely absorbed within the new TCAP library. They are not visible to the users of the new library unless one of the `-DANSI` or `-DCCITT` compile-time flags is explicitly specified. Only when one of these flags is specified can users access the earlier versions of the TCAP API library calls. Otherwise, only the current release versions of the API library calls are available.

In this release, all functions contained within the TCAP parameter extensions API library are renamed for consistency reasons. The names of all functions listed in the TCAP extensions API library now start with the **`tcx_`** identifier. On-line manual pages are available for every function contained in this library.

4.4.8.2 Backward Compatibility Issues

This release of the TCAP API library is largely backward compatible with the earlier releases of TCAP libraries, with the exception of the ***tcm_open()***, ***tcm_getdlgp()***, and ***tcm_putdlgp()*** calls (see Summary of Changes to API Calls below). The TCAP extensions API library is also largely compatible with its former releases. For a list of functions that are not supported by this release of the TCAP Extensions API library, please refer to the API backward compatibility charts provided in the *Distributed7 Application Development Manual*.

Summary of Changes to API Calls

- Changes in ***tcm_open():***
  Necessary to allow TCAP users to specify the number of dialogue identifiers needed, i.e., up to a maximm of 16384.
- Changes to ***tcm_getdlgp()*** and ***tcm_putdlgp()***:
  Necessary to support ANSI 96 protocol, which is the only ANSI variant that allows the dialogue protion to be present in TCAP messages exchanged.

4.4.8.3 Transaction Recovery APIs

To benefit from the transaction recovery features of the TCAP layer, a different set of functions have been added in this release for sending/receiving messages as well as retrieving and storing component and dialogue information. These functions use the full transaction identifier rather than the local dialogue identifier (which is a portion of the transaction identifier) and mostly have the suffix _n attached to the end of their names. The portions of the TCAP library which are not used for message, dialogue and component send/receive purposes can be used together with TCAP Transaction Recovery Functions (tcm_getopa, tcm_notify,m etc.).

4.4.9 JAIN TCAP API Support

Starting with Distributed7 1.3.0 release, JAIN (Java APIs for the Integrated Network) TCAP API is supported. JAIN TCAP API is implemented based on JAIN APIs for the Integrated Network, JAIN Transaction and Capabilities Part (TCAP) Specification Version 1.1. JAIN TCAP API supports the following TCAP specifications:

- ITU Q.771-Q.775, 1993, 1997
- ANSI T1.114.x, 1992, 1996

Some of the startup operations which are needed to be performed by the JAIN TCAP Listener application are listed in the following table. For the rest of the operations and the full JAIN TCAP API specification, please visit http://www.jcp.org/aboutJava/communityprocess/first/jsr017.

GetInstance	Returns an instance of a JainSS7Factory. This is a singleton type class so this method is the global access point for the JainSS7Factory.
setPathName	Sets the Pathname that identifies the location of a particular Vendor's implementation of the JAIN SS7 Objects. (e.g) com.newnet
createSS7Object	Returns an instance of a Peer JAIN SS7 Object identified by the supplied classname. This supplied classname is the lower-level package structure and classname of the required Peer JAIN SS7 Object.
setVendorName	Sets the vendor name for this stack. This name will be the Vendor's domain name inverted. e.g. com.newnet
setStackName	Sets the name of the stack as a string. e.g. "test"
setSignalingPointCode	Sets the Signaling point code of this stack.
setStackSpecification	Sets the stack specification. Possible values are: TcapConstants.STACK_SPECIFICATION_ANSI_92 TcapConstants.STACK_SPECIFICATION_ANSI_96 TcapConstants.STACK_SPECIFICATION_ITU_93 TcapConstants.STACK_SPECIFICATION_ITU_97
createProvider	Creates a vendor specific implementation (Peer) of JainTcapProvider and returns a reference to it. This newly created provider is then attached to the JainTcapStackImpl object in order to send and receive messages between the Provider and the stack.
addJainTcapListener	Adds a JainTcapListener to the list of registered Event Listeners being serviced by this JainTcapProvider object. This is also used to add new User Addresses being handled by a TCAP Listener so that a Listener can register for more than one User Address by repeatedly calling this method.

Distributed operation of JAIN TCAP Listener application is not supported with this release of Distributed7.

4.5 ISDN User Part (ISUP)

This section describes the distributed operations of the ISUP protocol layer of this release of Distributed7.

CCA OVERVIEW ISUP, or ***isupd*** in short, implements the ISDN-UP (ISUP) layer of Signaling System Number 7 and runs in two different modes, the *stand-alone* and the *distributed*. The ***isupd*** daemon process works in stand-alone operation mode when it is run with **-s** command line argument.

***Note: isupd** can be run in stand-alone configuration without using **-s** option, but the **-s** option makes **isupd** run faster and results in better performance because it does not perform time consuming data synchronization over distributed shared memory (DSM) framework.*

In this release of Distributed7, the default mode of ***isupd*** is *distributed*. In the distributed mode, ***isupd*** distributes and synchronizes its data over DSM framework and implements some loadsharing, message distribution and routing and, should a failure occur at run time, software recovery methods. The following sections describe these methods in more detail.

4.5.1 Multiple Instance Support

To achieve fault-tolerance and increase overall performance, multiple instances of ***isupd*** are allowed to co-exist on multiple hosts within a distributed environment. However, on a specified host, only one ***isupd*** instance per signaling point can exist. All ***isupd*** instances for a signaling point work in *"active/active"* mode. In other words, all the ***isupd*** instances can be used to process ISUP messages at any time if an ISUP message is delivered to them.

Call Control and maintenance system applications are the users of the ISUP protocol layer. A single process can achieve both Call Control and maintenance system activities. multiple instances of Call Control and maintenance system applications are allowed to co-exist on the same host and/or multiple hosts. However, ***isupd*** instances are able to work only on a machine that is equipped with the MTP/L3. The maximum number of ***isupd*** instances that can co-exist under a distributed CCA OVERVIEW environment are, therefore, restricted by the maximum number of hosts that are configured as part of a distributed environment.

4.5.2 Trunk Allocation And Load Sharing

In this release of CCA OVERVIEW, the implementation of ISUP allows defining a Call Control and maintenance system per trunk group so that it can loadshare among different Call Control or maintenance systems. Call Control and Maintenance system applications choose ownership of the trunk groups, i.e., CIC groups, and register only those groups. Each ISUP instance is responsible for processing the messages whose CICs belong to trunks that are registered totheir Call Control. More than one Call Control or maintenance system application cannot register to the same trunk. If an ISUP instance is not equipped with Call Control or maintenance system applications, the the instance waits in idle mode.

Registration per trunk group does not mean that the ISUP users cannot process the messages for non-registered trunks. On the contrary, they are capable of processing messages associated with different trunks.

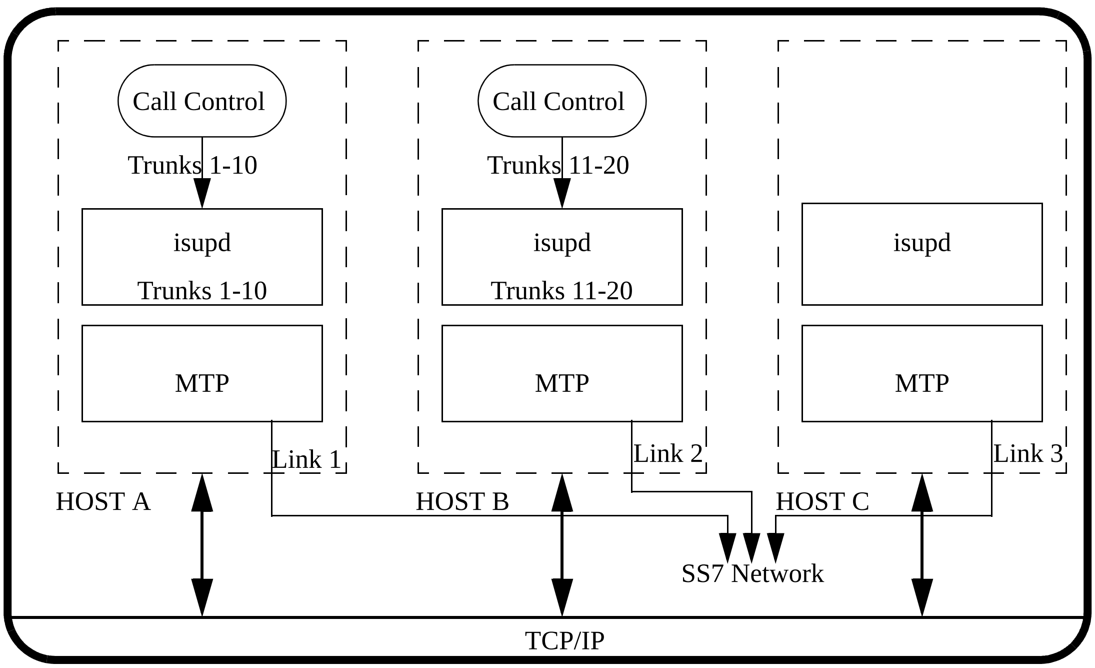


Figure 4-3: ISUP Trunk Allocation

In Figure 4-3, local Call Controls register for trunks 1 through 1Ø on host A, and 11 through 2Ø on host B. Therefore, the ISUP instance on host A is responsible for processing messages for trunks 1 through 1Ø, and the instance on host B is responsible for processing messages for trunks 11 through 2Ø. Since there is no registration to ISUP on host C, ISUP on this host waits in idle mode.

4.5.3 Message Distribution and Routing

ISUP instances are responsible for processing messages that have specific trunk groups. This approach is followed by each ISUP instance. ISUP instances do not request load sharing from MTP L3. Maintenance and Call Control applications deliver their messages to the ***isupd*** instance, which may run on a different host. To able to achieve this feature, they must bind their IPC addresses across the network. Each ISUP instance uses the same message distribution/routing algorithm, and decides whether the received message must be processed by itself or not. If the received message is not processed by an ISUP instance, the message is forwarded to an appropriate ISUP instance. Therefore, each ISUP instance knows of the existence of all other instances, and amongst the ISUP instances a unanimous decision is reached regarding which instance must process a specific message.

In this release, the two parts of an ISUP instance are:

- ***isupd* –** The ISUP daemon process in user-space
- ***isupmod* –** The ISUP module () in kernel-space. The main function of this module is to route messages to the correct ISUP daemon for processing without going into the user-space. This avoids the performance penalties associated with crossing into the user-space. ***isupmod*** is pushed by the ***isupd*** daemon process on top of the UPM multiplexer. If and when an ***isupd*** receives a message, this message is processed by this daemon and sent to its ultimate destination.

In Figure 4-3, the ISUP messages received from the SS7 network on Link 3 are delivered to the ISUP instance on host 3. Since there is no registered Call Control on this machine, the message is forwarded to Host A, if the trunk group is between 1 and 1Ø, or to host B, if the trunk group is between 11 and 2Ø. The ISUP messages received from the SS7 network on Link 2 are delivered to the ISUP instance on host B. Similarly, if the trunk group is between 1 and 1Ø, it is forwarded to Host A, and if the trunk group is between 11 and 2Ø, it is processed by the ISUP instance on host B and, if necessary, the message is delivered to the Call Control running on host B.

Call Control applications deliver their messages to the local ISUP daemon at all times. If a Call Control runs on a specific host, then there has to be a local ISUP daemon on that host.

Figure 4-4 illustrates the flow of messages received from MTP L3 by an ISUP instance. If it is decided that the received message should be sent to another ISUP instance, then destination address of the message is updated accordingly, and the message is sent back to MTP L3. Otherwise it is processed and sent to the Call Control application.

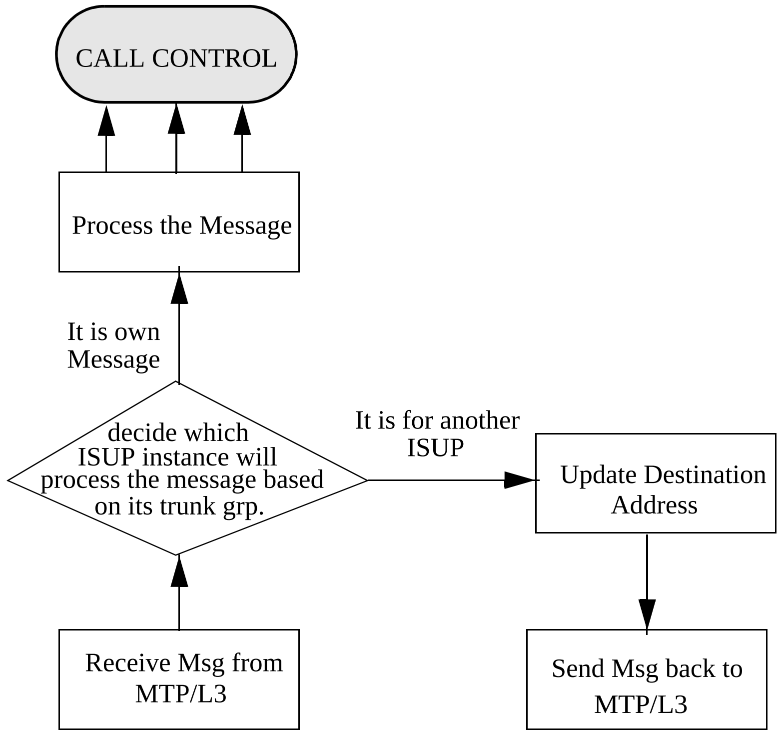


Figure 4-4: Flow of ISUP Messages Received From MTP/L3

4.5.4 ISUP Protocol Data Distribution Methods

ISUP has the following major pieces of data that are accessed and/or modified by all of its instances executing under a distributed CCA OVERVIEW environment:

- ISUP configuration data
- ISUP node data
- ISUP circuit group data
- ISUP circuit data
- ISUP timer data

Among these data, only node, circuit group, and circuit data change dynamically. These three pieces of data are replicated on each host using the DSM framework because they need to be kept synchronized across the individual hosts. Configuration and timer data do not change dynamically. They can be accessed or manipulated with standard management interfaces such as *MML* and *AccessMOB*.

In this release, both ISUP instances, i.e., ***isupd***, and ISUP module, i.e., ***isupmod***, maintain kernel-space data. ISUP module has a routing table that is replicated and synchronized on each host that is equipped to route ISUP messages to ISUP instances, i.e., within the ISUP layer (See Section 4.5.3 on page 4-34). The routing table is in the form of a distributed kernel memory (DKM) segment.

4.5.5 Software Recovery Mechanisms

To achieve fault-tolerance and high availability — as well as reliability in case of failures — ISUP instances implement some software recovery methods. If an ISUP instance dies or a host equipped with ISUP goes out of service, then the current work load, i.e., current calls or circuit supervision events, of this instance is shared and recovered by other ISUP instances using the *resume-call* or the *release-call* policy modes.

- *resume-call* policy – only the stable calls are preserved. This is the default software recovery policy in the ISUP layer.
- *release-call* policy – the current active and stable calls are released automatically after recovery.

To implement recovery mode, a new parameter called *RECMODE* of the ISUP Managed Object (MO) needs to be configured. The *RECMODE* parameter may assume *RELCALL* or *RESCALL* set values.

When a Call Control that is registered to at least one trunk group fails, or when an ***isupd*** fails, i.e., it dies or its host crashes, other active ***isupd*** instances receive this failure event. A common decision leads to ***isupd*** instances sharing the trunk groups and load of that failed ***isupd***, updating internal routing tables, and sending an *ISUP_ACCEPT_TRUNK* message to the ISUP users, i.e., Call Control and Maintenance System, running over each of them. The *ISUP_ACCEPT_TRUNK* message contains trunk groups that are processed after the load distribution mechanism from that point on. When an ISUP user receives an *ISUP_ACCEPT_TRUNK* message, all network messages that belong to these trunks are routed to the same ISUP user. ***isupd*** then expects all subsequent messages from the failed trunks of the Call Control to be generated by the same ISUP user.

- If the recovery policy is *resume-call*, then ***isupd*** preserves all active and stable calls, and takes over ongoing events, i.e. start timers, etc.
- If the recovery policy is *release-call*, then ***isupd*** releases all the active and stable calls first, and takes over ongoing circuit supervision events. For each released call, it sends an *ISUP_RELEASE_AFTER_RECOVERY* indication to Call Control with proper CIC and *RELEASE (REL)* ISUP message. Depending on the implementation, the ISUP users must take over ongoing events (timers etc.) when they receive an *ISUP_ACCEPT_TRUNK* indication on received trunk groups.

In Figure 4-5, Call Control on host B is registered to trunk groups 11 through 2Ø. When a failure occurs on this machine, i.e., host crash, Call Control failure, or ***isupd*** failure, other ***isupd*** instances are automatically notified about this failure and a software recovery procedure starts. After load distribution, the Call Control on host A receives an *ISUP_ACCEPT_TRUNK* indication for trunk groups 11-15, and the Call Control on host C receives the same indication for trunk groups 16-2Ø. If the recovery policy was configured as *release-call*, then all the active calls are released on trunk groups 11-2Ø automatically by sending a *REL* message to the SS7 network. For each call released on trunk groups 11-15, Call Control on host receives an *ISUP_RELEASE_AFTER_RECOVERY* indication. The same indication for trunk groups 16-2Ø are sent to the Call Control on host C. If the recovery policy is *resume-call*, no extra indication is given to the Call Controls. It is expected that the Call Controls share their private data and can understand/find if there are any active or stable calls on indicated trunk groups.

The trunk groups 11 through 2Ø are shared equally among active ***isupd*** instances. Here, *active instances* mean ***isupd*** daemons that have a Call Control that registered itself to ***isupd***).

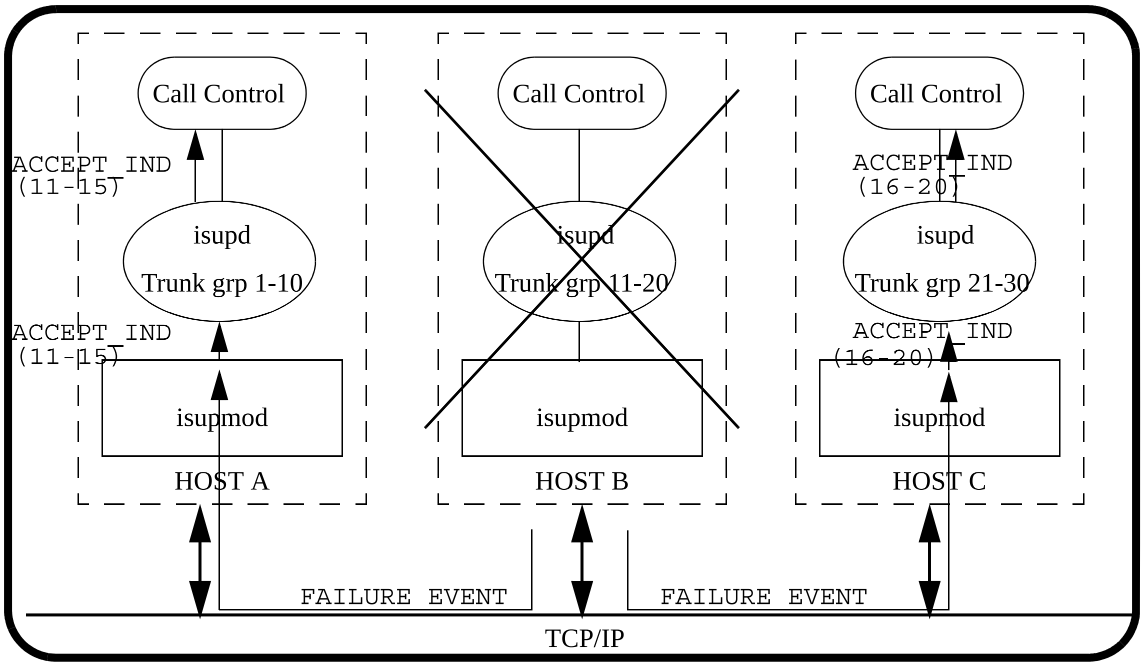


Figure 4-5: Software Recovery When an ISUP Instance Fails

4.5.6 Protocol Specific Issues

This release of CCA OVERVIEW ISUP supports all ANSI and ITU variants in single object code like other layers of the CCA OVERVIEW SS7 layers namely MTP/L2, MTP/L3, SCCP and TCAP. The supported ANSI and ITU variants are as follows:

- ANSI (GENERIC, Recommendation .113 1992)
- BELGIUM (ITU)
- BELL (variant of ANSI)
- CHILE (variant of Q.767)
- CHI24 (ITU)
- CZECH ISUP (ITU)
- DSC (variant of ANSI)
- ETSI97 (ITU97)
- FINLAND (ITU)
- FRANCE (ITU)
- GERMANY (ITU)
- HONG KONG (variant of ITU)
- ITALY (Q.767
- ITU92 (ITU White Book Rec. Q.761-Q764)
- ITU97 (ITU White Book Rec. Q.761-Q764, 09/97)
- MEXICO (Q.767)
- MCI (variant of ANSI)
- NEW ZEALAND (variant of Q.767
- NORWAY (variant of Q.767)
- PHILIPPINES (ITU)
- Q.767 (ITU Blue Book Rec. Q.767)
- RUSSIA (Q.767)
- SINGAPORE (ITU)
- SPAIN (variant of ITU)
- SWEDEN (variant of ITU)
- SWEDENVI (Q.767)
- SWITZERLAND (ITU)
- THAILAND (variant of ITU)
- TURKEY (variant of Q.767)
- UAE (variant of ITU)
- UNIPAC (ITU)

JAPAN and JAPAN_CTM variants are not currently supported. They will be supported in future releases of Distributed7. The default variant of ISUP layer is ITU. VARIANT is a parameter of the ISUP Managed Object. To change the current variant of the ISUP layer, the value of the VARIANT parameter of ISUP MO must be modified. For more information, see DISPLAY-ISUP and MODIFY-ISUP MML commands in *Chapter 9: Man-Machine Language Commands* of this manual.

Important: *CCA OVERVIEW ISUP's VARIANT must be compatible with its lower layers, namely MTP L2 and MTP L3. Before configuring ISUP variant, make sure that it is not inconsistent with MTP layers.*

4.5.7 Application Programming Interface

The following API functions were added to the ISUP Call Control library, ***libisup.a***. For more information about these functions, see the ISUP API Library Reference in the CCA OVERVIEW API Reference Manual.

isup_reg_req()

This function sends an *ISUP_ACTIVATE_REQ* primitive to local the ***isupd*** for the trunks that the ISUP user wants to register. The user type – Call Control or Maintenance System – and registration type – hard or soft – are specified externally as separate arguments to this function.

- If hard registration is requested, then ***isup_reg_req()*** does not fail in such condition. ***isupd*** sends an *ISUP_RELEASE_TRUNK* indication to the remote ***isupd*** to allow it to stop all ongoing events, and then starts these events on itself. The remote ***isupd*** passes an *ISUP_RELEASE_TRUNK* indication to its users to let it know the indicated trunk is no longer available to them for processing.
- If soft registration is requested, then ***isup_reg_req()*** fails if the requested trunks are already activated by other ISUP users running on other hosts.

isup_reg_rsp()

This function must be called whenever an *ISUP_ACTIVATE_CONF* primitive is received from the ***isupd*** daemon. This primitive is sent by ***isupd*** whenever an *ISUP_ACTIVATE_REQ* is received by itself from its user. It returns the trunk group number that activation requested to the caller.

isup_dereg_req()

This function builds a message with the *ISUP_DEACTIVATE_REQ* primitive with its arguments to inform ***isupd*** daemon that the ISUP user has gone out of service and that message processing must stop.

isup_dereg_rsp()

This function must be called whenever a *ISUP_DEACTIVATE_CONF* primitive is received from the ***isupd*** daemon. This primitive is sent by ***isupd*** whenever an *ISUP_DEACTIVATE_REQ* is received by itself from its user. It returns the trunk group number that deactivation requested to its caller.

isup_get_trk_list()

This function is used to retrieve a trunk list after *ISUP_ACCEPT_TRUNK* and *ISUP_RELEASE_TRUNK* indications are received. The number of trunks in the list and the list of concerned trunks are returned to the caller.

isup_set_regtype()

This function sets an ISUP user's registration type to be used to reach a decision upon activation failure. The two kinds of registration types are hard and soft. If soft registration is requested and the ISUP user wants to register a trunk that was already activated by another ISUP user running on other hosts, then the activation fails. This result may not be desired: an ISUP user may want to purposely register to already activated trunk . In this case, hard registration must be requested.

isup_get_regtype()

This function gets an ISUP user's registration type to be used to reach a decision on activation failure. The two kinds of registration types are hard and soft. See isup_reg_req() on page 4-41 above.

isup_get_cic_no()

This function builds and returns a Circuit Identification Code (CIC) from its arguments, which are circuit number and group number, respectively. This function uses the current variant and protocol to calculate the correct CIC value.

isup_get_cct_no()

This function returns a circuit number from CIC (circuit identification code), which is given as an argument. This function uses the current variant and protocol to calculate the correct circuit number value.

isup_get_grp_no()

This function returns a circuit group number from CIC, which is given as an argument. This function uses the current variant and protocol to calculate the correct circuit group number value.

isup_get_dpc_stat()

This function is called to retrieve DPC status when a *ISUP_DPCSTATUS* indication is received. *ISUP_DPCSTATUS* is used by the local ISUP to inform Call Control that a change of congestion status occurred for the ISUP layer at the remote end of a trunk group.

isup_get_var_no()

This function is called to retrieve ISUP variant information. ISUP API library keeps track of changes in the ISUP variant that are triggered by potential changes in ISUP MO database configuration, and updates its internal data to reflect these changes. Call Control or maintenance systems can find out about the current variant setting by calling this function whenever necessary.

Other ISUP API functions are backward compatible. Since ANSI and ITU protocols are both supported from single object code, ANSI and ITU ISUP API functions and parameter structures were unified. During this unification, some parameter structure definitions were modified. These parameters are shown in Table 4-1.

Table 4-1: Modified Parameter Structures

Structure	Protocol	Obsolete Field	New Field
isup_prm_t	ANSI	rAs	rngANDsta
isup_prm_t	ANSI	SupplementaryLineInfo	SuppLINEinfo
isup_prm_t	ANSI	xNETselection	ansi_xNETselection
isup_prm_t	ANSI	UserToUserInfo	ansi_u2uinfo
isup_prm_t	ITU	xNETselection	itu_xNETselection
isup_prm_t	ITU	userTOuserInfo	itu_u2uinfo
isup_prm_t	ITU	msgtypePSA	msgtypePAM
CLDP_NUMBER_t	ANSI	rsv14numplan57spare88	spare14numplan57INN8
RDTN_NUMBER_t	ANSI	rsv14numplan57spare88	spare14numplan57INN8
CRG_NUMBER_t	ANSI	rsv14numplan57spare88	spare14numplan57INN8
CLDP_NUMBER_t	ANSI	DigitInfo	addsig
RDTN_NUMBER_t	ANSI	DigitInfo	addsig
CRG_NUMBER_t	ANSI	DigitInfo	addsig
CLGP_NUMBER_t	ANSI	scrind12pres34numplan57spare88	scrind12presres34numplan57INN8
LOCATION_NUMBER_t	ANSI	scrind12pres34numplan57spare88	scrind12presres34numplan57INN8
CLGP_NUMBER_t	ANSI	DigitInfo	addsig
LOCATION_NUMBER_t	ANSI	DigitInfo	addsig
ORGCLD_NUMBER_t	ANSI	rsv12pres34numplan57spare88	spare12presres34numplan57spare88
RDTG_NUMBER_t	ANSI	rsv12pres34numplan57spare88	spare12presres34numplan57spare88
ORGCLD_NUMBER_t	ANSI	DigitInfo	addsig
RDTG_NUMBER_t	ANSI	DigitInfo	addsig

4.5.8 JAIN ISUP API Support

Starting with Distributed7 1.3.0 release, JAIN (Java APIs for the Integrated Network) ISUP API is supported. JAIN ISUP API is implemented based on JSR 17 JAIN^{TM} ISUP Specification Proposed Final Draft, dated November 30, 2001. JAIN ISUP API supports the following ISUP specifications:

- ITU Q.761-Q.764, 1993
- ANSI T1.113.x, 1996
- ANSI T1.113.x, 1992

Some of the startup operations which are needed to be performed by the JAIN ISUP Listener application are listed in the following table. For the rest of the operations and the full JAIN ISUP API specification, please visit http://www.jcp.org/aboutJava/communityprocess/first/jsr017.

GetInstance	Returns an instance of a JainSS7Factory. This is a singleton type class so this method is the global access point for the JainSS7Factory.
setPathName	Sets the Pathname that identifies the location of a particular Vendor's implementation of the JAIN SS7 Objects. (e.g) com.newnet
createSS7Object	Returns an instance of a Peer JAIN SS7 Object identified by the supplied classname. This supplied classname is the lower-level package structure and classname of the required Peer JAIN SS7 Object.
setVendorName	Sets the vendor name for this stack. This name will be the Vendor's domain name inverted. e.g. com.newnet
setStackName	Sets the name of the stack as a string. e.g. "test"
setSignalingPointCode	Sets the Signaling point code of this stack.
setStackSpecification	Sets the stack specification. Possible values are: IsupConstants.ISUP_PV_ANSI_1992 IsupConstants.ISUP_PV_ANSI_1995 IsupConstants.ISUP_PV_ITU_1993
createProvider	Creates a vendor specific implementation (Peer) of JainIsupProvider and returns a reference to it. This newly created provider is then attached to the JainIsupStackImpl object in order to send and receive messages between the Provider and the stack.
addIsupListener	Adds a JainIsupListener to the list of registered Event Listeners being serviced by this JainIsupProvider object. This is also used to add new User Addresses being handled by an ISUP Listener so that a Listener can register for more than one User Address by repeatedly calling this method.

Distributed operation of JAIN ISUP Listener application is not supported with this release of Distributed7.

This page is intentionally blank.

Chapter 5: User/Kernel-space Data Distribution Methods

5.1 Chapter Overview

This chapter describes the user/kernel-space data distribution methods employed by Distributed7 in this release. It is important to note here that these distribution methods not only have been used internally by the Distributed7 system software (to meet the needs of the distributed SS7 protocol stack) but also are available for external use by user/kernel-space application programmers, in order to develop distributed application programs executing under the Distributed7 environment.

5.1.1 Need for Data Distribution

One of the major challenges in designing a distributed system involves distribution of the data used by the system. Overall, the data distribution method used directly impacts the reliability, scalability, and performance of the system. As an example, consider a distributed system where all critical pieces of data used by the system are kept in a centralized place. While this approach may lend itself better to scalability, it brings in serious concerns regarding system reliability (e.g., high-availability) and performance. That is exactly why real-time sensitive distributed systems usually rely on distributed data.

While data distribution can be achieved in several different ways, the distribution method chosen by Distributed7 relies on data replication as the primary means of data distribution. In this approach, each host machine comprising a Distributed7 environment is equipped with a local copy of the data and the replicated copies of data on the individual hosts are kept in sync at appropriate times.

The two immediate benefits of the aforementioned data distribution approach are high-availability and improved performance. The high-availability results from the fact that each host is now equipped with its own copy of the data; therefore, the failure of a particular host machine is not very likely to effect the execution of the programs running on other hosts in the network. Thus, the overall system should be able to survive individual host failures. The improved performance results from programs, executing on each host being able to access local copies of the data without consulting with a centralized entity for each and every data operation. This is not to say no data synchronization takes place between the individual hosts at all times. It simply states that most of the time access to local data is sufficient for the correct operations of the programs running on the individual hosts. If and when data changes occur, the replicated copies of data need to be synchronized by the system automatically.

A major concern associated with data replication involves data integrity, i.e., how to keep replicated copies of the data distributed in sync. This is where the consistency model employed by the data distribution framework, as well as the communication methods used, for synchronization become critical. Also, there is always the question of scalability, i.e., whether things get horrible when the number of hosts involved increase in number. The data distribution methods designed and implemented for Distributed7 are based on proven technologies such as the Distributed Shared Memory (DSM) framework and its kernel-space extensions, which is what makes them extremely reliable. As far as the scalability issue is involved, all distribution infrastructures used by Distributed7 have been verified to present acceptable behavior for networks comprising one to eight host machines, interconnected with a 10 Mbps Local Area Network (LAN). Performance of the Distributed7 product can be improved considerably by utilizing LAN technologies operating at higher speeds (100 Mbps or more).

5.1.2 User-space versus Kernel-space Data

In this section, a deliberate distinction is made between user-space and kernel-space data distribution. There are two reasons for this. First, and most important, most of the SS7-specific Distributed7 system software runs at the kernel-level; while some software components run at the user-space level. Addressing the data distribution needs of either kernel-space or user-space is, therefore, not acceptable. Methods of data distribution must be available for both domains. Secondly, the Distributed7 environment has always been an application-development platform. And now that it supports a distributed system architecture, it should also support the means for data distribution for application programmers, both as user-space and kernel-space.

5.2 User-space Data Distribution Methods

In the Distributed7 implementation of the SS7 protocol stack, the ISDN User Part (ISUP) resides at the user-space. When operating under a Distributed7 environment, both of these layers are required to maintain replicated copies of a significant part of their user-space data across multiple hosts in a consistent manner. Similarly, several other layers of the Distributed7 system/application software are in need of sharing user-space data when running under a distributed environment.

The Distributed Shared Memory (DSM) framework has been chosen as the primary means of user-space data distribution under a Distributed7 environment. This release utilizes the functionality provided by the DSM framework to distribute the user-space protocol data maintained by the ISUP layer.

5.2.1 Distributed Shared Memory (DSM) Framework

The DSM framework allows user-space application programs executing under a Distributed7 environment to share data with each other in an efficient manner. This framework can be viewed as a natural extension of the UNIX-standard IPC communication mechanisms (IPC shared memory and semaphores) to a distributed computing environment. In summary, it provides access, in a synchronized manner, to a distributed set of shared memory segments located on the individual hosts within a network. For all practical purposes, application programs are unaware of the fact that the data being manipulated is stored in multiple places (shared memory segments) across the network. The manipulation of the distributed data among application programs is coordinated through network-wide locks.

The DSM framework on Distributed7 comprises an API library, a long-lived system process, and a set of command-line utilities. The functionality provided by these components is as follows:

- The DSM API library provides the interface between the application programs and the internals of the Distributed7 DSM implementation; thus, enables user-space application programs executing under a Distributed7 environment to manage their data through a well-defined set of library routines. It hides the implementation details of the DSM subsystem from user applications.
- The DSM daemon process is responsible for creating, accessing, manipulating, and coordinating the use of shared memory segments across a network of hosts. Each host is equipped with a local copy of this process.
- The DSM command-line utilities provide the means for retrieving various pieces of information about the DSM segments instantiated by application programs.

The Distributed7 DSM implementation is a pure software implementation of the DSM technology; thus, it requires no special hardware. While this possibly impacts the performance of the Distributed7 DSM framework in a negative way, it makes the Distributed7 system software (as well as the user application programs developed under Distributed7) portable across multiple UNIX platforms without requiring any code changes and/or special hardware.

5.2.1.1 DSM Application Programming Interface

In this section, we list all user-level library functions comprising the DSM API library and provide a brief description for each one. For more information about the individual functions, refer to the on-line manual pages available under this Distributed7 release or the DSM API manual section.

dsm_get()

This function allows a user thread to create a new DKM segment or obtain access to an existing DSM segment of specified size.

dsm_attach()

This function allows a user thread to attach a previously created DSM segment to its process's virtual address space. It is only after this function is called, one can access and/or manipulate the DSM segment contents.

dsm_rdlock() / dsm_rdlock_rec()

These functions allow a user thread to instantiate a read-only lock across a specified region of a previously created/attached DSM segment so that the calling thread can safely read data from this region.

dsm_wrlock() / dsm_wrlock_rec()

These functions allow a user thread to instantiate a read-write lock across a user-specified region of a previously created/attached DSM segment so that the calling thread can safely read/write data through this region.

dsm_unlock()

This function allows a user thread to release a read-only or read-write type DSM lock acquired previously and propagate the data changes incurred (if any) in the contents of the DSM segment to remote hosts.

dsm_getopts()

This function allows a user thread to retrieve the current settings of optional parameters associated with a DSM segment created previously.

dsm_setopts()

This function allows a user thread to manipulate the settings of the optional parameters associated with a DSM segment created previously.

dsm_getstat()

This function allows a user thread to retrieve various pieces of information about a DSM segment created previously.

dsm_setstat()

This function allows a user thread to manipulate various pieces of information about a DSM segment created previously.

dsm_detach()

This function allows a user thread to detach from its process's virtual address space a previously attached DSM segment.

dsm_destroy()

This function allows a user thread to remove from the system a specified DSM segment identifier and destroy the DSM segment associated with it.

dsm_rule()

This function allows a user thread to define read/write rules across a DSM segment for improved performance.

dsm_unrule()

This function allows a user thread to clear a previously defined read/write rule.

5.2.1.2 DSM Command-Line Interface

In this section, we list all command-line utilities available as part of the Distributed7 DSM framework and provide a brief description for each one. For more information about them, readers are referred to the on-line manual pages available under this release, or the User Command section of this manual.

dsm_apidemo

This utility allows a user to exercise and demonstrate the basic set of capabilities provided as part of the Distributed7 DSM API library.

dsm_audit

This utility allows a user to place a manual request to audit the dynamic data and/or communication resources associated with the Distributed7 DSM framework on a specified host.

dsm_bm

This utility allows a user to benchmark the run-time performance of the Distributed7 DSM framework in terms of the total number of DSM read-write operations that can be performed within a specified time interval.

dsm_list

This utility allows a user to retrieve and display information about the dynamic data records maintained by the DSM sub-system on a specified host.

dsm_rm

This utility allows a user to place a manual request to remove from the system a specified DSM segment identifier and destroy the DSM segment associated with it.

dsm_stat

This utility allows a user to retrieve and display information about the individual IPC shared memory segments comprising a DSM segment.

5.2.1.3 DSM Data Consistency Model

The DSM framework is *release-consistent*. Changes incorporated in the contents of a DSM segment are propagated to all remote hosts involved before the associated lock is released and the calling thread continues with its execution.

5.2.1.4 DSM Reliability Measures

The DSM framework features several built-in capabilities to assure reliability.

The most noteworthy one of these capabilities involves the DSM auditing capability. The DSM auditing capability has been designed to ensure the integrity as well as the consistency of various pieces of dynamic data associated with the DSM framework and maintained by the DSM daemon on behalf of application processes. Examples of such data include application service records, lock records, segment records, and user address records. The frequency of these audits varies based on the nature of the data being audited. For example, lock records are audited more frequently than other pieces of dynamic data in an effort to identify and release the leftover DSM locks as quickly as possible, so that other threads waiting to read/write data through the effected DSM region can continue their execution.

Other reliability measures of the DSM infrastructure involve automatic detection of application termination and/or remote host crash by the DSM daemon and release of the DSM resources, i.e., lock and address records, associated with the application and/or the remote host. The former reliability measure applies to both Distributed7 and non-Distributed7 applications (UNIX processes that are not registered with the Distributed7 platform). For Distributed7 applications, detection of the application process termination (and clean-up of the DSM resources) is instantaneous, it takes advantage of the Distributed7 asynchronous event detection capabilities. For non-Distributed7 applications, it may take up to a minute to detect the application termination and clean-up the DSM resources associated. The DSM daemon polls non-Distributed7 applications every minute to verify their existence).

Last but not least, the DSM framework contains intelligence to release all appropriate resources associated with a particular DSM segment when the segment is detached and/or destroyed.

5.2.1.5 DSM Multi-Threading Support

The DSM API library is `MT-SAFE`; thus, it can be used safely by multi-threaded application programs. Applications that can tremendously benefit from this aspect of the DSM library are those that can perform DSM related operations with individual threads in a concurrent manner. For example, consider a DSM application where the individual lock operations pertaining to a specified DSM segment can be performed by multiple threads in a concurrent manner. To force such an application to a single-threaded one is likely to reduce the performance of the application significantly. This is because the successive DSM lock operations will need to be performed by the application in a serial manner. If, instead, this application can operate in the form of multiple threads where all threads perform their lock operations in a parallel manner, the performance of the application should be significantly higher, as it can now acquire/release more locks in a given time interval.

5.2.1.6 DSM Performance Considerations

The Application Programming Guide for the DSM API Library contains a thorough performance analysis of the DSM framework, both for single-threaded and multi-threaded applications. It also contains hints about potential means of performance improvement when using the DSM framework.

5.3 Kernel-space Data Distribution Methods

In the Distributed7 implementation of the SS7 protocol stack, MTP, SCCP, and TCAP protocol layers are embedded mostly in the kernel-level code. When operating under a distributed environment, all of these protocol layers will be required to maintain replicated copies of a significant part of their kernel-space data across multiple hosts in a consistent manner. The Distributed Kernel Memory (DKM) and Distributed Record Access (DRA) frameworks are intended to address the kernel-space data distribution problems faced by this release of the Distributed7 product in a generic way.

The DKM framework constitutes the primary means of maintaining replicated copies of kernel-resident data that are in the form of DKM segments. The DRA framework, on the other hand, builds upon the DKM and is intended to fulfill the needs of database-oriented kernel-resident Distributed7 applications. It is through the DRA framework, a kernel application can view its kernel-resident data in the form of a distributed database and operate on it.

5.3.1 Distributed Kernel Memory (DKM) Framework

The DKM framework allows the kernel-resident Distributed7 software executing on individual hosts within a distributed environment to operate on replicated copies of kernel-space data in a coordinated manner. Thus, while the kernel-resident Distributed7 software executing on each host machine performs its data operations through local copies of one or more DKM segments, the DKM framework facilitates synchronization of replicated copies of these segments on all other hosts at all appropriate times.

In essence, a DKM segment consists of replicated copies of kernel-space memory allocated, either statically or dynamically, on the individual hosts comprising a Distributed7 environment. Kernel threads executing on the individual hosts can share information with each other by reading/writing data through local copies of a DKM segment. To prevent inconsistencies and/or collisions when reading/writing through a DKM segment, kernel threads must always use explicit synchronization variables (DKM read/write locks) when reading and writing data through local copies of the segment.

From a conceptual point of view, the DKM framework provides the kernel-resident Distributed7 software executing on different hosts with a means of inter-host communication that is equivalent to the communication the DSM framework provides user-space software with. If the Distributed7 implementation of the SS7 protocol stack were to be done completely at the user-space, in the form of UNIX processes, there would be no real need for the DKM framework. However, given that most of the Distributed7 SS7 intelligence resides in the kernel-space code, in the form of STREAMS components, the DKM framework becomes an essential part of this Distributed7 release, as it provides a standardized means of sharing kernel-space data among kernel-resident Distributed7 software executing on different hosts.

From an implementation point of view, the DKM framework consists of a new module (a STREAMS multiplexer) and a set of kernel-level library routines (referred to as the DKM library routines) that are defined as part of the kernel-resident software for this module. This DKM module is linked above the SPM multiplexer through a separate stream - see Figure 5-1. It is through this software architecture, the DKM module can readily make use of the kernel-level TCP/IP communication capabilities when communicating with its peers on remote hosts.

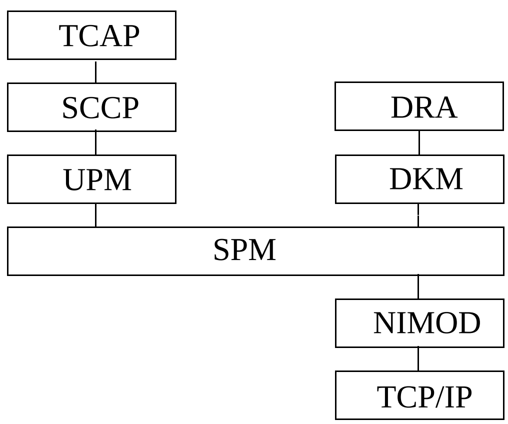


Figure 5-1: STREAMS Architecture: DKM/DRA

The interface between the DKM framework and the other kernel-resident Distributed7 software is through the kernel-level DKM library routines. The DKM library routines execute as part of the calling kernel thread and, whenever necessary, they result in messages to be exchanged with other hosts through the DKM module on the local host. Note that the communication with (and through) the local DKM module is transparent to the kernel thread that invokes the DKM library routine, as this intelligence is embedded in the DKM library routine itself. Users of the DKM library are unaware of the fact that additional STREAMS modules are involved in processing DKM related requests.

The DKM API library interface is an open interface and can potentially be used by all kernel-level application programmers, that are interested in writing STREAMS programs for the Distributed7 environment, without knowing the details of the kernel-resident Distributed7 software architecture.

5.3.1.1 DKM Application Programming Interface

In this section, we list all kernel-level library functions comprising the DKM API library and provide a brief description for each one. For more information about the individual functions, readers are referred to the on-line manual pages available under this release or the DKM section of the API Reference Manual.

dkm_get()

This function allows a kernel thread to create a new DKM segment or obtain access to an existing DKM segment of specified size.

dkm_extend()

This function allows a kernel thread to make dynamic extensions to an existing DKM segment.

dkm_getlist()

This function allows a kernel thread to retrieve the list of extensions associated with a DKM segment.

dkm_lock()

This function allows a kernel thread to instantiate a read-only or read-write type lock across a specified region of a previously created DKM segment so that the calling thread can safely read/write data through this region.

dkm_sync()

This function allows a kernel thread to initiate a manual request to propagate the changes made to the contents of a DKM segment and/or segment extension to all remote hosts involved while in possession of a read-write lock.

dkm_unlock()

This function allows a user thread to release a read-only or read-write type DKM lock acquired previously and propagate the data changes incurred (if any) in the contents of the DKM segment to remote hosts.

dkm_schedule()

This function allows a kernel thread to initiate an asynchronous request to acquire a DKM lock.

dkm_cancel()

This function allows a kernel thread to cancel a pending asynchronous DKM lock request.

dkm_query()

This function allows a kernel thread to retrieve various pieces of information about the data blocks comprising a DKM segment and/or segment extension.

dkm_gethostid()

This function allows a kernel thread to retrieve the logical host number associated with individual hosts comprising a distributed environment.

dkm_gethostaddr()

This function allows a kernel thread to retrieve the Internet Protocol (IP) address associated with a particular logical host number.

dkm_notify()

This function allows a kernel thread to register (or cancel a former registration) for the DKM event notification capability. It is with this capability, a kernel thread can be informed about DKM related activities performed by other kernel threads.

dkm_shrink()

This function allows a kernel thread to delete an existing DKM segment extension and de-allocate the kernel-space memory associated with the extension. The identifier associated with the segment extension is also removed from the system.

dkm_destroy()

This function allows a kernel thread to remove from the system a specified DKM segment identifier and destroy the DKM segment associated with it.

5.3.1.2 DKM Command-Line Interface

In this section, we list all command-line utilities available as part of the Distributed7 DKM framework and provide a brief description for each one. For more information about them, readers are referred to the on-line manual pages available under this release or the User Command section of this manual.

dkm_apidemo

This utility allows a user to exercise and demonstrate the basic set of capabilities provided as part of the Distributed7 DKM framework.

dkm_bm

This utility allows a user to benchmark the run-time performance of the Distributed7 DKM frameworks in terms of the total number of DKM read-write operations that can be performed within a specified time interval.

dkm_dump

This utility allows a user to retrieve and display the contents of a particular DKM segment and/or segment extension.

dkm_list

This utility allows a user to retrieve and display information about the dynamic data records maintained by the DKM sub-system on the local host.

dkm_rm

This utility allows a user to place a manual request to remove from the system a specified DKM segment identifier and destroy the DKM segment associated with it.

dkm_sar

This utility allows a user to activate the optional statistics collection capability that is available as part of the Distributed7 DKM framework.

dkm_stat

This utility allows a user to retrieve and display information about the individual data blocks comprising a DKM segment and/or segment extension.

5.3.1.3 DKM Data Consistency Model

The DKM framework can be configured to support either *release-time-consistency* or *acquire-time-consistency* as follows:

- To achieve *release-time-consistency*, all kernel threads must utilize the *sync-first* option when releasing a DKM read-write lock. This will assure that the changes incurred in the segment contents get propagated to remote hosts prior to releasing the lock associated. Note that the execution of the calling thread will be blocked until the data update operation is completed.
- To achieve *acquire-time-consistency*, all kernel threads must utilize the *sync-later* option when releasing a DKM read-write lock. This implies that the changes incurred in the segment contents can be propagated to remote hosts after releasing the lock associated. Note that in this case while the kernel thread that releases the lock can continue its execution right away, other kernel threads that are interested in accessing the same DKM region will be blocked until the contents of the region get updated.

The ultimate decision of choosing one consistency model over the other belongs to the kernel application itself. Note however that the *acquire-time-consistency* model is only applicable to *local* (non-exclusive) read-write locks.

Another consistency related issue is the use of local read-write locks. The basic idea behind local read-write locks is to permit kernel threads executing on remote hosts to read through a locked DKM region while the region is kept locked for read-write purposes on the local host, thus, allowing thread concurrence across multiple hosts. It should be well understood that this concurrence brings in some inconsistency to the overall DKM framework. While the kernel-space data on a specified host is in the process of being modified, threads on other hosts continue to read through its earlier (outdated) versions and this may not be acceptable for certain applications and/or certain pieces of data. If this turns out to be the case, global read-write locks should be used instead.

5.3.1.4 DKM Reliability Measures

The DKM framework contains built-in intelligence to detect a remote host crash and release all appropriate DKM resources associated with kernel threads executing on that host. It is also pre-programmed to perform all appropriate resource clean-up chores upon deletion of a DKM segment and/or segment extension.

An extremely important issue regarding the overall DKM reliability involves the DKM event notification capability. The DKM framework has no means of knowing whether a particular kernel thread, that has signed up for DKM event notification, stops attending the STREAMS queue that is specified during the sign-up procedure. Therefore, it remains the DKM user's responsibility to cancel any former registration for DKM event notification if and when it loses interest in DKM events, for whatever reason. A failure to do so is likely to result in system crashes, when an attempt is made by the system to post a DKM event indicator message on the STREAMS queue specified.

5.3.1.5 DKM Multi-Threading Support

The DKM infrastructure operates in the **`MT-EFFICIENT`** mode. This is to be able to service the requests initiated by multiple DKM API users concurrently.

The DKM framework does not make any assumptions on the side of the DKM users in regard to multi-threaded operations. Provided that a DKM user operates in the **`MT-SAFE`** mode, the level of concurrence within the module is immaterial to the DKM framework. The DKM multiplexer does not know anything about the level of concurrence within a user module. Care must be taken however on the side of the DKM user in regard to asynchronous events that may take place during processing of requests initiated by threads running as part of that user. These events consist of DKM events triggered by the DKM users on the local host or remote hosts and responses to the asynchronous lock requests placed by that DKM user. If and when a DKM user operates in the multi-threaded mode, it is the user's responsibility to take care of asynchronous events generated by the DKM multiplexer and associate them with the appropriate threads.

5.3.1.6 DKM Performance Considerations

The performance of the Distributed7 DKM framework is far superior than that of the DSM framework for the following reasons:

- Acquiring and/or releasing DKM locks for read-only purposes does not necessitate any inter-host communication; therefore, is expected to be extremely fast.
- The DKM framework provides kernel threads with an option to acquire *local-only* (as opposed to *global*) read-write locks. Acquiring a local-only read-write lock does not normally necessitate any inter-host communication. This translates to a direct savings in the time it takes to acquire a read-write lock.
- The DKM framework may be configured to support *acquire-time-consistency* as opposed to *release-time-consistency*. Note that in the former case, the propagation of changes made to a DKM segment takes place after the associated read-write lock is released and before the region is accessed by another thread for read/write purposes. This in return reduces the time it takes to release a read-write lock and makes it independent of the actual size of the region being updated. Provided that there are no pending locks for the region being updated, the overall performance of the system will be far superior (from the users point of view) compared to a scheme where the update operation precedes the release of the lock.
- Inter-host communication between the DKM modules on the individual hosts comprising a Distributed7 environment is through the kernel-level TCP/IP interface; thus, does not involve any user-space software. Stated in other terms, the DKM framework does not face the performance penalties involved in invoking UNIX system calls.

5.3.2 Distributed Record Access (DRA) Framework

Distributed Record Access (DRA) is an additional layer that provides structured and fast access to distributed data resources. It is implemented as a STREAMS module, which also provides a set of library procedures that use the DKM interface. The DRA module is pushed onto the first endpoint of the DKM driver by ***dkmd***.

DRA provides a record-oriented interface towards DKM data. These records are accessed with indexed-search operations based on the record key. DRA supports the notion of a *logical* record which can consist of *distributed* and *non-distributed* (a.k.a., *private*) data parts. DRA indexes are kept on non-distributed memory, and these indexes are kept up-to-date by using a DKM event triggering mechanism.

DRA provides two different types of indexed access to DKM data as follows:

- Sorted Access - Index portion is kept as a sorted table. Record search is performed with a binary search. The index portion is re-sorted after every record add/delete operation. This method has negligible data overhead (it does not use pre-allocated data, index entries are created along with data entries), however, it is slow in adding/removing records. It should be used for slow growing/shrinking discrete-keyed data.
- Hashed Access - Index portion is kept as a hash table. Record search is performed by a hash function applied to the requested key. It has considerable data overhead, since it pre-allocates the index part depending on the expected number of records, but it is fast in record addition/deletion. This method should be preferred for fast growing/shrinking numeric-keyed data.

Creating DRA segments is the only way of using DRA services. From a user point of view, a DRA segment is only a collection of records, whereas from a physical point of view, it consists of a DKM segment, any number of DKM extension segments, and buffers of non-distributed memory where record private portions, record indexes and record usage information is stored.

A mutual exclusion structure is provided for every used DRA record, this structure is used to resolve local race conditions on the record private part (DKM lock mechanisms are relied on for handling race conditions on the record distributed part). DRA also supports protected access of a memory portion which can be allocated on a per-segment basis. This can be used to store global information about the segment itself.

All the index data are created in non-distributed memory and are updated by the DRA layer with events generated by the DKM driver. All kinds of inconsistencies which result from using indexes on non-distributed memory to access distributed data are resolved by the DRA layer.

5.3.2.1 DRA Application Programming Interface

In this section, we list all kernel-level library functions comprising the DRA API library and provide a brief description for each one. For more information about the individual functions, readers are referred to the on-line manual pages available under this release or the DRA section of the API Reference Manual.

dra_construct()

This function allows a kernel thread to construct a new DRA segment, with the specified record definition, an optional segment private data buffer, a primary index and an optional secondary index.

dra_new_record()

This function allows a kernel thread to create a new DRA record with the given record key(s) and lock this record for read/write operations.

dra_find_record()

This function allows a kernel thread to locate and optionally lock the DRA record associated with a specified key.

dra_find_inseq()

This function allows a kernel thread to locate and optionally lock the set of DRA records associated with a specified partial key.

dra_del_record()

This function allows a kernel thread to locate and delete the DRA record associated with a specified key.

dra_del_locked()

This function allows a kernel thread to delete a previously locked DRA record.

dra_relock_sync()

This function allows a kernel thread to upgrade/downgrade the lock type associated with a previously located/locked DRA record.

dra_relock_async()

This function allows a kernel thread to place an asynchronous request to read/write lock a previously located/locked DRA record.

dra_rls_lock()

This function allows a kernel thread to release a previously locked DRA record.

dra_lock_seg_priv()

This function allows a kernel thread to lock the private data buffer of a DRA segment.

dra_rls_seg_priv()

This function allows a kernel thread to unlock the private data buffer of a DRA segment.

dra_validate()

This function allows a kernel thread to check on the consistency of a DRA record that is accessed without acquiring a lock of appropriate type.

dra_get_dkm_id()

This function allows a kernel thread to obtain the DKM segment ID associated with a particular DRA segment.

dra_get_dkm_addr()

This function allows a kernel thread to obtain the DKM address associated with a particular DRA record so that direct DKM function calls can be placed by the calling thread.

dra_destroy()

This function allows a kernel thread to destroy an existing DRA segment.

5.3.2.2 DRA Command-Line Interface

In this section, we list all command-line utilities available as part of the Distributed7 DRA framework and provide a brief description for each one. For more information about them, readers are referred to the on-line manual pages available under this release or the User Command section of this manual.

dratest

This utility allows a user to exercise and demonstrate the basic set of capabilities provided as part of the Distributed7 DRA framework. ***dratest*** is a tcl shell with add-on commands to exercise DRA functionality.

Some of the DRA related commands are implemented as TCL scripts, and they reside in file ***$EBSHOME/access/bin/dra.tcl***. This file can be modified to add new ***dratest*** commands, or to modify the existing ones.

Following is the list of DRA related ***dratest*** commands:

- **dra_all_segs**: Display detailed information about all DRA segments, including segment pointers, segments names, and segment keys.
- **dra_construct**: Construct a new DRA segment or attach to the segment, if it already exists.
- **dra_del_locked**: Delete a previously locked DRA record.
- **dra_del_rec**: Delete a DRA record with the given key.
- **dra_destroy**: Destroy a DRA segment
- **dra_find_inseq**: In-sequence find for a number of DRA records with the requested key prefix.
- **dra_find_rec**: Find and lock a DRA record with the given key.

- **dra_new_rec**: Create a new DRA record, assign the record key(s) and read/write lock the record.
- **dra_relock_async**: Request asynchronous read/write lock for a previously located DRA record.
- **dra_relock_sync**: Request lock upgrade/downgrade for a previously located DRA record.
- **dra_rls_lock**: Release a locked DRA record.
- **dra_seginfo**: Display detailed segment info for the requested DRA segment.
- **dra_seglist**: Display short info. for all the DRA segments
- **dra_setopts**: Set DRA debug options and debug level. Used to enable/disable DRA debug messages.
- **dra_validate**: Validate the sanity of a DRA record accessed without locking.
- **get_mem**: Display the contents of a kernel memory buffer.
- **set_mem**: Modify the contents of a kernel memory buffer.

5.3.2.3 DRA Data Consistency Methods

Apart from the DKM provided data consistency methods, DRA provides a “safe mode of operation”. This feature is useful for applications that access distributed data without specifying any DKM related lock flags (without locking record distributed part) The safe-lock mode of operation can be used with all sorts of lock and delete operations, and is used to mark the record throughout the network as “being modified” prior to modification of its contents. If a record is safe-locked, all new requests to locate (or validate) the record will fail with a certain error code.

5.3.2.4 DRA Reliability Measures

Consistency of internal DRA data completely relies on the DKM event notification mechanism. DRA registers to DKM and processes all the DKM events (if they are related to any of the locally created DRA segments). If a DKM segment is destroyed, corresponding DRA resources are removed. DKM shrink and extend events are used to allocate/de-allocate the DRA counterparts of DKM extension segments, and DKM sync. events are used to keep track of added/removed DRA records (by remote hosts).

DRA keeps a copy of record usage information and copies of record keys for used records in non-distributed data. It is through the usage of this data (and the related indexes) that DRA provides the requested services. DRA also handles inconsistencies between private DRA data and distributed DRA data (on the DKM segment). These inconsistencies are handled in different ways:

- Normal: The inconsistency is decided to be acceptable, DRA proceeds with the ongoing action (if DRA private data indicates a record is not used, whereas the status of the DRA record on the DKM segment is marked as used).
- Incorrect Record: The inconsistency effects the usage of a particular DRA record, record is deleted (if a record addition event is received and the private DRA data indicates that the record is already allocated).

- Fatal: Inconsistency effects the whole DRA segment, the segment is destroyed (an extension event received for an already used extension segment identifier).

5.3.2.1 DRA Multi-Threading Support

DRA framework does not make any assumptions about the level of concurrence of a user module. All access to DRA data is properly protected (including DKM event handling), and all the DRA provided user entities (DRA records and segment private data area) have means of mutual exclusion, if requested by the user.

5.3.2.2 DRA Performance Considerations

From a performance point of view DRA adds the following items to DKM related considerations:

- Fast access to non-frequently modified data with the use of “dra-nolock” and “dra-safelock” options.
- Quickly locating the requested address regions with the use non-distributed index data.
- Private only locks to that part of the data, if the ongoing action does not necessitate distributed locks.
- For hashed primary indexes, the DRA key assignment service provides the user with record keys that can be accessed most efficiently.

This page is intentionally blank.

Chapter 6: Operations

6.1 Chapter Overview

This chapter provides detailed coverage of the following:

- Starting the Software
- Shutting Down
- Using MMI/MML
- Using SNMP
- Using AccessMOB
- Using the Command File Navigator
- Stand-alone Operation
- Process Management
- Configuration
- Viewing the Status of System Processes
- Using OMAP

6.2 Starting the Software

This section describes how to start the software manually. The software can be set up to start automatically at boot-up. The *Installation and Maintenance Manual* provides the instructions for automatic start-up.

6.2.1 Starting in a Distributed Environment

In a distributed environment, there can be more than one instance of a Managed Object Server (MOS). The first instance started becomes the global instance. All other instances are local and receive configuration information from the global instance. If the global instance is terminated, one of the local instances will become the global instance and provide configuration information to all new instances. If all the instances are terminated, then the instance with the most recent termination time becomes the global instance upon restart, and all other instances are re-synchronized. Prior to synchronization, all instances, other than the global instance, will erase their local databases. If the global instance (first instance to start) is not configured, all subsequent instances (previously configured or not) will be synchronized using an empty database, even if the local MOS had a valid database from the previous session.

The following example steps *must* be followed to avoid deleting the contents and configuration of the Managed Object Server (MOS) databases:

(The example uses a network configuration with two hosts (A and B) running Distributed7.)

4. MOSx databases on host-A and host-B are empty
5. MOSx on host-A is started and it becomes the global instance
6. MOSx on host-A is configured with CONFx
7. MOSx on host-B is started
8. The global instance on host-A will synchronize the newly started MOSx on host-B with CONFx
9. MOSx on both hosts are stopped
10. MOSx on host-B with CONFx is restarted and becomes the global instance
11. MOSx on host-A with CONFx is restarted
12. MOSx on host-A will erase its database and get synchronized by the global instance on host-B with CONFx

Important*: It is possible for a configuration on all hosts to be erased if the MOS on the first host is started, configured, stopped, and then the second host is started and not configured, and the first host is started again. This happens because the host that has been configured is no longer the global instance of the MOS and will synchronize with the empty configuration from the global instance, the second host.*

Important*: Users of Distributed7 must not change the system clock, i.e., date/time, of any host machine while the Distributed7 system software is running on that machine. In a distributed product configuration, this rule applies to all machines that are part of the distributed product.*

6.2.2 Manual Start-up of the Software

All hardware and software must already be installed.

6.2.2.1 New Installation Distributed7 Start-up

These steps should be followed if Distributed7 has just been installed and not yet started.

1. Verify that the environment variables have been set:
   ```
   echo $EBSHOME
   echo $PATH
   ```
2. If the environment variables are not set, set them:
   ```
   setenv EBSHOME <software_installation_directory>
   setenv PATH ${PATH}:$EBSHOME/access/bin
   ```
3. If you want to start the software from a null configuration for a specific signalling point (SP), remove all files from the corresponding ***$EBSHOME/access/RUN#/DBfiles*** directory (where # is the SP number). Otherwise, the system will be started with the configuration files that exist in the corresponding directory.
4. If the start-up configuration file (***$EBSHOME/access/RUN/config/PMGR/*** *apmconfig*) needs to be customized, modify it to start the appropriate Distributed7 daemon and application processes. A sample of the default file is listed on page 7-46. Also, the ***/etc/nsswitch.conf*** file must be modified along with */ect/host*.
5. Make certain you have root privileges, then type `ebs_config`.
6. Choose `(1)distributed` for the distributed product configuration.
   a. Copy ***license.dat*** to the ***$EBSHOME/access/etc*** directory.
   b. Run ***ebs_tune*** from the ***$EBSHOME/access/install*** directory.
   c. Set TCL environment for GUI:
      ```
      setenv TCL_LIBRARY $EBSHOME/access/lib/tcl_lib
      setenv TK_LIBRARY $EBSHOME/access/lib/tk_lib
      ```
   d. `setenv MANPATH ${MANPATH}:${EBSHOME}/access/manpages`
   e. Reboot host for ***ebs_tune*** parameters to take effect in ***/etc/system***.
   f. If the SS7 board is installed, then run ***getcfg*** to ge the instance number that is used for configuring SS7 in MMI/MML or AccessMOB interface.
7. To start the Distributed7 software, enter: `apm_start`

8. The following banner appears during the start-up process:

```
+++++++++++++++++++++++++++++++++++++++++++++++++++
+                                                 +
+    <<<<<<<<< IMPORTANT NOTE >>>>>>>>>>          +
+                                                 +
+   Operations of the Application Process Manager +
+   are now suspended indefinitely !!!            +
+                                                 +
+   To continue, use either MMI/MML or AccessMOB  +
+   interface to set the product operation mode   +
+   (i.e., stand-alone vs. distributed) and       +
+   introduce any remote host machines involved.  +
+                                                 +
+   It is only after you perform these tasks,     +
+   Application Process Manager will resume its   +
+   normal operations & proceed with its routine  +
+   system software initialization/start-up ...   +
+                                                 +
+++++++++++++++++++++++++++++++++++++++++++++++++++
```

9. Verify that the mandatory daemon processes are running properly. Enter: `ebs_ps`
The apmd, mlogd, spmd, and netd processes should be listed.
10. Start MMI/MML <sp#> or AccessMOB <sp#>.
11. Use the ADD-HOST command to add all the hosts to the distributed network. The operations of the *apmd* daemon will be suspended indefinitely until this step is completed. Once the all hosts are added, *apmd* will continue with its start-up procedures and spawn the remaining set of mandatory daemon processes.
12. Verify that the remaining mandatory daemon processes are running properly. Enter: `ebs_ps`. *The **alarmd**, dsmd, and dkmd processes are listed*.
13. If TCAP over TCP/IP will be in use, configure TCP/IP connections to all appropriate remote hosts. Use the MODIFY-TCPCON (***page 9-144***) command in MMI/MML or the TCPCON managed object icon in AccessMOB. Make sure that the ***tcmod*** module is pushed over all such TCP/IP connections.
14. Start the SS7 node processes associated with the SP of interest (e.g., to start SS7 node processes associated with SP 0, enter: **`apm_setstate sp0u`**). It is important to note here that the default *apmconfig* file contains the rules to start and stop the SS7 node processes for all signalling points; therefore, they can be started/stopped using the *apm_setstate* command and specifying an appropriate state.
15. Verify that the required SS7 node processes are running properly. Enter: `ebs_ps`
The upmd, and scmd processes associated with your SP should be listed.
16. If TCAP applications will be used, start up the *tcmd* daemon.
17. If the correct processes are not listed, enter: `apm_stop`
Go to Step 7.

18. The software is now running. If this is the first time it is started, it must be configured further before any other activities are performed.

6.2.2.2 Distributed Start-up (existing installation)

If the system is not set up for automatic start-up or the software has been stopped with the *apm_stop* command, follow the steps in this section to properly start the software.

1. To start the Distributed7 software, enter: `apm_start`
2. To verify that the mandatory daemon processes are running properly, enter: `ebs_ps`
   The apmd, mlogd, spmd, netd, alarmd, dsmd, and dkmd processes are listed.
3. If TCAP over TCP/IP will be in use, then configure TCP/IP connections to all appropriate remote hosts. Use the MODIFY-TCPCON (***page 9-144***) command in MMI/MML or the TCPCON managed object icon in AccessMOB. Make sure that the ***tcmod*** module is pushed over all such TCP/IP connections.
4. Start the SS7 node processes associated with the SP of interest (e.g., to start SS7 node processes associated with SP 0, enter: **`apm_setstate sp0u)`**. The default *apmconfig* file contains the rules to start and stop the SS7 node processes for all signalling points. They can, therefore, be started/stopped using the *apm_setstate* command and specifying an appropriate state.
5. To verify that the required SS7 node processes are running properly, enter: `ebs_ps`
   The upmd and scmd processes associated with your SP are listed.
6. If TCAP applications are used, then start up the *tcmd* daemon process.
7. If the correct processes are not listed, then enter `apm_stop`, and go to Step 7 in Section 6.2.2.1.
   The software is now running.

6.2.2.3 Simplex Start-up

1. Verify that the environment variables have been set:
   `echo $EBSHOME`
   `echo $PATH`
2. If the environment variables are not set, set them:
   `setenv EBSHOME <software_installation_directory>`
   `setenv PATH ${PATH}:$EBSHOME/access/bin`
3. If you want to start the software from a null configuration for a specific signalling point (SP), remove all files from the corresponding ***$EBSHOME/access/RUN#/DBfiles*** directory (where # is the SP number). Otherwise, the system will be started with the configuration files that exist in the corresponding directory.
4. If the start-up configuration file (***$EBSHOME/access/RUN/config/PMGR/****apmconfig*) needs to be customized, modify it to start the appropriate Distributed7 daemons and application processes. A sample of the default file is listed on page 7-46. Also, the ***/etc/nsswitch.conf*** file must be modified.
5. Make certain you have root privileges, and then type `ebs_config`.
6. Choose **`(2)simplex`** for the Simplex product configuration.

7. To start the Distributed7 software, enter: `apm_start`
8. To verify that the mandatory daemon processes are running properly, enter: `ebs_ps`
   The apmd, mlogd, spmd, netd, alarmd, dsmd, and dkmd processes are listed.
9. If TCAP over TCP/IP will be in use, then configure TCP/IP connections to all appropriate remote hosts. Use the MODIFY-TCPCON (***page 9-144***) command in MMI/MML or the TCPCON managed object icon in AccessMOB. Make sure that the ***tcmod*** module is pushed over all such TCP/IP connections.
10. Start the SS7 node processes associated with the SP of interest, i.e., to start SS7 node processes associated with SP 0, enter: **`apm_setstate sp0u`**. The default *apmconfig* file contains the rules to start and stop the SS7 node processes for all signalling points; therefore, they can be started/stopped using the *apm_setstate* command and specifying an appropriate state.
11. To verify that the required SS7 node processes are running properly, enter: `ebs_ps`
    The upmd, and scmd processes associated with your SP are listed.
12. If TCAP applications are used, then start the *tcmd* daemon process.
13. If the correct processes are not listed, then enter `apm_stop`, and go to Step 7 in Section 6.2.2.3.
14. The software is now running. If this is the first time it is started, it must be configured further before any other activities are performed.

6.3 Shutting Down

To prevent damage to files or components, shut down the system according to the steps below:

1. Log in as *superuser.*
2. To stop the SS7 node and Distributed7 processes, enter: `apm_stop`
3. To stop the Solaris operating system, enter: **`init 0`**
4. To stop a Motorola AIX operating system, enter: **`shutdown`**
5. Once the operating system is done shutting down, turn the power switch off.

6.3.1 Using the SIGTERM and SIGKILL Signals

If the system cannot be shut down as outlined in the steps above, or if an individual process must be terminated, use the SIGTERM and SIGKILL signals. Distributed7 daemons treat the SIGTERM and SIGKILL signals in the following ways:

6.3.1.1 Treatment by apmd

SIGTERM

apmd interprets the SIGTERM signal as a graceful request to shutdown Distributed7 software on the local host. On receipt of the SIGTERM signal, *apmd* initiates a local shutdown request, waits until all processes that it spawned terminate, and then terminates gracefully with an exit code of 0.

SIGKILL

apmd cannot interpret the SIGKILL signal. When SIGKILL is sent to *apmd*, the result is premature termination of the *apmd* daemon process. On *apmd* termination, the Distributed7 platform initiates a local Distributed7 software shutdown.

If the user wants *apmd* to continue running, then an entry should be inserted in the **/etc/inittab** file to initiate re-spawning from there so that the UNIX ***init*** daemon process monitors it.

6.3.1.2 Treatment by mlogd

SIGTERM

mlogd interprets the SIGTERM signal as a graceful request to terminate. On receipt of the SIGTERM signal, *mlogd* terminates with an exit code of 0. Because *apmd* recognizes this as a graceful termination, *mlogd* is not automatically re-spawned by *apmd* following termination. Note that termination of *mlogd* does not result in any major malfunction in system operations. However, it is no longer possible to make entries in the **mlog** files that are maintained by the *mlogd* daemon process. Log messages submitted by other daemons while the *mlogd* daemon process is down accumulate in an IPC message queue, provided there is room. If there is no room on the message queue, log messages submitted by other daemons are lost. When *mlogd* restarts, it reads all messages accumulated on the IPC message queue, and makes appropriate entries in the ***mlog*** file.

SIGKILL

mlogd cannot interpret the SIGKILL signal. When SIGKILL is sent to *mlogd*, the result is premature termination of the *mlogd* daemon process. Termination is detected by *apmd*, which automatically re-spawns *mlogd* because of a FAILSAFE process defined in the *apmconfig* file.

6.3.1.3 Treatment by netd

SIGTERM

netd interprets the SIGTERM signal as a graceful request to terminate when there are no TCP/IP connections to remote host(s). *netd* receives the SIGTERM signal, and terminates with an exit code of 0.

netd ignores the SIGTERM signal when there are TCP/IP connections to one or more remote hosts—unless the local machine is in the process of shutting down. If local shutdown is in progress, *netd* interprets the SIGTERM signal as a graceful request to terminate. *netd* receives the SIGTERM signal, and terminates with an exit code of 0.

SIGKILL

netd cannot interpret the SIGKILL signal. When SIGKILL is sent to *netd*, the result is premature termination of the *netd* daemon process with a non-zero exit code.

- Stand-alone mode:
  When *netd* terminates gracefully, the *apmd* daemon process does not initiate any actions. However, if *netd* does not terminate gracefully, the *apmd* daemon process re-spawns it.
- Distributed mode:
  The Distributed7 platform initiates an automatic local Distributed7 software shutdown upon termination of the *netd* daemon process, whether the termination was graceful or not. This means that termination of *netd* through SIGTERM or SIGKILL signals results in same local Distributed7 software shutdown as that of *apmd*.

Note that since *netd* is responsible for setting up and maintaining Distributed7 TCP/IP connectivity, termination of the *netd* daemon process automatically causes loss of connection to all remote hosts.

6.3.1.4 Treatment by spmd, alarmd, dsmd

SIGTERM

The *spmd*, *alarmd*, and *dsmd* daemon processes interpret the SIGTERM signal as a graceful request to terminate. On receipt of the SIGTERM signal, they terminate with an exit code of 0. Since this is a graceful termination, *apmd* does not respawn these processes.

SIGKILL

The *spmd*, *alarmd*, and *dsmd* daemon processes cannot interpret SIGKILL. When SIGKILL is sent to these daemon processes, they terminate prematurely. Termination is detected by *apmd*, which automatically re-spawns them because of a FAILSAFE process defined in the *apmconfig* file.

- Termination of *spmd* automatically causes dismantling of the SS7 board connectivity. The *spmd* daemon process is responsible for initialization and maintenance of non-SS7 specific kernel-resident data used by Distributed7, and maintenance of SS7 signalling boards on the local host.
- Termination of the *alarmd* daemon process causes all local alarm conditions to be displayed on the system console in the form of raw alarms instead of being logged in the alarm log files.
- Termination of the *dsmd* daemon process causes the DSM framework on the local host to become inoperative, and subsequent failures occur in all DSM API library calls that are initiated by user-space threads on the local host.

6.3.1.5 Treatment by dkmd

SIGTERM

dkmd interprets the SIGTERM signal as a graceful request to terminate. On receipt of the SIGTERM signal, *dkmd* terminates with an exit code of 0. *apmd* does not respawn *dkmd*.

Termination of the *dkmd* daemon process causes DKM/DRA frameworks on the local host to become inoperative, and subsequent failures occur in all DKM/DRA API library calls that are initiated by kernel-space threads on the local host. Since the successful operation of the MTP-L3, SCCP, and TCAP layers under the Distributed7 platform relies on the constant availability of DKM/DRA frameworks, these layers terminate automatically when the *dkmd* daemon process on the local host terminates.

SIGKILL

The SIGKILL signal cannot kill the *dkmd* daemon process because the *dkmd* daemon process has user-space threads intentionally stuck in the kernel. If a SIGKILL signal is sent to *dkmd*, the result is only loss of the APM heartbeat. This is because the *dkmd* daemon process is in no state to respond to a heartbeat request from *apmd*. As a result, the *dkmd* entry is marked in the *apm_ps* output as "heartbeat failed" within 60 seconds time—*apmd* initiates heartbeat request messages every 60 seconds, according to the rules in the *apmconfig* file. A "heartbeat failed" message is also printed in the MLog file. In other words, DKM/DRA frameworks remain operational even after a SIGKILL signal is sent to *dkmd*.

Note that after sending a SIGKILL signal to *dkmd*, when trying to shutdown Distributed7 software, all Distributed7 multiplexors—with the exception of the DKM multiplexor—dismantle properly. The corresponding daemons terminate normally. It takes about 90 seconds, however, for the DKM multiplexor to dismantle itself, and for all *dkmd* user-space threads whose execution were stuck in the kernel to exit. Therefore, if a SIGKILL signal is sent to *dkmd*, the *dkmd* entry in the *ebs_ps* output does disappear until the 90 second timeout occurs.

6.3.1.6 Treatment by tcmd

SIGTERM

When no TC applications are being used, *tcmd* interprets the SIGTERM signal as a graceful request to terminate. On receipt of the SIGTERM signal, *tcmd* terminates with an exit code of 0.

SIGKILL

tcmd cannot interpret the SIGKILL signal. When SIGKILL is sent to *tcmd*, the result is premature termination of the *tcmd* daemon process with a non-zero exit code.

When TC applications are used, the *tcmd* daemon process cannot be killed by sending SIGTERM or SIGKILL signals. This is because the *tcmd* daemon process has user-space threads intentionally stuck in the kernel. Attempts to send SIGTERM or SIGKILL signals to *tcmd* when TC applications are in use do not harm existing TC applications. However, all subsequent TCAP level registration attempts return timeout failures.

6.4 Using MMI/MML

Configuration and administration of the Distributed7 system can be done through a Man-Machine Language Terminal Handler (MMI/MML). An MMI/MML session is started by entering the following command at the command line: **`mmi`** or **`mml sp`**
where *sp* is the logical signalling point number of the system, such as 0. The command is described in Section 7.2.13 on page 7-26.

MMI/MML commands are used for configuring the system as an SS7 node and for general system management, such as managing the display and configuration of the alarm process. All MMI/MML commands are defined in Chapter 9: Man-Machine Language Commands. The conventions, rules, and general output information for MMI/MML are also provided in that chapter.

On-line help is available for all MMI/MML commands through the HELP MMI/MML command (see Section 6.4.1).

An MMI/MML session is ended with the EXIT MMI/MML command (see Section 9.7.9 on page 9-146).

6.4.1 Help Command

MMI/MML command descriptions and syntax can be seen by executing the HELP command.

```
HELP:;
```

When you run this command, the list of executable MMI/MML commands is displayed. Only the MMI/MML commands that are executable will be listed. After executing the HELP command, the HELP prompt will appear and the command for which help is desired should be entered at the prompt. For example:

```
HELP_MML_CMD>MODIFY-SP:;
```

MMI/MML displays the help text about the MODIFY-SP (page 9-74) operation on the screen. You can also see the same help text by including the desired command as a parameter in the help command:

```
HELP:CMD=MODIFY-SP:;
```

6.4.2 Filtering Display Command Output

The results of DISPLAY commands are tabular in MMI/MML. All the parameters of a particular MO are displayed side by side. If the resulting output cannot fit the screen, then the output is displayed page by page, allowing the user to manually page through it.

The normal output of a display command can be filtered to only display certain parameters out of the ones available. To use this feature of MMI/MML, the desired parameters should be added to the display command as arguments with a value of **?**. The parameters should be listed after the required key parameters that are defined in the command listing. For example,

```
DISPLAY-ROUTE:RTSET=RS1,LSET=LS1,LSSTATE=?;
```


In this command, RTSET and LSET are the required key parameters. By adding the argument, 'LSSTATE=?', the output will only show the linkset state and key parameters for the route instead of all the statuses. More than one parameter could have been specified in this manner.

Key parameters must always have a value specified. If the ***** wildcard is valid for the key parameter, then **?** can also be used. For example,

```
DISPLAY-ROUTE:RTSET=RS1,LSET=?;
```

This version of the command will display the RTSET and LSET parameters of all linksets in the specified route set. Key parameters that do not accept *****, such as RTSET for this command, will not accept **?** as a value. Refer to the DISPLAY commands for key parameters.

6.4.3 Logging MMI/MML Commands

MMI/MML logs all MMI/MML commands, except DISPLAY commands, that are entered by an operator into an ASCII file. The file is ***$EBSHOME/access/RUN*/backup/MMLcmnds.current***. Each log of a command also contains the login name and ID of the user who entered the command along with the date and time at which it was entered.

Reviewing this file can help to verify the commands and values that were entered. This logging operation can be turned on and off with the MODIFY-MMLCONF command, which is described in Section 9.7.6 on page 9-140.

6.4.4 Changing the MMI/MML Time-Out

After an MMI/MML command is entered, MMI/MML waits up to 15 seconds for a response from the system. If no response is received, it displays an SPM time-out error message. The time-out value is configurable through the timeout parameter of the MODIFY-MMLCONF command (Section 9.7.6). The time-out value can be in the range from 0 to 65000 milliseconds.

6.4.5 User-Defined Commands

Since users can create their own managed objects, the operations (add, delete, display, or modify) that are defined on them can be introduced to MMI/MML to allow configuration of the managed objects with the MMI/MML. HELP information on these new MMI/MMLs can also be included. This topic is covered in the *Application Development Manual*.

6.4.6 History Facility

The mml history facility has been removed. The history functionality can be achieved through various third party shell and command interface software. A way of doing so, through the emacs editor, is explained below.

6.4.6.1 Using MML through Emacs Shell/Telnet Buffers

- Run emacs
- If D7 is installed on the same machine as emacs, activate an emacs shell (<alt-x> shell command)
- If D7 is installed on another machine, activate an emacs telnet session (<alt-x> telnet command)
- Run mml from inside the shell/telnet buffer
- Use emacs shell/telnet history keys to get/execute mml command history

Usage of the following four emacs history keys should be enough for most of the command history requirements.

- <ctrl-p> or <ctrl - up arrow> previous command
- <ctrl-n> or <ctrl - down arrow > next command
- <ctrl-r> previous command matching input
- <ctrl-c ctrl-l> list command history

The user can use the emacs rename-buffer command to rename telnet/shell sessions and create multiple mml sessions with different names. The emacs shell/telnet buffers are also normal editor buffers, which means all the other emacs commands for search, etc. are applicable to these buffers as well.

6.4.6.2 Key Translations

What follows is a subset of emacs commands and key bindings that can be used from emacs shell/telnet buffers for mml configuration purposes. Note that the ALT and ESC prefixes are equivalent.

Table 6-1: Key Translations

Key	Binding
<C-down>	next-input
<C-up>	previous-input
ESC s	next-matching-input
ESC r	previous-matching-input
ESC n	next-input
ESC p	previous-input
C-c C-b	shell-backward-command
C-c C-f	shell-forward-command
C-c C-p	previous-prompt
C-c C-n	next-prompt
C-c C-l	list-history
C-c C-\	quit-subjob
C-c C-z	stop-subjob
C-c C-c	interrupt-subjob
C-c C-w	backward-kill-word

Table 6-1: Key Translations (Continued)

Key	Binding
C-c C-u	kill-input
C-c C-a	bol-or-process-mark
C-c C-x	get-next-from-history
C-c SPC	accumulate
C-a	beginning-of-line
C-b	backward-char
C-e	end-of-line
C-f	forward-char
C-k	kill-line
C-n	next-line
C-o	open-line
C-p	previous-line
C-q	quoted-insert
C-r	search-backward
C-s	search-forward
C-v	scroll-up
C-w	kill-region
C-y	yank
<end>	end-of-buffer
<next>	scroll-up
<prior>	scroll-down
<down>	next-line
<right>	forward-char
<up>	previous-line
<left>	backward-char
<home>	beginning-of-buffer
<find>	search-forward
C-x <	scroll-left
C-x >	scroll-right
C-x [	backward-page
C-x]	forward-page
ESC C-r	isearch-backward-regexp
ESC C-s	isearch-forward-regexp
ESC <	beginning-of-buffer
ESC =	count-lines-region
ESC >	end-of-buffer
ESC d	kill-word
ESC e	forward-sentence
ESC f	forward-word

Table 6-1: Key Translations (Continued)

Key	Binding
ESC k	kill-sentence
ESC v	scroll-down

6.5 Using SNMP

6.5.1 Overview

This section provides general information on SNMP, the Distributed7 SNMP agent, configuring the agent, and using the agent. The information in *Chapter 3: Concepts* should be reviewed before reading this chapter.

The Distributed7 SNMP agent supports both SNMPv1 and SNMPv2 protocols. Before using the SNMP agent, the configuration files must be updated.

6.5.2 SNMP Background

Network Management is a requirement for controlling and managing network elements. Simple Network Management Protocol (SNMP) is currently the most popular network management protocol used to remotely control and monitor heterogeneous network elements.

A network management system may contain many network elements, each with a processing entity, termed an agent. Network elements are devices such as hosts, routers, terminal servers, etc. Normally, one centralized location, called the network management station, monitors and controls the other network elements by accessing their network management information. Each agent has access to management instrumentation, at least one management station, and a management protocol that is used to convey management information between the agents and management stations. The network elements are monitored and controlled by accessing their management information.

Management information is a collection of managed objects, residing in a virtual information store, called a Management Information Base (MIB). Collections of related objects are defined in MIB modules. These modules are defined by the Structure of Management Information (SMI), which is a subset of OSI's Abstract Syntax Notation One (ASN.1).

SNMP was designed to be an application-level protocol that is part of the TCP/IP protocol suite. It is intended to operate over the connectionless User Datagram Protocol (UDP). No ongoing connections are maintained between a management station and its agents. Instead, each message exchange is a separate transaction between a management station and an agent. For a stand-alone management station, a manager process controls access to the stations's central MIB and provides an interface to the network manager. SNMP can be thought of as a query language on the MIB tree. Figure 6-1 depicts the SNMP architecture.

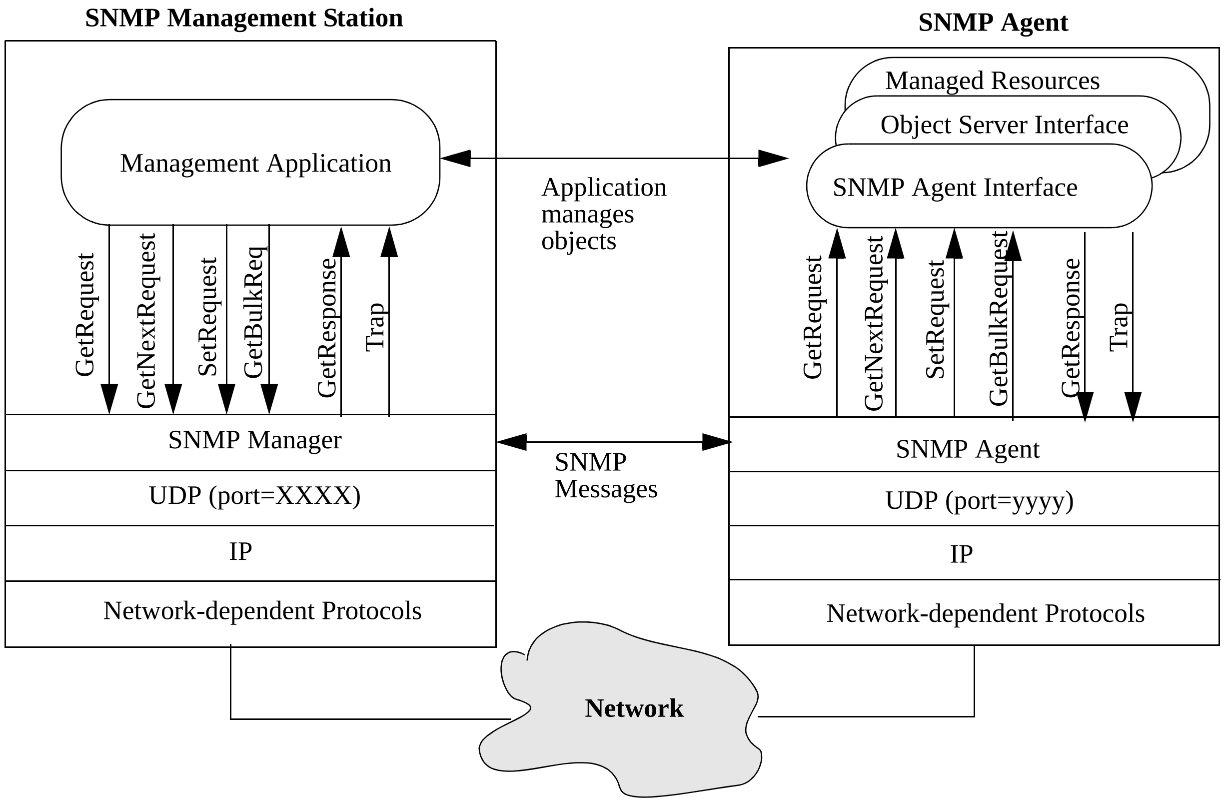


Figure 6-1: SNMP Architecture

6.5.3 Distributed7 MIB

In reality, information is stored at a device as a combination of switch settings and hardware counters. The information can be stored in files or in memory, as tables or variables. In the SNMP standards, this logical database of network management information is called a Management Information Base (MIB). The physical form of this data is less important than the accessibility to this data. SMI defines the scheme for the MIB as a tree structure.

Figure 6-2 shows part of the Internet MIB subtree, highlighting the NewNet Communication Technologies, LLC private branch. Distributed7's managed objects are defined by the *accessMANAGER* subtree under the *NewNet* branch (*.1.4.1.newnet.accessMANAGER*). Each instance of any object type defined in the MIB is uniquely identified in SNMP operations by a unique OBJECT IDENTIFIER of the form 'x.y' where x is the name of an non-aggregate object type defined in the MIB and y is an OBJECT IDENTIFIER fragment that identifies the desired instance. For example, if the network manager (management station) wants to get the information about **l3tValue** in the **l3timerEntry**, it sends the *.accessMANAGER.signalling.ss7.1.2.1.1.0 (*'0' is the value of the key l3tTimer for the first conceptual row) in a SNMP *GetReq PDU*. In response, it will get *.accessMANAGER.signalling.ss7.1.2.1.2.0 = 160 ms*, the value of the l3tValue in the first conceptual row (see the MTP MIB view in Figure 6-3).

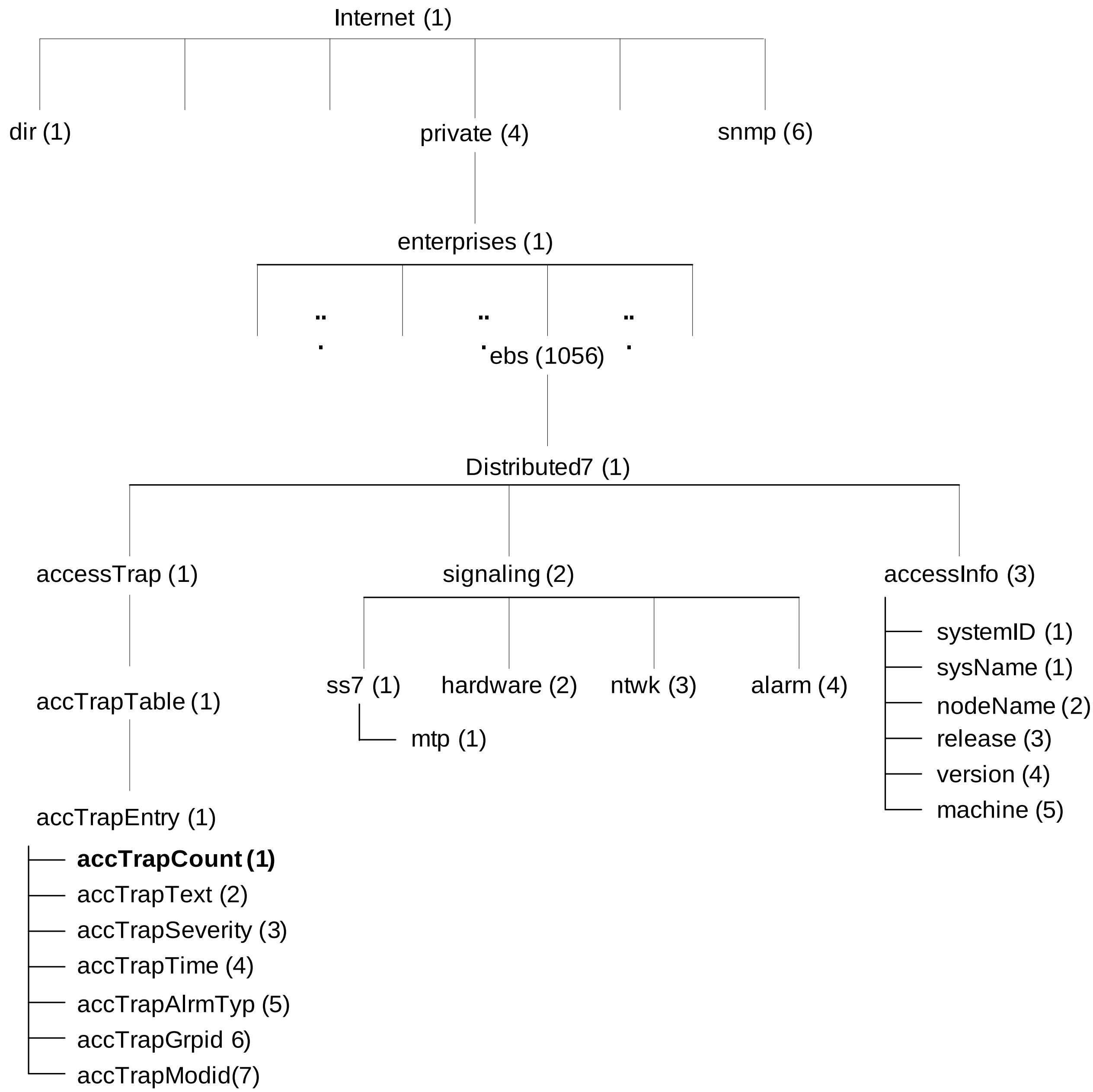


Figure 6-2: Internet MIB View

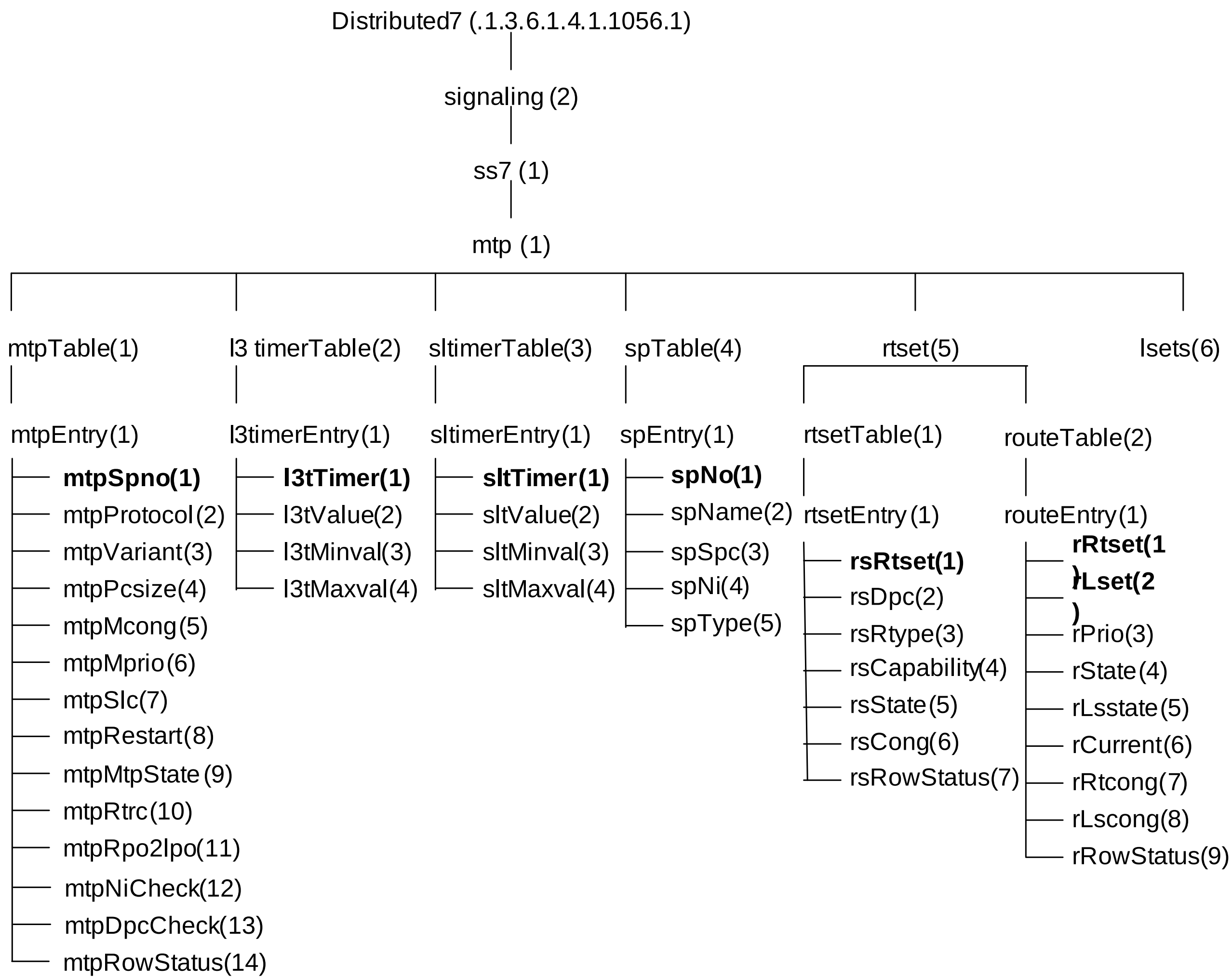


Figure 6-3: Distributed7 MIB View—MTP Layers

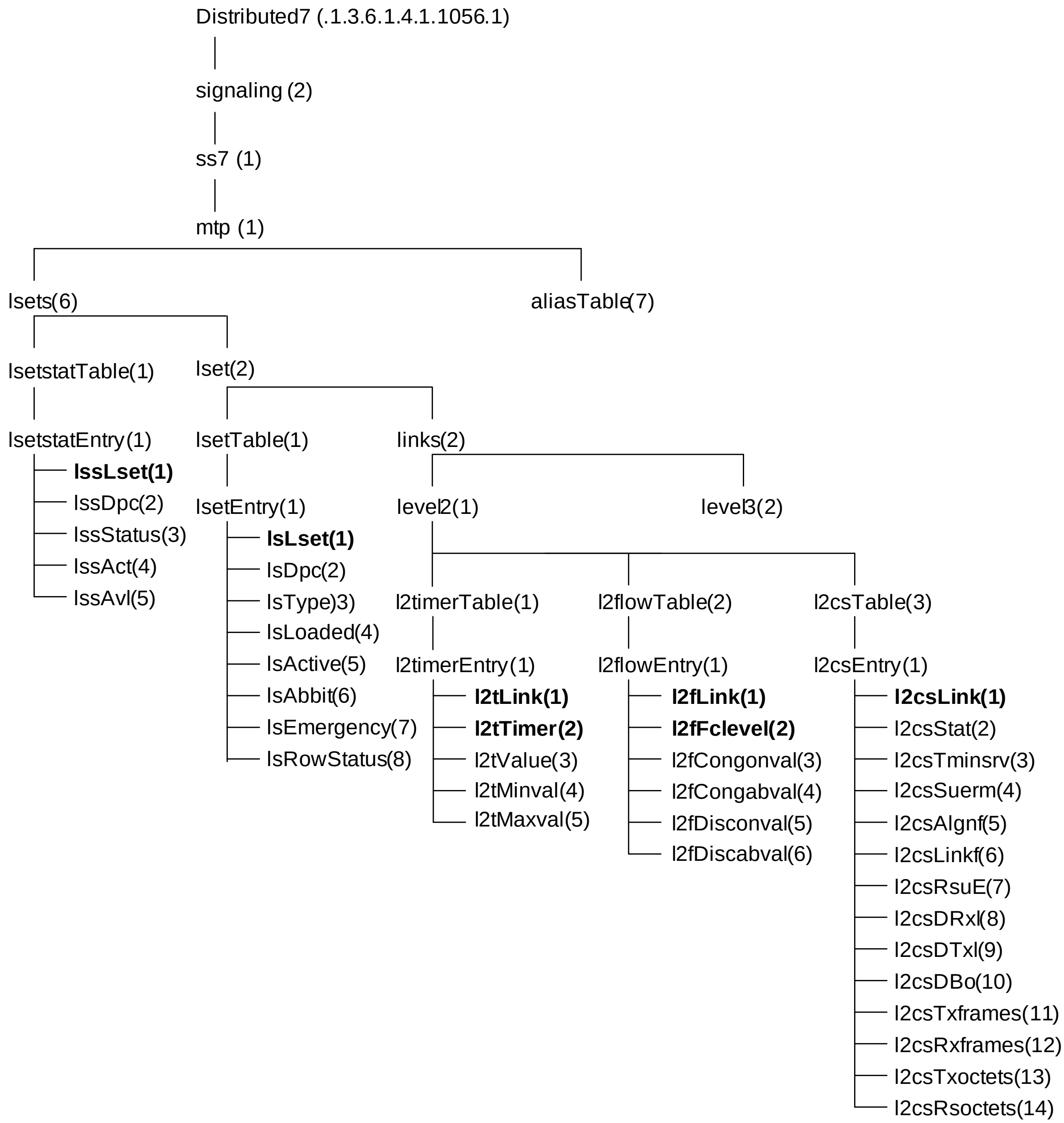


Figure 6-4: Distributed7 MIB View—MTP Layers, continued

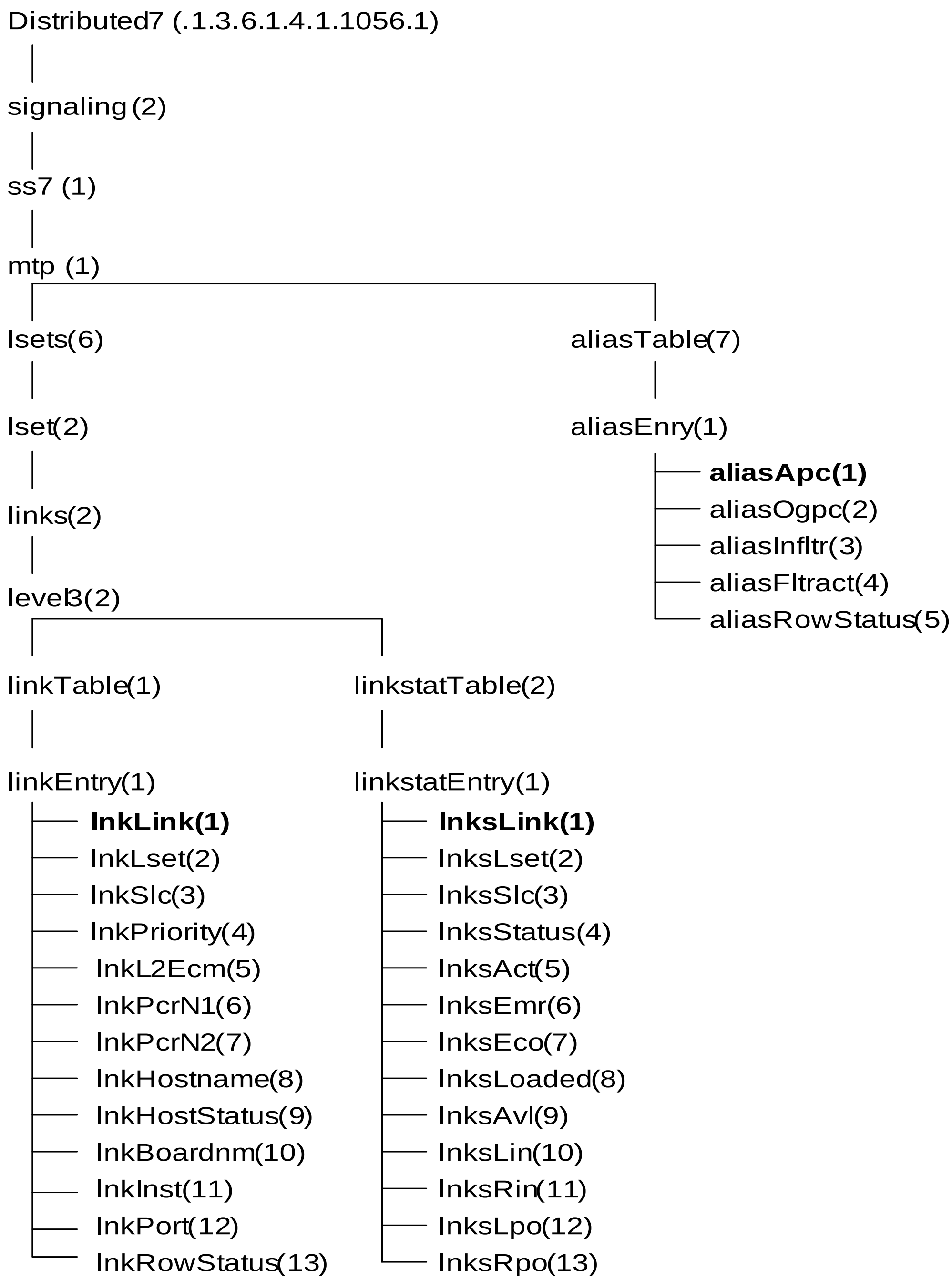


Figure 6-5: Distributed7 MIB View—More MTP Layers

For the *NewNet* subtree, some important design points exist. The SNMP SMI does not permit nesting. For example, an element of a table cannot be defined to be another table. However, conceptual table rows can be defined. For example, the link is a conceptual table row under *linkEntry* in Figure 6-3. Each conceptual table row has a **status** column (not shown in the diagrams) which uses the **RowStatus** textual convention of SNMPv2. The **RowStatus** textual convention manages the creation and deletion of conceptual rows. The **status** column has six defined values:

- *active*: indicates that the conceptual row is available for use by the managed device. This state value may be read and written. Supported in this version of Distributed7.
- *notInService*: indicates that the conceptual row exists in the agent, but is unavailable for use by the managed device. This state value may be read and written. Not supported in this version of Distributed7.
- *notReady*: indicates that the conceptual row exists in the agent, but is missing information that is necessary for it to be available for use by the managed device. This state value may be read, but not written. Not supported in this version of Distributed7.
- *createAndGo*: sent by a management station to create a new instance of a conceptual row and make it available for use by the managed device. This action value may be written, but never read. Supported in this version of Distributed7.
- *createAndWait*: sent by a management station to create a new instance of a conceptual row, but not have it available for use by the managed device. This action value may be written, but never read. Not supported in this version of Distributed7.
- *destroy*: sent by a management station to delete all instances associated with an existing conceptual row. This action value may be written but never read. Supported in this version of Distributed7.

The MIB for SCCP is diagrammed in Figure 6-6 and Figure 6-7.

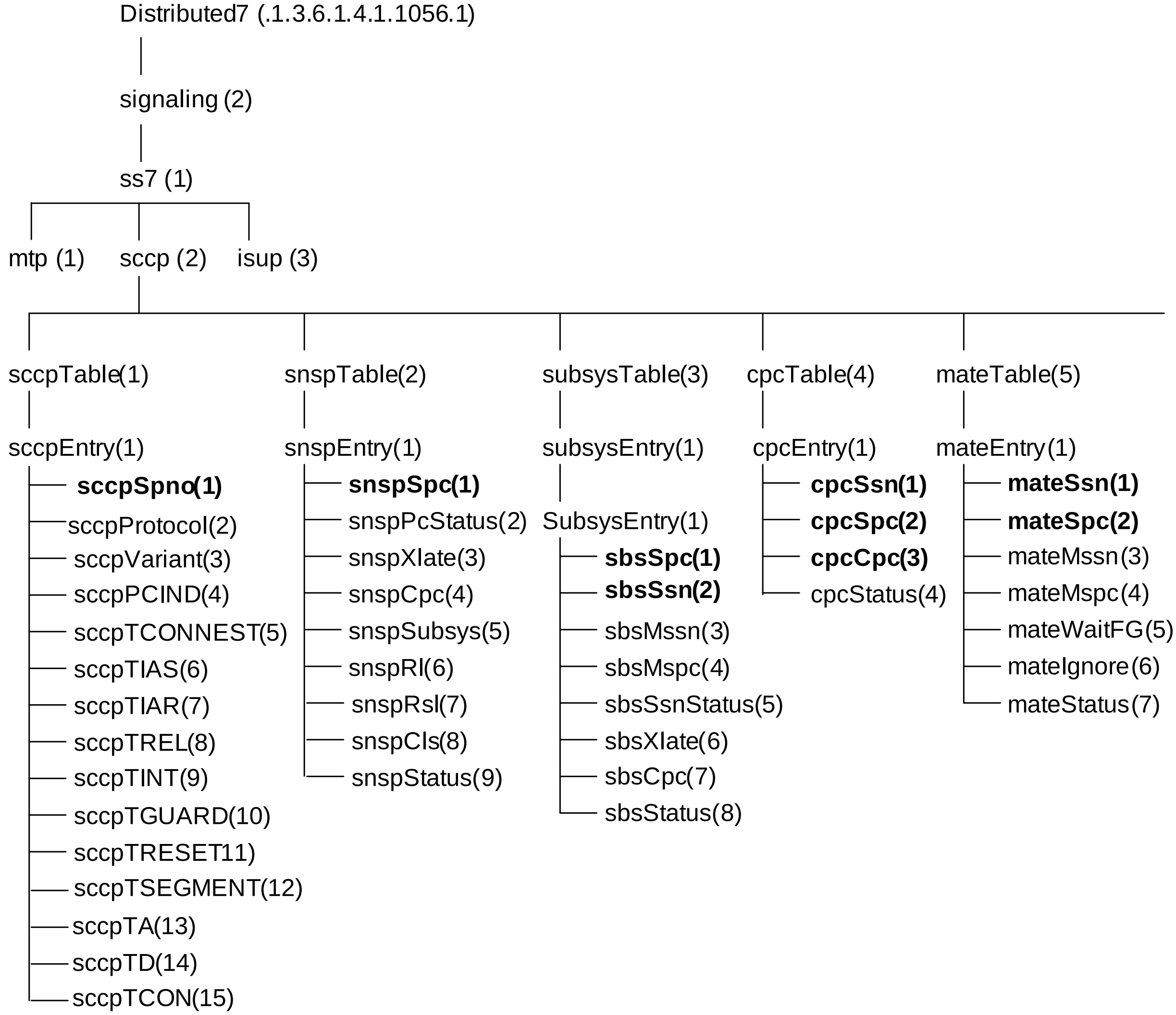


Figure 6-6: Distributed7 MIB—SCCP Layer

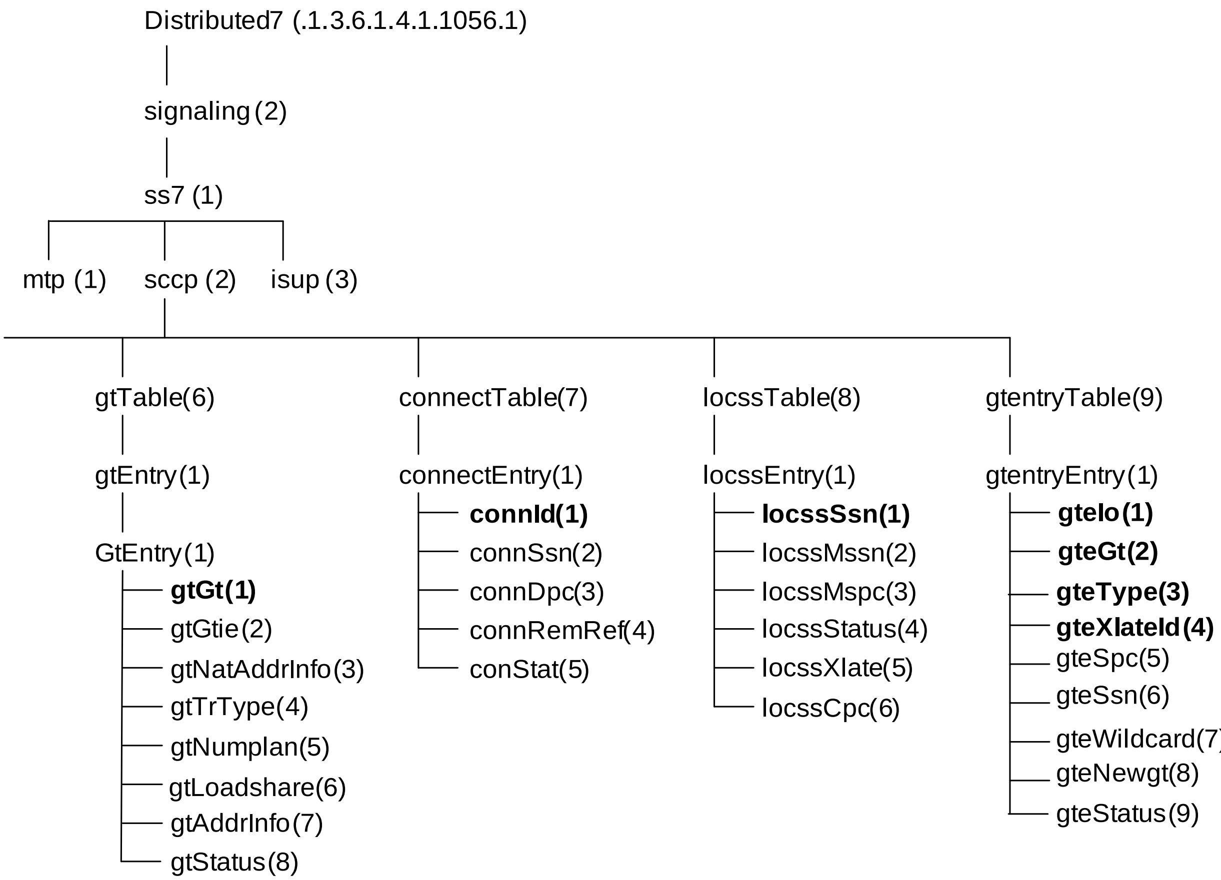


Figure 6-7: Distributed7 MIB—SCCP Layer, continued

The MIB for ISUP is diagrammed in Figure 6-8 and Figure 6-9.

- Distributed7 (.1.3.6.1.4.1.1056.1)
  - signaling (2)
    - ss7 (1)
      - isup (3)
        - isupTable(1)
          - isupEntry(1)
            - **isupName(1)**
            - isupVariant(2)
            - isupMntcInd(3)
            - isupConges(4)
            - isupXlate(5)
            - isupRecMod(6)
            - isupMbgind(7)
            - isupAutorsp(8)
            - isupAggrind(9)
            - isupExchodc(10)
            - isupUpmind(11)
        - isupnodeTable(2)
          - isupnodeEntry(1)
            - **isupnodePcno(1)**
            - isupnodeDpc(2)
            - isupnodeCongestion(3)
            - isupnodeAccess(4)
            - isupnodeAnmoff(5)
            - isupnodeAcmoff(6)
            - isupnodeCrgoff(7)
            - isupnodeCicControl(8)
            - isupnodeFirstCic(9)
            - isupnodeMaxCct(10)
            - isupnodeLocation(11)
            - isupnodeRowStatus(12)
        - isupcgrpTable(3)
          - isupcgrpEntry(1)
            - **isupcgrpPcno(1)**
            - isupcgrpDpc(2)
            - **isupcgrpGrpId(3)**
            - isupcgrpCctnum(4)
            - isupcgrpTrnkGId(5)
            - isupcgrpSCGA(6)
            - isupcgrpCCName(7)
            - isupcgrpMntcName(8)
            - isupcgrpStatus(9)
        - isupcctTable(4)
          - isupcctEntry(1)
            - **isupcctPcno(1)**
            - isupcctDpc(2)
            - **isupcctGrpid (3)**
            - **isupcctCctnum(4)**
            - isupcctRange(5)
            - isupcctStatus(6)
            - isupcctMtcStatus (7)
            - isupcctHwdStatus(8)
            - isupcctSusStatus(9)
            - isupcctOperStatus (10)
            - isupcctCic(11)
            - isupcctRowStatus(12)

Figure 6-8: Distributed7 MIB—ISUP Layer

Add isupcctCic(11) before isupcctRowStatus(11) to isupcctTable(4)

And change isupcctRowStatus(11) to isupcctRowStatus(12)

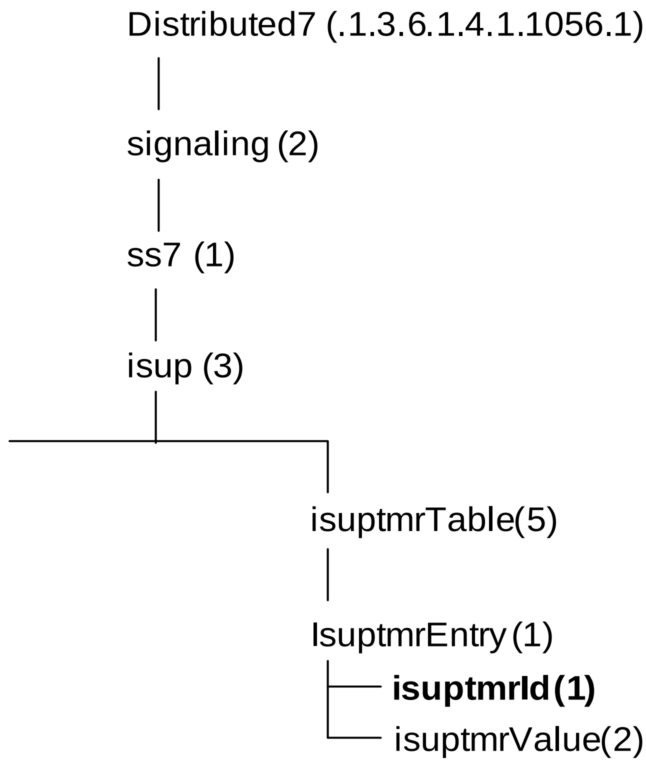


Figure 6-9: Distributed7 MIB—ISUP Layer, continued

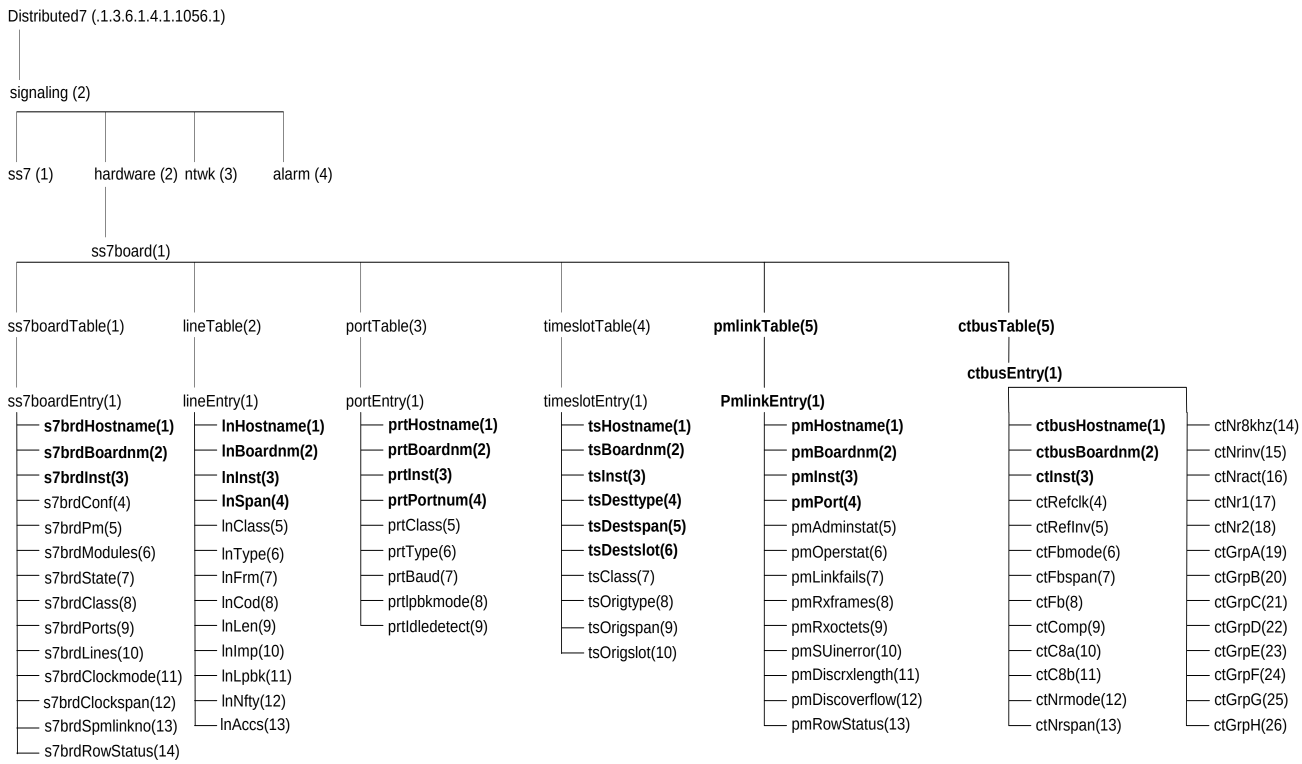


Figure 6-10: Distributed7 MIB—HARDWARE Layer

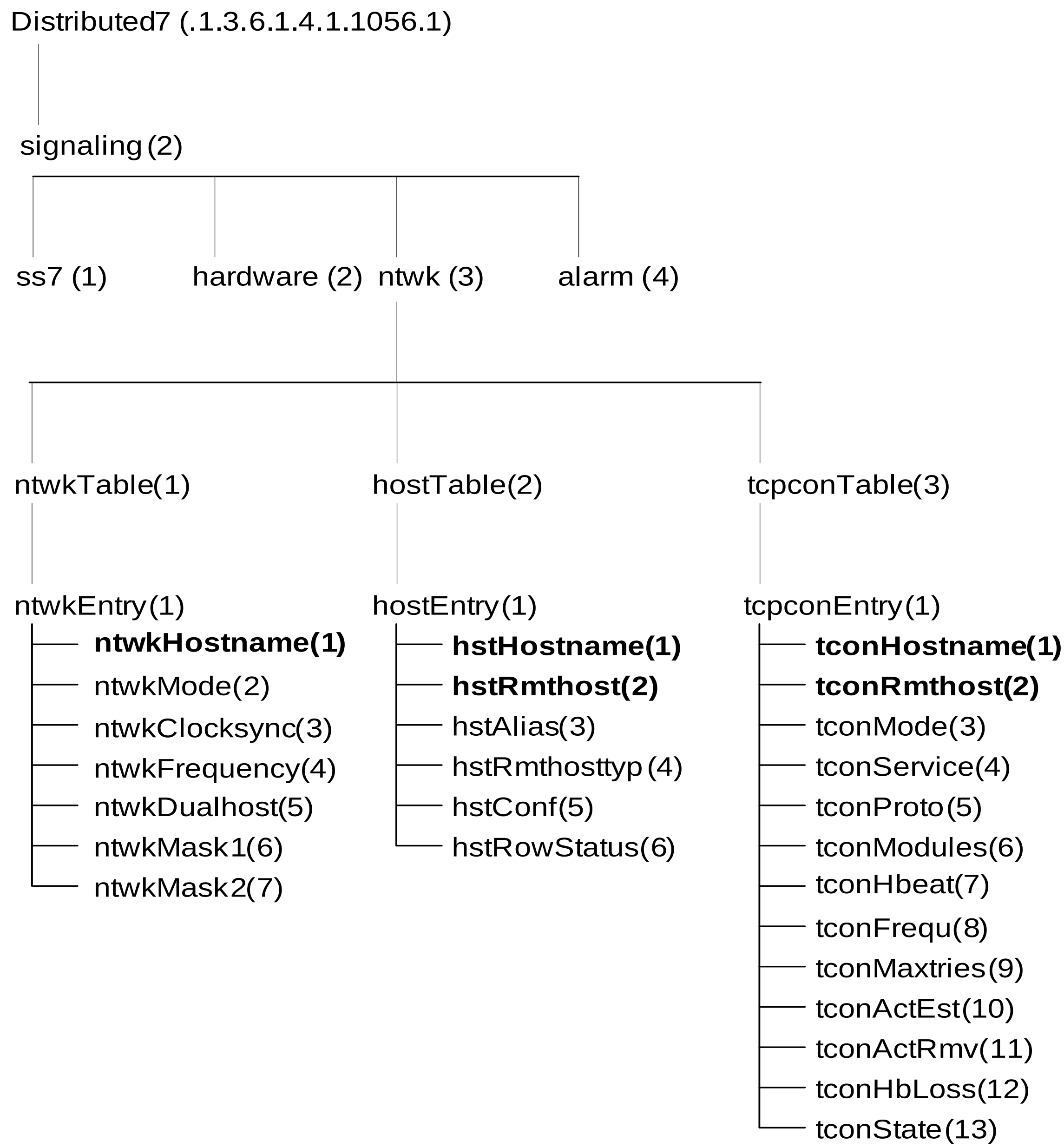


Figure 6-11: Distributed7 MIB—NTWK Layer

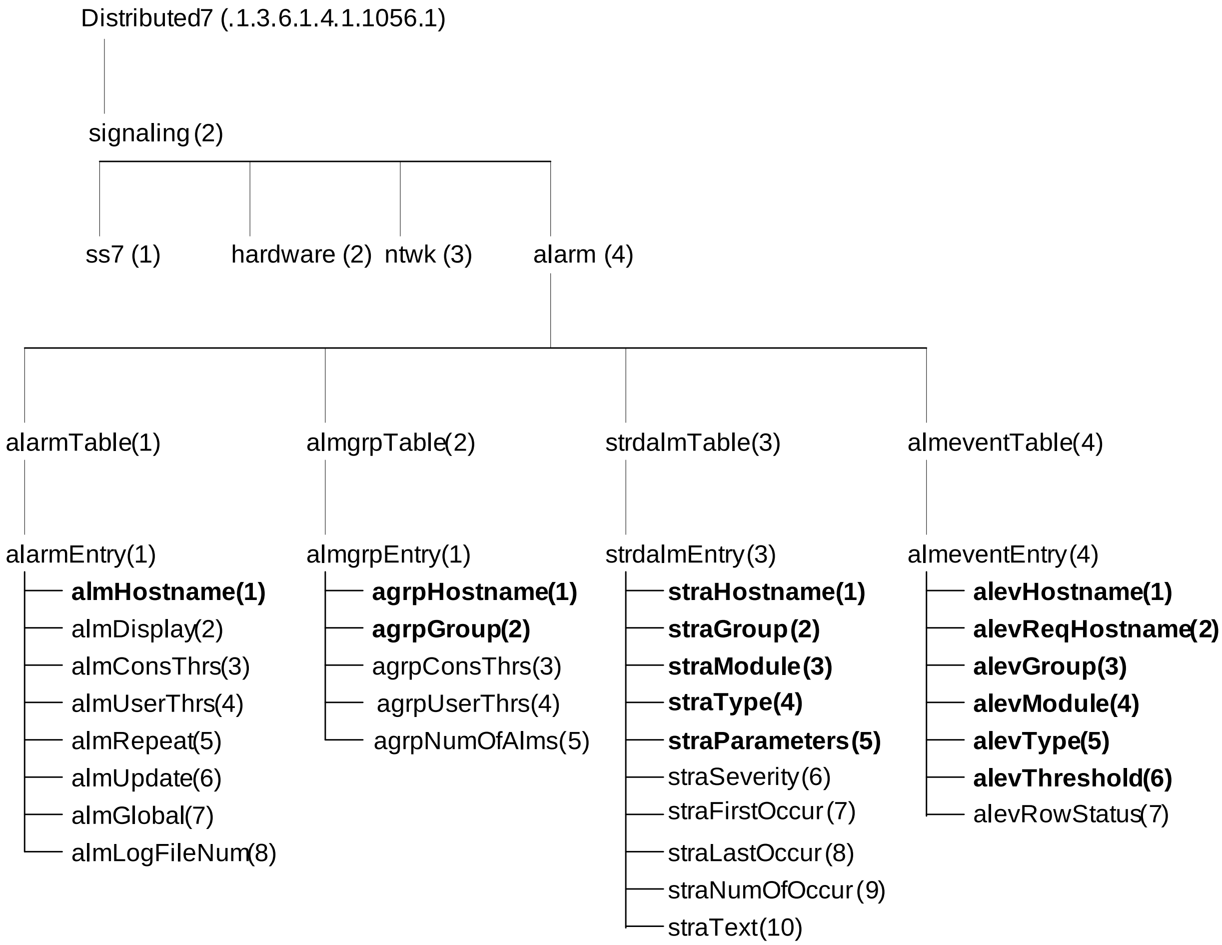


Figure 6-12: Distributed7 MIB—ALARM Layer

6.5.4 Configuration

The Distributed7 software supports control of its managed objects via SNMP through the *AccessSNMP* facility. This SNMP agent supports both SNMPv1 and SNMPv2 protocols. Before using the SNMP agent, configuration files must be updated.

The ***mib_text.v1, mib_text.v2*** and ***snmp_cmnd.tbl*** files are automatically installed upon installation of Distributed7 software. These files are located in the directory ***$EBSHOME/access/RUN/config/SNMP***.

To configure the Distributed7 SNMP agent, ****.ini*** configuration files in the ***$EBSHOME/access/RUN/config/SNMP/etc*** directory must be edited and copied to the ***$EBSHOME/access/RUN***<sp#>***/config/SNMP*** directory with a ****.conf*** extension. These files are SNMPv1 configuration files (***trap.conf***, ***community.conf***) and SNMPv2 configuration files (***party.conf***, ***context.conf***, ***view.conf***, ***acl.conf***). To use both SNMPv1 and SNMPv2, all of these files should be copied. If only one version will be used, only copy the set of files for that version.

After the configuration is completed by following the instructions below, the SNMP agent can be started with the AccessSNMP command. The command is described in *Section 7.2.4 on page 7-8*.

6.5.4.1 SNMPv1 Configuration Files

Two SNMPv1 configuration files exist. For SNMPv1, security is only at the community information level. Each file should be edited as described below.

community.conf

Contains the port numbers for listening SNMPv1 requests. Up to 4 ports can be defined for listening SNMPv1 requests on the first line. The community names to be accepted by the SNMP agent are defined in the second line. After initialization, port number ‘7778’ and community name ‘public’ are defined for your system by default. Both lines can be modified. Sample lines from the file appear below:

```
#local-port-nums for listening snmpV1 managers (maximum 4 ports)
#defined community-names for identifing snmpV1 managers (maximum 4 communities)
7778 yyy zzz
public
```

trap.conf

Contains the community information, network manager IP address, and port number to which the SNMPv1 traps will be sent. Sample lines from the file appear below:

```
#community name     remote manager net address remote port number
public              yyy.yyy.yyy.yyy  xxx
```


6.5.4.2 SNMPv2 Configuration Files

The four SNMPv2 configuration files incorporate three security concepts - party concept, context concept and access policy concept. Each file should be edited as described.

party.conf

Contains party concept elements. The required elements are party friendly name, party object identifier, domain name, IP address, and port number. The file must be edited to replace xxx.xxx.xxx.xxx with the network address of the agent station and yyy.yyy.yyy.yyy with the network address of the management station. Port numbers are set to defaults, but they can be modified. If the chosen party is not using authentication, then there is no need to configure any other elements. If the chosen party is using *SNMPv2 md5* authentication, then **snmpv2md5Auth** must be placed in *authProtocol* and the *authPrivate* text string key has to be verified to be the same as the manager party *authPrivate* key. Sample lines from the file follow:

```
# FriendlyName PartyID
# TDomain      IP-address     UDP-port
# authProtocol   privProtocol
# lifetime maxmessagesize
# clock
# authPrivate authPublic
# privPrivate privPublic

agent_accessMANAGER          .1.3.6.1.6.3.3.1.3.xxx.xxx.xxx.xxx.1
snmpUDPDomain                xxx.xxx.xxx.xxx      7777
noAuth                       noPriv
300                          484
00000000
00000000000000000000000000000000      Null
00000000000000000000000000000000      Null

manager_accessMANAGER        .1.3.6.1.6.3.3.1.3.xxx.xxx.xxx.xxx.2
snmpUDPDomain                yyy.yyy.yyy.yyy      7788
noAuth                       noPriv
300                          484
00000000
00000000000000000000000000000000      Null
00000000000000000000000000000000      Null
```

context.conf

Defines the contexts between agent and manager parties. In this file, xxx.xxx.xxx.xxx refers to the network address of the agent station. No other information needs editing. Sample lines from the file appear below:

```
# contextName   contextIdentity
# contextViewIndex contextLocalEntity contextLocalTime
# contextDstPartyIndex  contextSrcPartyIndex    contextProxyContext

context_1_accessMANAGER       .1.3.6.1.6.3.3.1.4.xxx.xxx.xxx.xxx.1
1                      Null        currentTime
2              1                         .0.0

context_2_accessMANAGER       .1.3.6.1.6.3.3.1.4.xxx.xxx.xxx.xxx.2
1                      Null        currentTime
4              3                         .0.0
```

view.conf

Defines the supported MIB view. The Distributed7 SNMP agent supports only the *accessManager (.1.2.6.1.4.1.1056.2)* subtree.

acl.conf

Defines the access privileges for each source party, destination party, and context triple. If the default agent manager party definitions are used, then there is no need to edit this file. Whenever new parties are defined, the required privilege information must be added to this file. Sample lines from the file appear below:

```
# targetParty sourceParty context privileges
# and privileges is [gnsrt]*
# where g = get, n = getnext, s = set, r = get Response,
# b = bulk, i = inform, u = trap2

agent_accessMANAGER manager_accessMANAGER context_1_accessMANAGER gnb

manager_accessMANAGER agent_accessMANAGER context_1_accessMANAGER ru
```

6.5.4.3 Defining Parties

When using the parties that are created by default, only the IP addresses of agent and management systems parties need to be edited. However, when new parties are created, the following steps should be followed:

1. Create new agent and management parties in ***party.conf*** file. Verify authentication keys when using authentication. (Distributed7 SNMP does not support privacy (*noPriv*)).
2. Define a context in ***context.conf*** for the party pair with party indexes. The same view number will be used for this context.
3. Define access privileges in ***acl.conf*** for the new party pair.

6.5.5 Using the SNMP Agent

This section describes the functions that can be performed by the SNMP agent.

6.5.5.1 Platform Management with SNMP Agent

An SNMP agent communicates with managed objects through the Object Server. The Object Server acts as a name server that provides binding to the right managed object depending on the request type issued by the SNMP agent.

The types of commands used by system tasks are ADD, DELETE, MODIFY, and GET. Each request is assigned a unique transaction identifier which is used to correlate the responses to the requests. For additional detail, the GET command is further divided into the subtypes GET-FIRST and GET-NEXT.

The parameters of a managed object may be acted upon locally or remotely. The actions are coarsely defined as GET and SET operations. Each managed object is uniquely addressed by its name.

The Distributed7 environment is managed by an MMI/MML agent. Operations on the managed objects can be invoked through this application. An example of MMI/MML-to-managed object interaction is given during an execution of a command, as shown in Figure 6-13. The command is issued by the user and accepted by MMI/MML. MMI/MML sends a PDU containing the managed object name, operation, and parameter value using the CNFG library. The Managed Object Server performs the operation. MMI/MML will also receive a response PDU from the MO Server. The same request can be issued from a management station using SNMP.

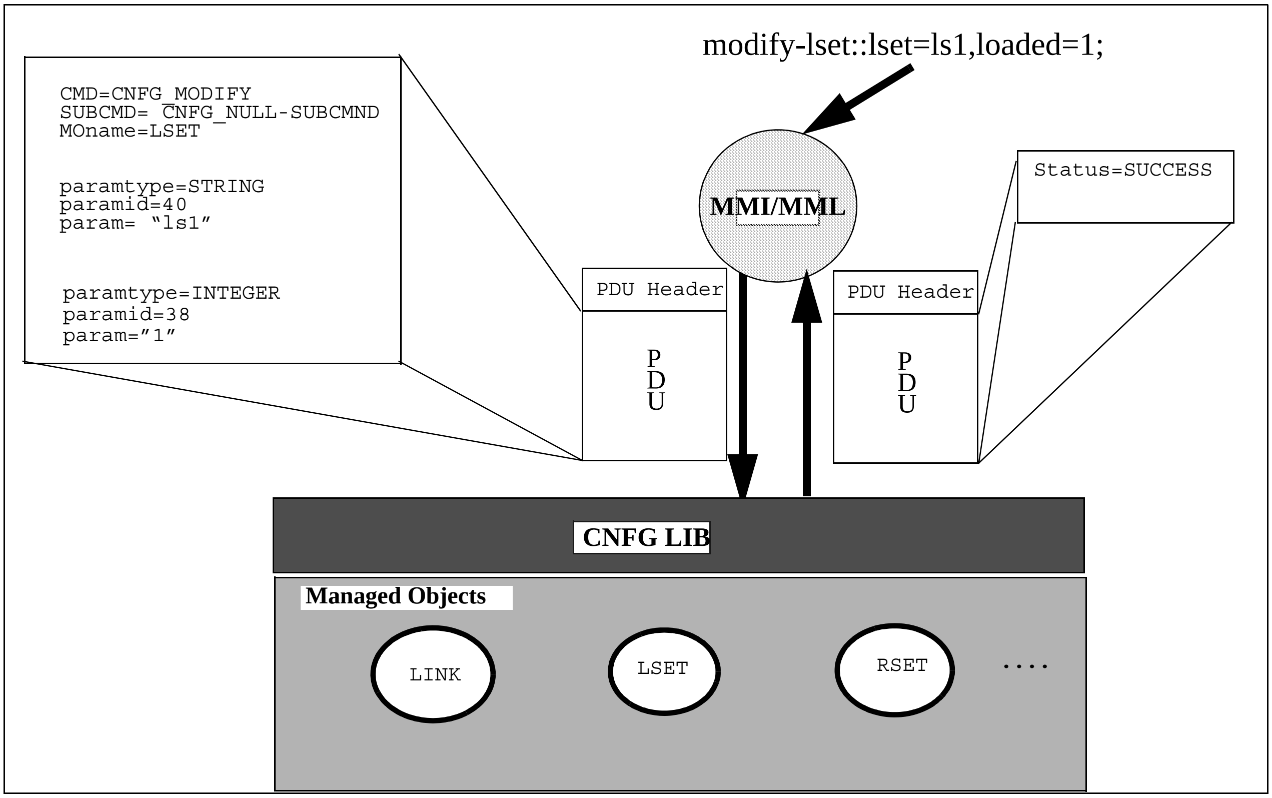


Figure 6-13: Typical MMI/MML-to-Managed Object Interaction

In Figure 6-14, the same example that was presented with MMI/MML in Figure 6-13 is shown with an SNMP agent. The SNMP agent receives an SNMP PDU, containing the object identifier and value for the desired managed object, from a management station. Then, the SNMP agent maps the information to the managed object containment structure and sends a PDU using the CNFG library, just as MMI/MML did in Figure 6-13.

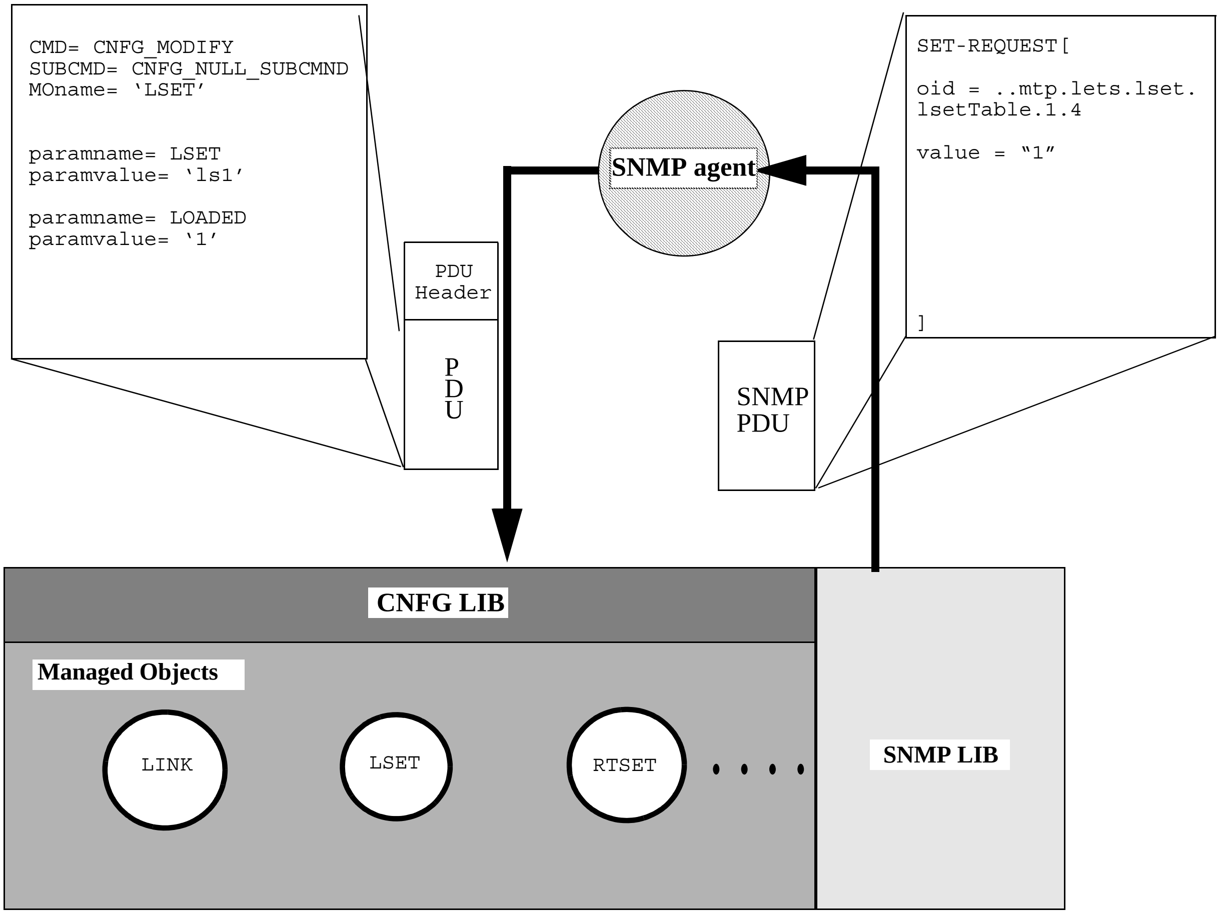


Figure 6-14: Management Station-Agent Interaction

6.5.5.2 Alarm Reporting with SNMP Agent (Traps)

Traps are notification mechanisms for extraordinary events which convey critical information. With SNMP, exception handling is carried over with a trap-directed polling scenario. Each time an unusual condition exceeds a predetermined threshold, one trap PDU is generated. Then, the management station polls the agent further to determine the details of the condition.

In the Distributed7 platform, exceptions and errors are handled through the ALARM daemon object. The ALARM daemon object is a heavy weight UNIX process registered to the SPM multiplexing STREAMS driver. User and system applications relay their error conditions to the ALARM object via the IPC messages. The ALARM process is described in the Initial Configuration chapter. Each alarm has a severity value, INFO, MINOR, MAJOR, CRITICAL or FATAL, and a default message string. Critical alarms usually signify outage conditions which should be reported to a management station immediately.

With the introduction of the SNMP, a trap extension has been added to the ALARM object. The trap extension allows all alarms to be forwarded to the SNMP agent. By default, no alarms are forwarded, however it is possible to make the alarm daemon forward all alarms (See *Chapter 7: System Processes AccessAlarm*). As an example, Figure 6-15 shows the path that an alarm generated by the *upmd* daemon takes through the system and to the SNMP trap. First, an alarm IPC message is sent to the Alarm Object by *upmd*. The Alarm Object's trap extension copies the alarm message and sends it to the SNMP agent in an IPC message. The SNMP agent parses the alarm message prepares the TRAP PDU and sends it to the manager station. The *alm_trap()* function call has been designed to establish an interface between the alarm daemon on the SNMP agent for generating SNMP traps and notifying remote network management entities.

To forward alarm messages to the SNMP agent, the ***alm_trap()*** function is used. The files *alarmd.o* and *AlarmExt.c* are provided in the ***$EBSHOME/access/sample/alarm*** directory. AlarmExt.c contains two functions:

`void P_InitExt(void)` - The P_InitExt() routine is called only once when the ***alarmd*** starts running. Users can use this routine to customize the initialization.

`void P_AlarmExt(APIalarm_t *alarm_ptr, int fd)` - The ***alarmd*** makes a call to the P_AlarmExt() routine after processing each and every alarm message generated by the Distributed7 platform. Users can modify this routine to append their custom processing for each and every alarm message received by the ***alarmd*** daemon. Alarm messages can be forwarded to the SNMP agent selectively using ***alm_trap()***. This check should be done in P_AlarmExt().

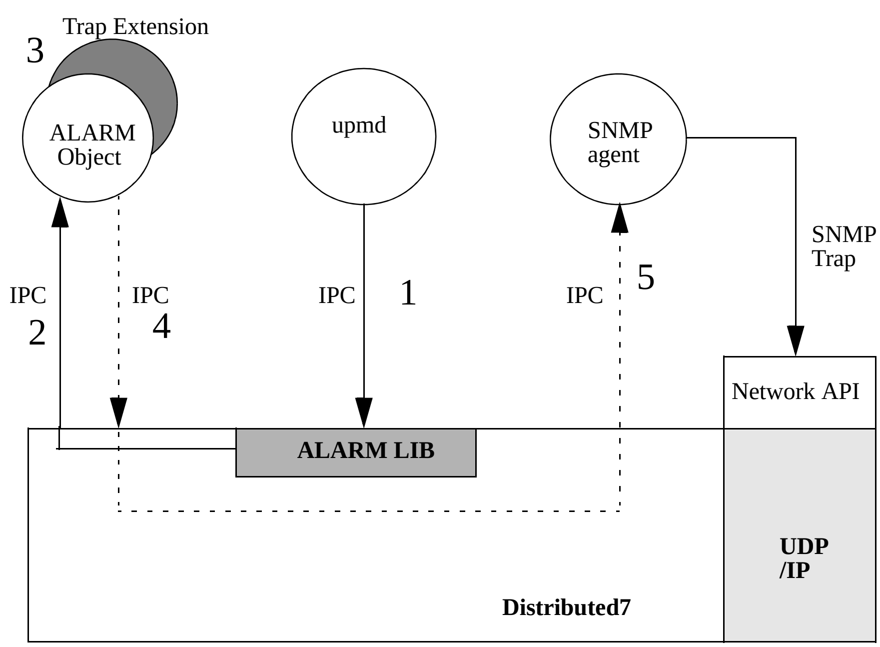


Figure 6-15: Proposed SNMP Trap Reporting Mechanism

6.5.5.3 Adding New Managed Object Definitions

This section describes the actions which must be taken to add new managed objects that will be maintained by the SNMP agent. Three files must be updated for a new managed object:

- mib_text.v1 MIB file (for SNMPv1 only)
- mib_text.v2 MIB file (for SNMPv2 only)
- snmp_cmnd.tbl file

These files are located in the ***$EBSHOME/access/RUN/config/SNMP*** directory.

The definition of the MO must be added to the **mib_text.v1** and/or **mib_text.v2** MIB files using the *ASN1 notation*. The relationship between the MIB definition and the object server definitions must be subsequently defined in the **snmp_cmnd.tbl** file. The SNMP agent sends log messages that contain information about the parsing steps of the **mib_text.v1**, **mib_text.v2**, and **snmp_cmnd.tbl** files to the current ***Mlog.****ddmmyy* file in the ***$EBSHOME/access/RUN/mlog*** directory. This log file may be viewed to determine whether all the new objects are defined to the Distributed7 system. The agent also sends information about any abnormalities that occur.

Examples for a currently defined MO and a brand new MO are provided on the following pages.

MIB File

The *level2 timer* managed object (e.g. *l2timerTable, l2timerEntry* - see Figure 6-3) has been defined in the **mib_text.v2** file as shown below.

```
            l2timerTable OBJECT-TYPE
    SYNTAX SEQUENCE OF L2timerEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "MTP Level2 Timer Information"
::={ level2 1 }

l2timerEntry OBJECT-TYPE
    SYNTAX L2timerEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A particular Level2 Timer Information"
    INDEX { l2tLink, l2tTimer }
::={ l2timerTable 1 }
```

```
L2timerEntry ::=
SEQUENCE {
    l2tLink OCTET STRING,
    l2tTimer INTEGER,
    l2tValue INTEGER,
    l2tMinval INTEGER,
    l2tMaxval INTEGER,
}

l2tLink OBJECT-TYPE
    SYNTAX OCTET STRING (SIZE(12))
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Link Name of Level2 timer"
::={ l2timerEntry 1 }

l2tTimer OBJECT-TYPE
    SYNTAX INTEGER (1..8)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "One of the seven timers:
        1 : T1: Alignment ready
        2 : T2: Not aligned
        3 : T3: Aligned
        4 :T4N: Normal Proving period timer
        8 :T4E: Emergancy Proving period timer
        5 : T5: Sending SIB (Status indication BUSY)
        6 : T6: Remote congestion
        7 : T7: Excessive delay of acknowledgment"

            ::={ l2timerEntry 2 }

            l2tValue OBJECT-TYPE
```



```
        SYNTAX INTEGER (0..600000)
        ACCESS read-write
        STATUS mandatory
        DESCRIPTION
            "Timer value for corresponding Level2 timer"
    ::={ l2timerEntry 3 }

    l2tMinval OBJECT-TYPE
        SYNTAX INTEGER (0..600000)
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Min Timer value for corresponding Level 2 timer"
    ::={ l2timerEntry 4 }

    l2tMaxval OBJECT-TYPE
        SYNTAX INTEGER (0..600000)
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Max Timer value for corresponding Level2 timer"
    ::={ l2timerEntry 5 }
```

Command Table File

In the **snmp_cmnd.tbl** file, the *level2 timer* managed object is mapped to the object server definition (see Figure 6-3), as shown in the following lines.

```
#
# SNMP requests re: MTP-Level 2 managed objects
#

l2timerEntry,L2TIMER;
l2flowEntry,L2FLOW;
l2csEntry,L2CS;

#
# SNMP requests re: MTP-Level 3 managed objects
#

mtpEntry,MTP;
spEntry,SP;
aliasEntry,ALIAS;
```

```
lsetEntry,LSET;
lsetstatEntry,LSETSTAT;
linkEntry,LINK;
linkstatEntry,LINKSTAT;
routeEntry,ROUTE;
rtsetEntry,RTSET;
l3timerEntry,L3TIMER;
sltimerEntry,SLTIMER;

#
# SNMP requests re: SCCP managed objects
#

snspEntry,SNSP;
subsysEntry,SUBSYS;
cpcEntry,CPC;
mateEntry,MATE;
gtEntry,GT;
connectionEntry,CONNECTION;
gtentryEntry,GTENTRY;

#
# SNMP requests re: ISUP managed objects
#

isupEntry,ISUP;
isupnodeEntry,ISUPNODE;
isupcgrpEntry,ISUPCGRP;
isupcctEntry,ISUPCCT;
isuptmrEntry,ISUPTMR;

#
# SNMP requests re: HARDWARE managed objects
#

ss7boardEntry,SS7BOARD;
lineEntry,LINE;
portEntry,PORT;

#
# SNMP requests re: NTWK managed objects
#

ntwkEntry,NTWK;
hostEntry,HOST;
tcpconEntry,TCPCON;

#
# SNMP requests re: ALARM managed objects
#

alarmEntry,ALARM;
almgrpEntry,ALMGRP;
```


strdalmEntry,STRDALM;
almeventEntry,ALMEVENT;

MIB File

As an example, a new MO called **aabb** is added to the MIB file. The MO is defined with name **aabb** and has the following parameter list: **key**(integer, read-create), **val1**(integer, read-write), **val2**(string, read-only), **val3**(integer, write-only). This MO is under the **newlev** branch, which is under the **accessMANAGER** node.

***Important**: (To add or delete an MO instance with the SNMP protocol, the last parameter of each MO should be a RowStatus type of parameter. This type was described on page 6-20.)*

The following lines must be added to the MIB file, **mib_text.v2**.

```
newlev OBJECT IDENTIFIER ::={ accessMANAGER x }

aabbTable OBJECT-TYPE
    SYNTAX SEQUENCE OF AabbEntry
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "."
::={ newlev 1}

aabbEntry OBJECT-TYPE
    SYNTAX AabbEntry
    MAX-ACCESS not-accessible
    STATUS  current
    DESCRIPTION
        "."
    INDEX { key }
::={ aabbTable 1 }

AabbEntry ::=
SEQUENCE {
    key  INTEGER,
    val1  INTEGER,
    val2  STRING,
    val3 INTEGER,
    aabbStatus RowStatus
}

     key   OBJECT-TYPE
    SYNTAX INTEGER
    MAX-ACCESS read-create
    STATUS current
    DESCRIPTION
        " "
::={ aabbEntry 1 }
```

```
val1   OBJECT-TYPE
       SYNTAX INTEGER
       MAX-ACCESS read-write
       STATUS current
       DESCRIPTION
              ""
::={ aabbEntry 2 }

val2   OBJECT-TYPE
       SYNTAX STRING
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION
              ""
::={ aabbEntry 3 }

val3   OBJECT-TYPE
       SYNTAX INTEGER
       MAX-ACCESS read-create
       STATUS current
       DESCRIPTION
              ""
::={ aabbEntry 4 }

aabbStatus OBJECT-TYPE
       SYNTAX RowStatus
       MAX-ACCESS read-create
       STATUS current
       DESCRIPTION
              "For row creation."
::={ aabbEntry 5 }
```

Command Table File

The new MO called **aabb** is then added to the **snmp_cmnd.tbl** file, as follows.

```
# snmp entryname , object name;
#
aabbEntry,aabb;
```

6.6 Using AccessMOB

6.6.1 Introduction

The Graphical User Interface (GUI) for Distributed7 is called the Managed Object Browser (MOB). This interface displays the hierarchical model of Distributed7 on the screen and permits convenient access through *point-and-click* mouse sequences instead of through typed commands. Managed Objects (MO) presented in tree form, can be selected at will for viewing or modifying.

6.6.1.1 Requirements

To access the Managed Object Browser, you will need the following:

- Distributed7 software;
- X Window System Release 5 (X11R5) server software, or equivalent, e.g., OpenWindows for Solaris, and the corresponding shared (run-time) libraries;
- Motif Version 1.2.4 shared (run-time) libraries. (See the *Environment Variables* section);
- Motif or another compatible window manager (e.g., OPEN LOOK for Sun systems).

For additional details on using the Motif environment, refer to the Open Software Foundation's *OSF/Motif User's Guide*.

For details on using OPEN LOOK, refer to your Sun system documentation.

6.6.1.2 Environment Variable Settings

Environment variables must be set before running the Managed Object Browser, if they were not set at installation. The following commands are given for the C Shell.

1. The ***$EBSHOME*** variable must be set to the directory where the Distributed7 software was installed. If it is not set, enter the command:

   ```
   setenv EBSHOME <install_directory>
   ```

   (<install_directory> should be replaced with the actual directory path where the software is installed.)

***Important**: **$EBSHOME** can be up to 1024 characters.*

2. Check that the path in your ***.chrc*** file is set up to run the Distributed7 software. If not, set the ***$PATH*** variable with the following command:

   ```
   setenv PATH ${PATH}:$EBSHOME/access/bin
   ```

3. Set the ***$DISPLAY*** variable to the host name of your machine (network node name) using the command: `setenv DISPLAY <hostname>:0.0`
   This command can be placed in your ***.cshrc*** file.

4. The ***$LD_LIBRARY_PATH*** environment variable MAY have to be set to access shared libraries at run-time. For example, if an error such as *fatal: libXm.so.4: can't open file: errno=2* occurs, then the ***libXm.so*** shared library for Motif Version 1.2.4 cannot be found. In this case, the variable must be set according to one of the two methods below.

   a. For a variable that has other settings, enter:

   ```
   setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:$MOTIFHOME/lib
   ```

   b. For a variable that has not been set previously, enter:

   ```
   setenv LD_LIBRARY_PATH $MOTIFHOME/lib
   ```

   *(**$MOTIFHOME** is an environment variable which represents the location of the Motif installation directory for your particular system - e.g. **/usr/dt** for Sun platforms Consult your system administrator.)*

Important: *The X and Motif shared libraries used by AccessMOB are* ***libXm.so***, ***libXt.so***, ***libX11.so***, *and* ***libXext.so***. *If any of these libraries reside in non-standard directories on your system, their location must be determined and the location must be added to the* ***$LD_LIBRARY_PATH*** *variable as described above. (Consult your system administrator for help in finding the location.)*

5. Place these settings permanently in your ***.cshrc*** file.

6.6.1.3 Conventions

The following are the conventions used within this chapter:

- File names and dialog box buttons within paragraphs appear in *Bold Italic*;
- Commands that must be typed in appear in **`Courier Bold`** while the options of the command appear in `Courier` that is not bold;
- Menu names and options appear in Bold.
- Point codes are defined as Network-Cluster-Members, for ANSI versions. For ITU/CCITT versions, point codes are defined as Zone-Network-Signalling Points;
- Left and right mouse buttons will be referred to as LEFT and RIGHT.
- DOUBLE CLICK means a rapid press-release-press-release of the mouse button.

Important: *If the mouse buttons or other functions do not seem to operate as described in this manual, you can reset the entire environment to use the default behavior. To do this, press these four keys simultaneously:* ALT + CTRL + Shift + !

6.6.1.4 Starting the Managed Object Browser

To run the Managed Object Browser, the Distributed7 software must be running. The Managed Object Servers that control the MTP and SCCP managed objects will be running. Other user part managed objects, such as the ones for ISUP, can also be browsed with the MOB if their Managed Object Servers are running. (See the Initial Configuration chapter.)

The command to start the Managed Object Browser (MOB) is:

```
AccessMOB sp
```

The **`sp`** argument is the signalling point number (0, 1, 2, 3, 4, 5, 6, or 7) of a Distributed7 logical node that is already running and needs to be configured. It should be a number that was used with a *upmd* command.

The Managed Object Browser registers non-exclusively with the Distributed7 environment. Multiple copies of the MOB can exist for the same signalling point, either on the local host or across the distributed network.

The Managed Object Browser can be stopped with the Ctrl C key combination or the **Exit** option under the **File** menu. It will also be stopped automatically when the software is stopped with the *ebs_stop* command.

NOTE: If the status of a Managed Object Server changes (for example, one of the daemons is started after AccessMOB), AccessMOB is automatically updated. It does not have to be restarted.

6.6.2 Managed Object Browser

The managed object browser consists of a main window and dialog boxes. The operation mode and the managed object are selected at the main window. Then, dialog boxes appear to specify the unique managed object instance and perform the operation.

All of the dialog boxes have the same components and display information in the same manner. The parameters of a managed object are shown in a list, with a text field next to each parameter. The fields show the present values of the parameters. This value may or may not be changed by the user depending on the context. If an parameter's value cannot be changed, the parameter is shown in grey and it cannot be clicked on with the mouse.

Generally, dialog boxes pop up at full size but can be resized smaller, if desired. When necessary, scrollbars are provided to allow viewing lists that are too large to fit in a normal window. The scrollbars are activated with the mouse.

6.6.2.1 Window Managers

While window manager functions will not be discussed in this manual, knowledge of them will allow the most flexible use of the Managed Object Browser. For example, the main window of the Managed Object Browser can be *iconized* to free up screen space while viewing other subwindows. All viewing windows of the Managed Object Browser can be sized and arranged as desired. Minimizing and resizing is done using the window border or the border menu of the window manager.

The window manager controls the screen and the inputs from the mouse and keyboard to the Managed Object Browser. For example, to accept an input, the desired window of the Managed Object Browser must be selected. To select the window, the user would either click on the window with the mouse or simply move the mouse pointer inside the window, depending on the window manager. A selected window will have a color change in the window border or some other visual indication.

The program for the window manager can be started by entering the name at the bottom of the ***.xinitrc*** file that exists in the home directory. The name of the Motif Window Manager is ***mwm***. The name of the OPEN LOOK Window Manager is ***olwm***. The program must NOT be run in the background (do not use the & with the command).

6.6.2.2 Accessing Menus

The menus of the MOB can be accessed using the mouse to click and drag. The menus may also be accessed by using the keyboard. However, if [NUM LOCK] is on, the keyboard will appear to be disabled.

Two ways exist to access menus using the keyboard instead of the mouse. The first method pops up a menu so that a selection can be chosen visually. By pressing the [◇] (*Meta* or diamond) key and then the letter key of the menu name (e.g. [F] for File), that menu will display its choices. When the menu choices have been displayed (either by the mouse or a key combination) a choice can be selected by pressing the key of the underlined letter (e.g. [E] for Exit).

In a menu, keyboard actions also include using arrow keys ([↓] [↑] [→] [←]) to move the cursor, the [Ret] or [Space] keys to activate, and [Esc] to cancel.

The second method of accessing menus is through *menu accelerators*. Menu items can be directly selected WITHOUT going to the menu by using [Ctrl]-key combinations. The combinations are identified in the menus next to the associated menu item for which they apply. They are also provided in the following subsections. The Main Window must be selected as the current window for the key combinations to work. If [CAPS LOCK] is on while running the program, the [Ctrl] key menu accelerators will be disabled. They require lower case.

6.6.2.3 Using the Mouse

The following list summarizes the valid mouse actions:

- Clicking the LEFT mouse button once activates an operation in the current mode.
- Clicking the MIDDLE mouse button shows or hides the subtree of a managed object node.
- Pressing the RIGHT mouse button brings up a menu to choose an operation from a mode that is not in the current mode.
- Double clicking the LEFT mouse button opens a view box of all instances when in View mode.
- Pressing [Shift] and clicking the LEFT button, when in View mode, opens a view box of all instances.
- Pressing [Shift] and clicking the RIGHT button, when in a mode other than View, opens the popup menu for selection of the view operation to view all instances.

***Important**: If the mouse buttons or other functions do not seem to operate as described in this manual, you can reset the entire environment to use the default behavior. To do this, press these four keys simultaneously:* [ALT] + [CTRL] + [Shift] + [!]

6.6.2.4 Entering Data in the Dialog Box

Dialog boxes are used to enter data for a managed object selected from the main window. First, a unique instance of the managed object is identified in a key dialog box. Then, another dialog box will appear in which to perform an operation. The data entry into any of the dialog boxes follows the same general rules.

A field can be selected by clicking on it with the mouse. Movement between the fields can also be accomplished using the [Tab] key to go down the list or the [Shift] + [Tab] combination to go up the list. The [Ret] key acts the same as the [Tab] key. Each time [Ret] is hit, the next field down on the list is made active for input.

Within the field, the [BackSpace], [Delete], [Insert], [Home], [End] and the arrow keys ([↓] [↑] [→] [←]) may be used for editing. The mouse may also be used to point and click directly. Copy and Paste operations are available using the LEFT and MIDDLE buttons of the mouse or by using the keyboard.

A *Range* or *Set* menu that appears at the end of the field can also be used to input the data. Please see Range Menus and Set Menus and Set Type Values for more details.

After data is entered or changed, the ***Apply*** button must be selected to complete the operation. The mouse can be used to click on the button. If the ***Apply*** button is indicated as the current button by an outline around it, either the [Ret] or [Space] key can be pressed to complete the operation.

Selecting the ***Cancel*** button closes the dialog box without making any changes. Usually this is done by clicking on the button with the mouse. However, if ***Cancel*** is the outlined button, then either the [Ret] or [Space] key can be used.

Range Menus

A *Range* menu occurs on the right side of an integer field if the allowed range of values is small enough (e.g. 1 - 32). It lists the allowed values, from minimum to maximum. The *Range* menu is viewed by clicking on the ***Range*** button with the LEFT mouse button or by pressing the [◇] [R] (Meta or diamond key and R) key combination when the desired field is the active field.

A value can be selected directly from the menu by using the mouse. That value is transferred into the text box, eliminating the need to type it. Within the *Range* menu, the up and down arrow keys ([↓] [↑]), [Ret], [Space], and [Esc] may be used.

Set Menus and Set Type Values

A *Set* menu occurs on the right side of a string or numeric field if it is restricted to a small number of valid values (e.g. ON, OFF). The *Set* type is provided to reduce the possibilities of error and the amount of typing required for a string value. The *Set* menu is viewed by clicking on the ***Set*** button with the LEFT mouse button or by pressing the [◇] [R] (Meta or diamond key and R) key combination when the desired field is the active field.

A value can be selected from the menu using the mouse. That value is transferred into the text box, eliminating the need to type it. Within the *Set* menu, the up and down arrow keys ([↓] [↑]), [Ret], [Space], and [Esc] may be used.

When typing in the value for this type of field, only the initial characters which uniquely identify a value from the set are required. The value will be automatically completed and accepted. For example, in the set {ON, OFF} it is necessary to type the first two letters to identify the choice.

For long strings, the [Esc] key may be used to perform a partial *completion* while you are typing a set value. For example, in the following set,

{sbs332, sbs334, sbs370, sbs372, ax7000, pri200}

typing an [S] followed by [Esc] will automatically provide the substring **sbs3**. This is the maximum substring identified by the given character. Then, you must type the rest of the characters needed to uniquely identify one element of the set. The [Esc] key may complete the set value if that is possible. However, using the [Esc] key to complete a value is not necessary. The automatic completion occurs when you move to a different text entry field or when the ***Apply*** button is selected. If a value could not be completed to form a valid set member value, an error message will appear.

***Important**: For values which have multiple completions (i.e. the set {I, II, III}), any substring which is entered will be accepted. Be careful to enter the appropriate value.*

6.6.2.5 Managed Objects Parameters

Managed objects are a functional or physical resource of the system, such as subsystems or linksets. For example, each box in the Main Window of the Managed Object Browser (Figure 6-16) is a managed object. Each managed object has a set of operations (add, modify, delete, view) that are allowed to be performed on it. An individual instance of a managed object, such as a specific linkset, is defined by its parameters. They provide the managed object with a unique identity.

A more in-depth description of managed objects can be found in *Chapter 2: Distributed7 Overview*. However, information about parameters are provided in the following subsections to provide a better understanding of data entry in the Managed Object Browser.

Key Parameters

Key parameters are identified by a key symbol as seen in Figure 6-17 on page 6-52. In the key selection dialog box, only the key parameters are listed for input by the user. The other dialog boxes show the key values, but do not allow them to be changed. They cannot be changed because they act as the *title* of a particular instance.

Data Types

The data type of an parameter identifies whether it must be a numerical value, a point code value, or a general alphabetical string. The data type can either be a *Set* type or a *Range* type which is deduced from the range information in the **Range** popup menu next to the parameter's value field. Information on sets or ranges can also be found in the MMI/MML chapters of this manual. Menus can be viewed by clicking on the ***Range*** or ***Set*** button with the LEFT mouse button (see Figure 6-17). These menus list the complete set of allowed values or the range of values, from minimum to maximum. Values can be selected directly from these menus (see Range Menus).

Values entered in the fields are checked for the data type and the range. Error messages will indicate any illegal values that need to be corrected. The constraints of each data type are described below. Chapter 2 of this manual contains tables which identify the data types of all managed object parameters.

Integer

- Must be an integer value within the range shown; the minimum and maximum values are valid.
- May include a K or M suffix (lower or upper case) after the value to indicate thousands. *This is NOT binary (1024).*

Point Code

- Must be three sets of integers with a dash between each set.

- Must be within the range and format required by the protocol version (ANSI or CCITT) and identified by the Range popup menu.

Set Type

- Must be chosen from a given list of numeric or alphanumeric values.
- Displays allowed values in the Set popup menu.
- Requires only initial characters to be typed to identify a value (see Set Menus and Set Type Values).

String

- May be any set of alphanumeric characters, except the asterisk *. (See the Wildcards section.)
- Must have a character length within the specified range.

Wildcards

The special character, *, represents a *wildcard* value. A wildcard means that ALL existing values of a particular parameter are selected. It can only be used for KEY-type parameters chosen in the **Keys** selection dialog. The * can be used for viewing instances of a Managed Object as a group. If there is one key, then all of them are viewed. If there are several keys, the instances may be viewed by category.

The * character can be used for any data type, *Integer*, *Point Code*, *Set Type*, or *String*. However, the following limitations on wildcard usage exist:

- Existing Managed Objects only accept ONE wildcard in the key values list, if there are multiple keys. Refer to the specific MMI/MML commands to see which Managed Object parameter key will accept a wildcard as a value.
- Wildcards can only be used in the View operation. The other operations require full specification of a unique instance.
- Wildcards can only be used for a key parameter.

Access Types

An access type determines the type of access a user has to a parameter of a managed object. They identify which operations the user is allowed to use on a parameter - view, add, delete, or modify. Illegal operations will result in an error message. *Chapter 2* of this manual contains tables which identify the access types of all managed object parameters. The chapters on MMI/MML commands also identify which parameters are valid for a particular operation. The four access types are defined as follows:

READ-WRITE

- Parameter is always displayed.
- Parameter can be modified in Add and Modify dialog boxes.
- Entry of a value can be optional.

READ-ONLY

- Parameter is always displayed.
  (Usually status information from the Managed Object Server)
- Parameter cannot be modified.

READ-CREATE

- Parameter is always displayed.
- Parameter can be defined in Add dialog boxes.
- Parameter cannot be modified in Modify dialog boxes.
- Entry of a value can be optional.
- Parameter is or behaves like a key parameter but does not have to be one.

WRITE-ONLY

- Parameter can be supplied by the user in Add, Delete, and Modify dialog boxes.
- Entry of a value can be optional (default value exists).
- Entry is usually a setting for an operation. (e.g. a range of instances to Add or Delete)

The special WRITE-ONLY parameter type is identified by a *pen* symbol on the left side of the text entry field.

6.6.3 Managed Object Browser Windows and Operation

6.6.3.1 The Main Window

Figure 6-16 is an example of the Managed Object Browser's main window. If other MO Servers, such as ISUP, are running, the associated managed objects will appear.

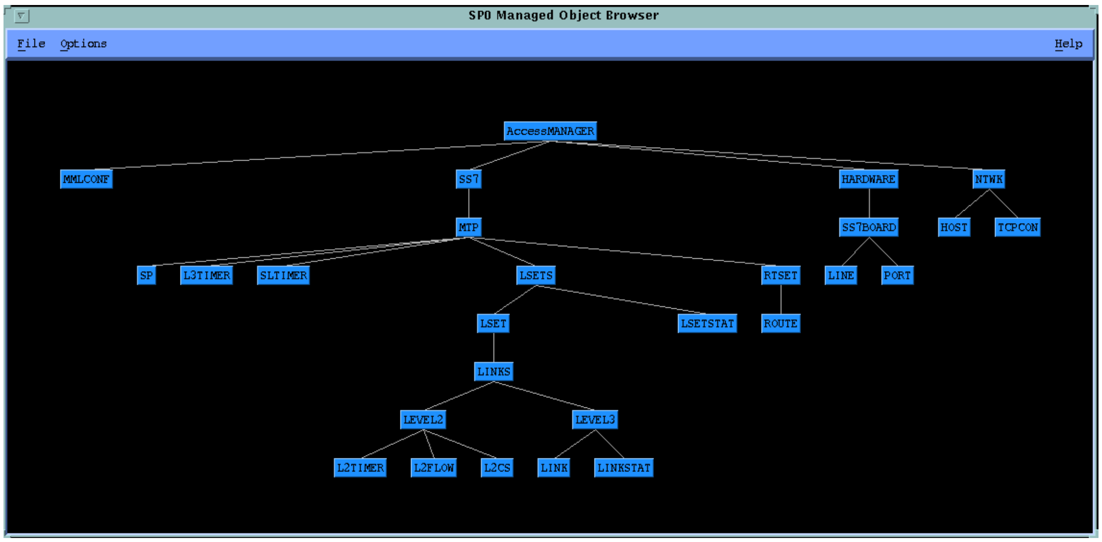


Figure 6-16: MOB Main Window

The main window of the MOB contains a tree representing the managed objects in your Distributed7 environment. The subtree of a managed object node can be shown by a single click of the MIDDLE mouse button on the desired node. A displayed subtree can be hidden by the same action. Subtrees can also be shown or hidden by holding down the RIGHT mouse button while over the node and selecting the **Show/Hide** choice from the five-color popup menu that appears.

The main window supports the following actions and inputs:

- Pulldown menus selected with the LEFT mouse button
- Keyboard *quick key* combinations that perform the menu actions
- Mouse point and click operations on the nodes of the tree

Three menus exist at the top of the Managed Object Browser main window. They are *File*, *Options*, and *Help*. A menu is accessed by placing the mouse on the menu name and clicking the LEFT mouse button. The available options will be displayed.

File Menu

The only choice in the **File** menu is **Exit**. When **Exit** is selected, all open dialog windows are closed and the program is ended. The Ctrl C key combination can also be used.

Options Menu

The **Options** menu allows the user to select the mode and to select the display style for the tree. The Ctrl-key combination for the choice is shown beside it. The choices are:

- View - to choose the mode for viewing managed objects (Ctrl V)
- Modify - to choose the mode for modifying an instance of a managed object (Ctrl M)
- Add - to choose the mode for adding an instance of a managed object (Ctrl A)
- Delete - to choose the mode for deleting an instance of a managed object (Ctrl D)
- Refresh Tree - to refresh the managed object tree when the managed object configuration has changed *(the tree is normally checked and automatically refreshed while the program is running, so using this selection is unnecessary)*
- V Tree - to choose a vertical display of the tree
- H Tree - to choose a horizontal display of the tree
- Dialog Auto Place - to enable an alternate method of positioning new *View* dialogs on the screen (Ctrl P). When this option is set (square indicator appears at left), *View* dialogs are popped up around the edge of the screen, instead of being placed according to your window manager's default placement.
- Change Title - to set a new title for windows (Ctrl T)
  Windows already open when the window title is changed will not display the change but new windows that appear after the change is made will have the new title.

Help Menu

The **Help** menu provides information about the main window and modes. The choices are:

- Help on AccessMOB - describes the main application window
- Managed Object Tree - describes the mouse actions pertaining to the tree
- Keys Dialog - describes the dialog box for entering key choices
- View Dialog - describes the *view* mode and its dialog box
- Modify Dialog - describes the *modify* mode and its dialog box
- Add Dialog - describes the *add* mode and its dialog box
- Delete Dialog - describes the *delete* mode and its dialog box

6.6.3.2 Selecting an Operation Mode

The Managed Object Browser operates in one of four modes, each identified by a specific color. The modes are View (blue), Modify (yellow), Add (green), and Delete (red). When a given mode is active, the managed object tree is shown in the associated color. After selecting a mode, an operation is initiated by a single click of the LEFT mouse button on the desired managed object.

The mode is set using the **Options** menu or a Ctrl key combination. The combinations are Ctrl A for Add, Ctrl M for Modify, Ctrl D for Delete, and Ctrl V for View, as shown in the menu. These key combinations are called *menu accelerators*. See *Accessing Menus on page 6-43*.

One mode can be quickly accessed from another for a single operation (e.g. to modify a single managed object while in view mode). To access a mode in this way, the RIGHT mouse button should be held down while the cursor is over the desired managed object. A five-color popup menu appears over the node while the button is still held down. The cursor should be moved to the desired operation and the button released to choose that operation, similar to a menu. The overall mode does not change after the operation is complete.

Important: *<CAPS LOCK> cannot be used for the above actions. If <CAPS LOCK> is on while running the program, the* Ctrl *key menu accelerators will be disabled. They require lower case.*

Important: *<NUM LOCK> cannot be set while the Managed Object Browser is running. If <NUM LOCK> is on, the keyboard will appear to be disabled.*

6.6.3.3 Selecting Managed Objects

Once the mode is selected, you must pick the managed object to perform an operation on. An operation is initiated by a single click of the LEFT mouse button on the desired managed object box in the main window. Remember, if the desired managed object is in a subtree that is hidden, simply click the MIDDLE mouse button on the node of the managed object, then select the managed object.

Note: Some managed objects in the tree have no associated operations (example Distributed7 at the top of the tree). These objects only serve as parents for other managed objects. If one of these objects is selected, an error message popup window will appear.

For Modify and Delete operations, a managed object instance can also be selected from the managed object's *View* dialog list. The view list displays all instances of a managed object. This method is described in *Selecting Other Modes From the View Dialog Box*.

After selecting a valid managed object, a popup key selection dialog box similar to Figure 6-17 appears. The dialog box shows the managed object name that was selected and the key parameter(s) for which a value must be supplied. Only the key parameters are listed in this box. Other dialog boxes will show the key values, but do not allow them to be changed. In this dialog box, space is available next to each key parameter name to enter the key value. If more than one key parameter appears in the list, all must be specified in order to uniquely identify the single instance. Entering data in a dialog box is covered in Section 6.6.2.4 on page 6-44.

The *Range* or *Set* menu on the right side of each key field contains the possible values that can be entered. The field's menu is viewed by clicking on the button with the LEFT mouse button. The field's value should be a new value when in Add mode, or a known value when in View, Delete, or Modify modes. Normally, only one instance may be added, modified, or deleted at a time.

The managed object instance is chosen by clicking on the ***Apply*** button of the dialog box. When this dialog box closes, an operation dialog box appears containing information for the chosen managed object instance.

The selection can be ended by clicking on the ***Cancel*** button.

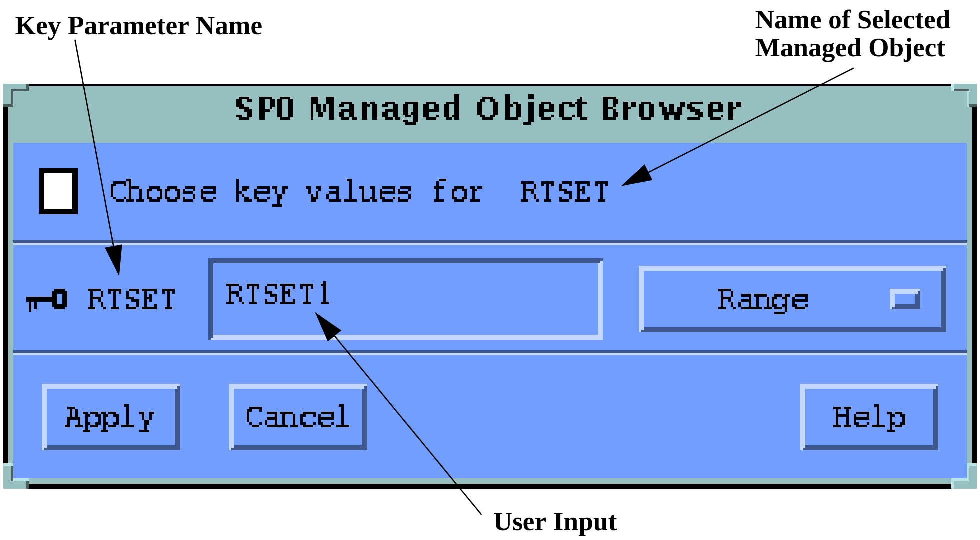


Figure 6-17: Key Selection Dialog Box

For the view operation, managed object instances can be viewed all at once instead of individually through the key dialog box. By double clicking the LEFT mouse button or pressing Shift and clicking a mouse button on the desired managed object class, the keys dialog box will be bypassed and a window will appear showing ALL existing instances of a managed object. This viewing operation can be invoked at any time, even with a key dialog box open.

Important*: Only one Modify, one Add, and one Delete dialog box can be open at one time, but an unlimited number of View dialogs can be open simultaneously.*
While a Key dialog box remains open, no other dialogs can be opened except View dialogs for ALL instances.

6.6.3.4 Operation Dialog Boxes

The following sections show the four types of dialog boxes that exist. Each type of dialog box will have unique action buttons. During use, the contents of the dialog boxes will differ based on the managed object that was selected. However, the functions remain the same.

Each dialog box type shows the color code associated with the operation at the top of the box, to the left of the operation name. The managed object class is identified to the right of the operation name.

Specific characteristics of each dialog box are described following each figure.

Add Dialog Box

Figure 6-18 shows a sample Add Dialog Box for the managed object, *RTSET*. The box will be similar for any managed object. The list of parameters and which ones cannot be changed will be unique to each managed object.

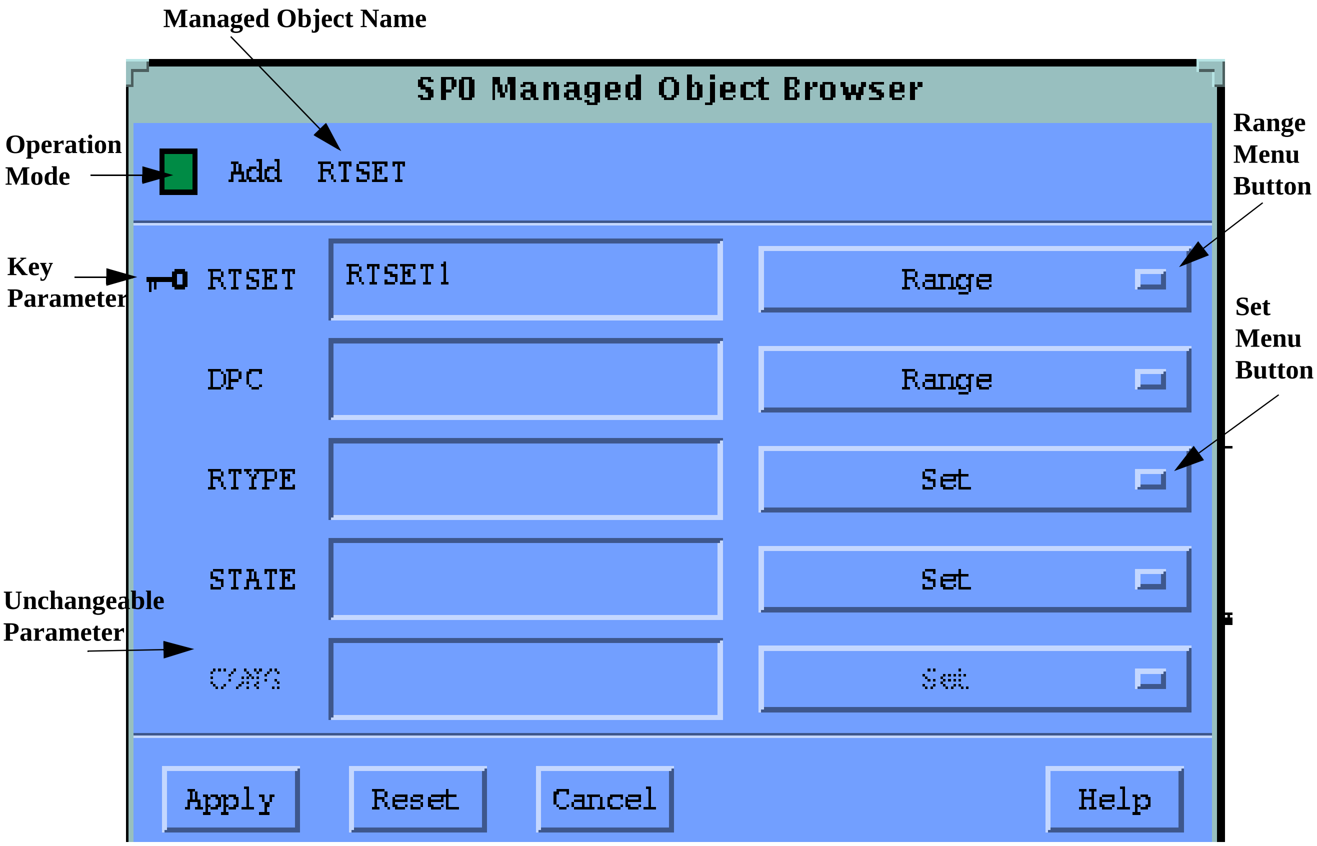


Figure 6-18: Add Dialog Box

When the Add Dialog Box first appears, the key parameter values and any defined default values are shown. Default values can be accepted or changed. Some empty fields require an entry, while others do not. The chapters on MMI/MML commands will identify those parameters that have default values or are optional.

Key field values can be changed. If a different key value is desired, select the field using the mouse, the Tab key, or Shift + Tab, then edit the value. Any parameters shown in grey (e.g. CONG in Figure 6-18) cannot be set by the user.

The ***Apply*** button completes the Add operation to create an instance of the managed object. The ***Cancel*** button exits from the Add operation without making any changes. The ***Reset*** button clears the entries in all fields and resets the key fields to their original values.

To add a managed object instance:

1. Select the Add operation mode in the main window. Either select **Add** from the *Options* menu or press Ctrl A.
2. Click the LEFT mouse button once on the managed object in the main window.

3. Enter values for all fields in the key selection dialog box that pops up. (Figure 6-17 on page 6-52)
4. Enter the values for all required and desired fields in the Add operation dialog box.
5. Click the ***Apply*** button.

Modify Dialog Box

Figure 6-19 shows a sample Modify Dialog Box for the managed object, *LSET*. The box will be similar for any managed object. The list of parameters and which ones cannot be changed will be unique to each managed object.

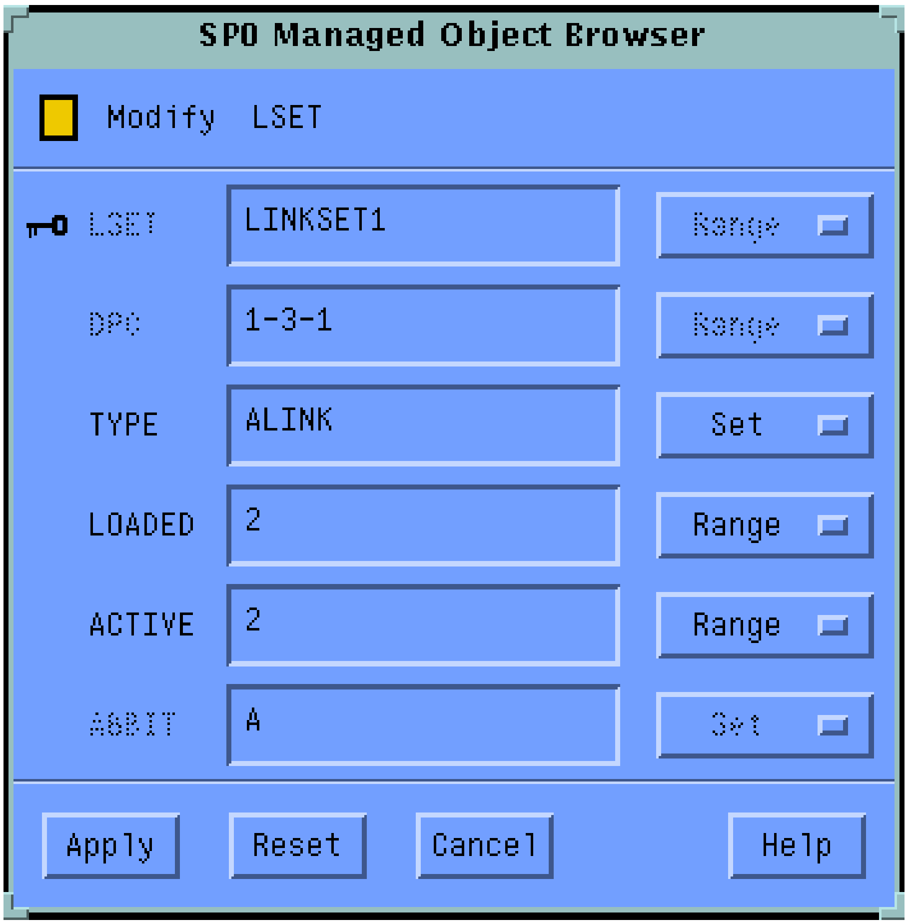


Figure 6-19: Modify Dialog Box

When the Modify Dialog Box appears, the current parameter settings are displayed in the fields. Any parameters shown in grey cannot be changed by the user. The chapters on MMI/ MML commands describe the parameter fields and the valid settings.

The ***Apply*** button completes the Modify operation. The ***Cancel*** button exits from the Modify operation without making any changes. The ***Reset*** button sets all entries in the fields back to their original settings before any changes were made.

To modify a managed object instance:

1. Select the Modify operation mode in the main window. Either select **Modify** from the *Options* menu or press Ctrl M.
2. Click the LEFT mouse button once on the managed object in the main window.
3. Identify the instance through the key selection dialog box that pops up. (Figure 6-17 on page 6-52)
4. Enter the values to be modified in the Modify operation dialog box.
5. Click the ***Apply*** button.

Delete Dialog Box

Figure 6-20 shows a sample Delete Dialog Box for the managed object, *LSET*. The box will be similar for any managed object. The list of parameters will be unique to each managed object.

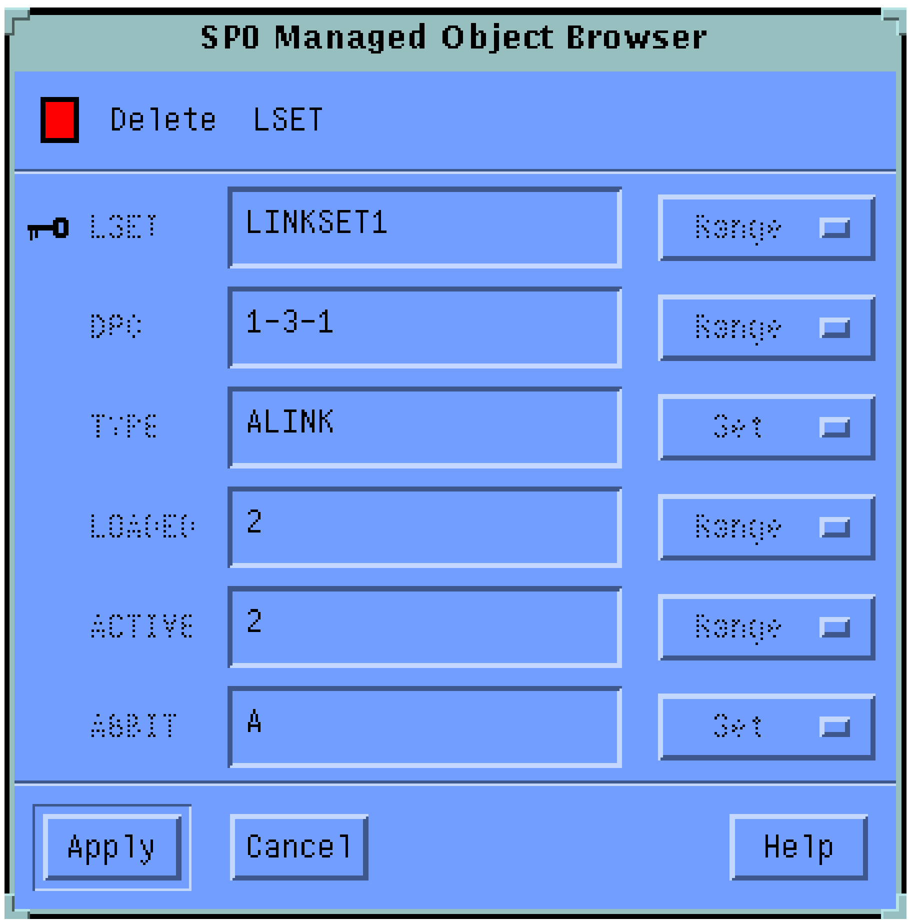


Figure 6-20: Delete Dialog Box

When the Delete Dialog Box appears, all parameter settings are displayed, but shown in grey. However, if WRITE-ONLY parameters exist, a value can be entered (e.g. a range to be deleted).

The only action to take is to select the ***Apply*** button to delete the instance of the managed object, or to select the ***Cancel*** button to exit from the Delete operation without deleting the instance.

To delete a managed object instance:

1. Select the Delete operation mode in the main window. Either select **Delete** from the *Options* menu or press Ctrl D.
2. Click the LEFT mouse button once on the managed object in the main window.
3. Identify the instance through the key selection dialog box that pops up. (Figure 6-17 on page 6-52)
4. Enter any values that are writable in the Delete operation box (e.g. range to be deleted)
5. Click the ***Apply*** button.

View Dialog Box

The view operation can be performed for a single instance of a managed object or for all instances of a managed object.

To view a single instance:

1. Select the View operation mode in the main window. Either select **View** from the *Options* menu or press Ctrl V.
2. Click the LEFT mouse button once on the managed object in the main window.
3. Identify the instance through the key selection dialog box that pops up. (Figure 6-17 on page 6-52)

A window will appear showing the parameters of the single instance.

To view all instances:

1. Select the View operation mode in the main window. Either select **View** from the *Options* menu or press Ctrl V.
2. Double click the LEFT mouse button or press Shift and click the LEFT mouse button on the desired managed object in the main window. The Shift key must remain held down until the mouse button is released.

A window will appear showing ALL existing instances of a managed object. This viewing operation can be invoked at any time, even with a key dialog box open.

When in other operation modes, a View dialog with all instances can be opened by holding down the Shift key while clicking the RIGHT mouse button and then selecting the View operation from the menu that appears.

Figure 6-21 shows a sample View Dialog Box for the managed object, *LSET*. The box will be similar for any managed object, but the output will differ.

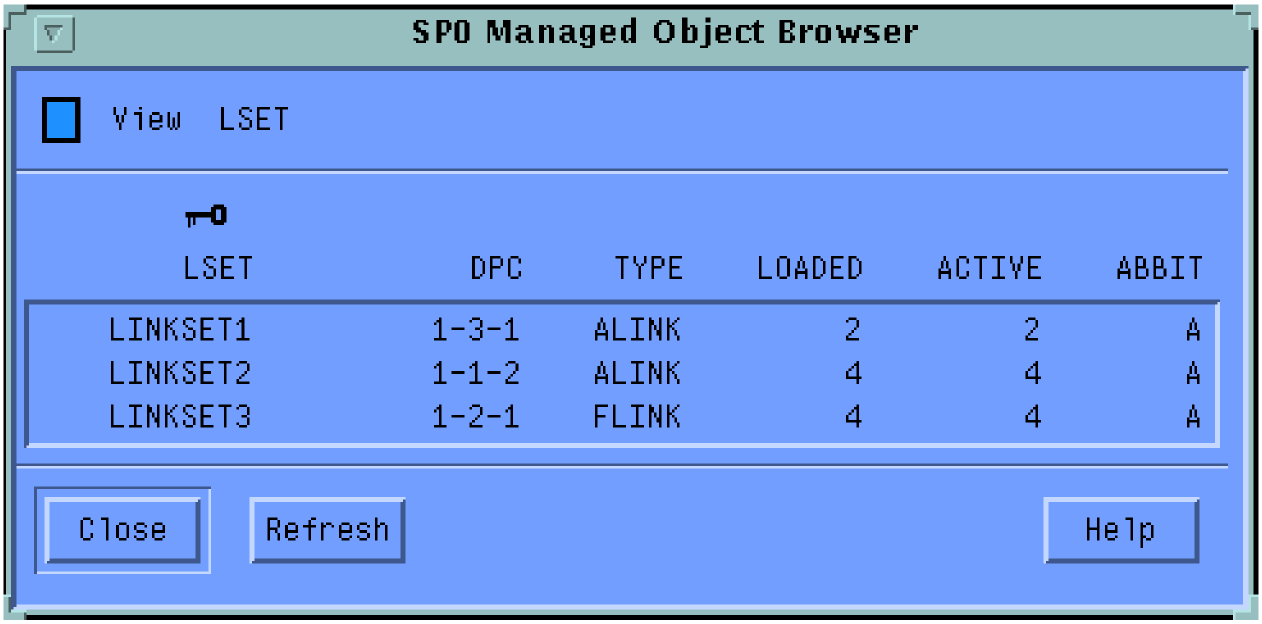


Figure 6-21: View Dialog Box

The View Dialog Box can display one or multiple instances of a managed object. No information can be changed on this window.

A refresh of the screen occurs at a predetermined time interval to retrieve any changes in current information from the Managed Object Server. The ***Refresh*** button can also be selected by a mouse click to force a refresh.

The window will stay open until the ***Close*** button is selected. The window may stay open while other dialog boxes are being used.

*Note: View Dialog Boxes are positioned on the screen according to the **Dialog Auto Place** option in the **Options** menu (Options Menu on page 6-50). When the option is set **on** (the default), View Dialog boxes will be popped up around the edge of your screen in a tiled, non-overlapping manner, for convenience of viewing. When set **off**, the dialog boxes will be placed according to your window manager's current default placement.*

Selecting Other Modes From the View Dialog Box

The View Dialog Box allows other dialog boxes to be called up for a selected instance in the box. Selecting a displayed instance from a View Dialog Box will create a new dialog box for that individual instance in the current operation mode of the main tree (View, Modify, Add, or Delete). This selection action is allowed in <u>any</u> View Dialog Box, whether there is one instance displayed or a list of instances. For example, an instance could be selected for modification from a *view-all* list, without having to enter its key values.

To create a new dialog box in the current mode from a view box, click on any instance in the dialog display area to select it. Then, click on it again to create a new dialog. A double click combines these two actions.

To create a dialog in any chosen mode, press on the instance with the RIGHT mouse button but do not release it. A popup menu will appear. Use the cursor to choose View, Modify, Add, or Delete. When the mouse button is released, a dialog box in that mode for the selected instance will appear.

The following keys and key combinations may also be used:

- The [Tab] key or [Shift] [Tab] can be used to move between the action buttons and the display area of the dialog box.
- The [↑], [↓], [Ctrl] [Home], and [Ctrl] [End] keys are used to move through the list to select an instance.
- The [Ret] or [Space] key creates the new dialog for the currently highlighted instance.

6.6.4 Error Messages

Error messages are issued either by the Managed Object Browser or by the Managed Object Servers. The Managed Object Browser performs syntactic and range checks on the entered values and produces error messages when problems occur.

If no syntax or range errors occur at the MOB level, the information is sent to the Managed Object Server. If the operation could not be performed, an error message will be returned indicating a failure and the reason for the failure.

Managed Object Browser Error Messages

Cannot attach to apmd environment

SPM connection error

Either the signalling point software or the APM daemon is not running. Start the software before executing AccessMOB.

Inapplicable operation: <operation>

The selected operation, ADD, MODIFY, or DELETE, is not permitted or not meaningful for this Managed Object.

Privileged operation: <operation>

The selected operation, VIEW, ADD, MODIFY, or DELETE, is not permitted for this user on this Managed Object. The operation is protected and can be performed only by a user on the Managed Object's access control list.

No operations defined for <MO-name>

The node that was selected on the managed object tree does not correspond to a managed object that can be viewed, created, or modified by the user.

<MO-name> Managed Object Server not available

The daemon process responsible for <MO-name> is not running (*upmd, snmd, scmd, or isupd)*. The required daemon should be started.

<MO-name> Managed Object Server communication timeout

Communication failed between the Managed Object Browser and the daemon process responsible for <MO-name>. To resolve, the operation can be retried, the MOB can be restarted with *AccessMOB*, or network/system problems can be investigated.

<MO-name> has no instances - Press OK to close View dialog

A view dialog box became invalid when an automatic refresh (or forced refresh) occurred. The box becomes invalid when instances of a managed object no longer exist; for example, all instances of a Managed Object may have been deleted since the box was last updated. The dialog box will be closed once *OK* is selected.

Managed Objects changed - Press OK to close invalid dialogs

One or more dialog boxes became invalid when an automatic (or forced) refresh of the managed object tree occurred. This means that the tree has changed and managed objects that were being accessed are no longer available. The affected dialog boxes will be closed once OK is selected, but other dialog boxes will remain open.

Instance already exists

An ADD operation was attempted using one or more key parameters that match an existing instance.

<operation>: Wildcard not allowed

A wildcard is only permitted for the VIEW operation. For ADD, MODIFY, or DELETE, a specific instance must be chosen by supplying all key values.

Wildcard not allowed for: <parameter-name>

The named parameter does not accept a wildcard value. Only certain keys accept a wildcard, which is a characteristic defined for the specific Managed Object.

Value out of range: <value>

The value that was entered is not within the specified range. The Range popup menu and the MML chapter identify the valid range of values.

Integer required for: <parameter-name>

The value entered for the named parameter is not a valid integer or an integer suffixed with *K* or *M*.

Point code out of range or incorrect format: <value>

The value entered is not a valid point code for the protocol standard (ANSI or CCITT) being used. The Range popup menu shows the correct format and valid minimum and maximum values.

Ambiguous set choice: <value>

Not a valid set choice: <value>

The value must be chosen from the given list of values. Type the last or remaining characters required to uniquely identify the value, or select the item directly from the Set popup menu.

String length out of range: <value>

The string value that was entered is too short or too long. The Range popup menu identifies the valid string length.

No new values were entered

Values were not changed since the last time *Apply* was selected, or a null MODIFY operation is being attempted. The *Cancel* button should be used to exit.

Could not open help file

The applicable *.info* file is not available in the *access/help* directory, or does not have the correct permissions. Check the directory.

Managed Object Server Error Messages

Messages from the Managed Object Server are presented in one of the following forms, depending on the requested operation. The message that is displayed is specific to the actual error and is self-explanatory.

GET VALUES: <message>

MODIFY: <message>

ADD: <message>

DELETE: <message>

6.7 Using AccessMonitor

6.7.1 Introduction

AccessMonitor, a Graphical User Interface (GUI) application program, has been designed to monitor the status of Distributed7 system software for a specified signalling point, on a continued basis. It operates in so-called “asynchronous” mode, meaning that it is capable of detecting the changes in system software status on all involved hosts in an asynchronous manner. This mechanism is based on the internal event detection capabilities that are available in Distributed7.

AccessMonitor supports both stand-alone and distributed product configurations. The main drive behind AccessMonitor is to be able to monitor the status of SS7 protocol stack running on multiple hosts within a Distributed7 environment via a selected host. When executed under a distributed configuration, it also monitors the health of the kernel-level TCP/IP connections to all remote hosts on an on-going basis.

AccessMonitor is started by invoking the `AccessMonitor <sp#>` command for a specified signalling point from the command line. Upon start-up, AccessMonitor brings up a map of hosts that are currently configured and are accessible via the local host. As new host machines get configured and/or software on them gets started, AccessMonitor will resize the associated main window to display the status of all such hosts. At any time, users can obtain more detailed information by clicking on the layer buttons for the host of interest.

6.7.2 Main Window

This window gives general information about the current status of SS7 layers for each host and the TCP/IP connections among all active hosts in the system.

The main menu includes **Refresh**, **Help**, and **Exit** control buttons. The **Refresh** button is used to force AccessMonitor to retrieve all status information from scratch and repaint the screen. The **Help** button creates a new window that displays the manual page of AccessMonitor and the **Exit** button quits the program.

By default, each host figure includes all SS7 layers and each layer is represented either as *active* or *hidden*. Hidden layers are displayed in light gray. For active layers, green is used for fully operational layers, yellow is used for partially operational layers, and red is used if at least one critical software component is missing or experiencing trouble.

A system with two hosts for signalling point 0 can be seen in Figure 6-22 . In this figure, Distributed7 software is started on both hosts. Moreover, on host *neptun* MTP-L3 and SCCP layers are fully operational, the TCAP layer is partially operational, and at least one SS7 signalling board is configured.

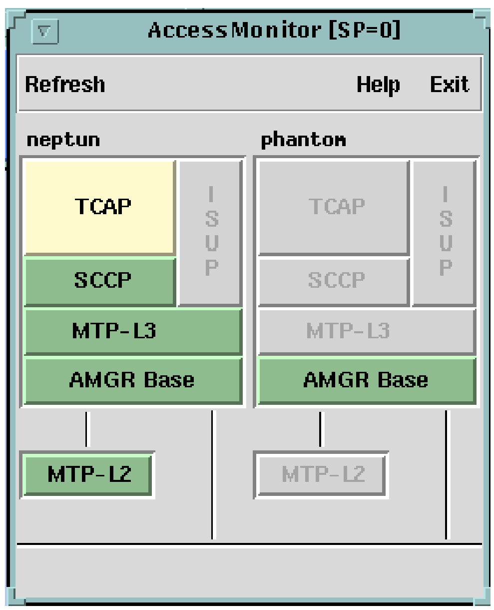


Figure 6-22: AccessMonitor Main Window

6.7.3 Subwindows

Clicking on each active layer button creates a new window (referred to as a subwindow) that displays further status information about daemon processes and STREAMS components. Once again, the status information is represented by colors. Green is used for normal status, yellow is used if a process is blocked or not yet started, and red is used if a previously running process does not exist anymore or is in trouble.

The MTP-L2 layer window gives information about the status of SS7 signalling boards or virtual SS7 boards configured on the specified host. (Figure 6-23)

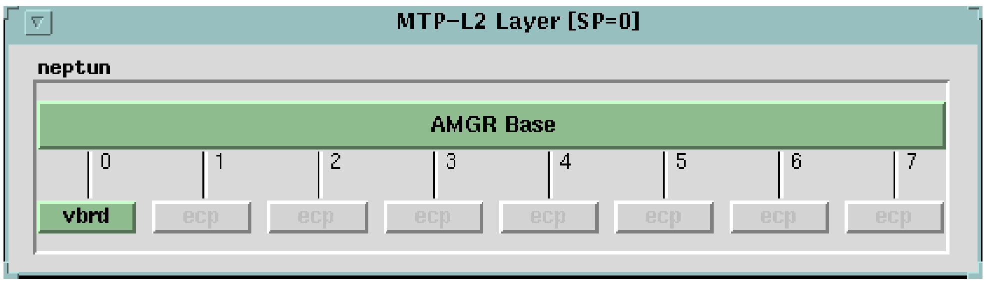


Figure 6-23: AccessMonitor MTP-L2 window

The DISTRIBUTED7 Base layer window monitors the status of daemon processes and STREAMS components that are essential for Distributed7 software to start up. Note that, in Figure 6-24 , all mandatory daemon processes except ***dsmd*** are running.

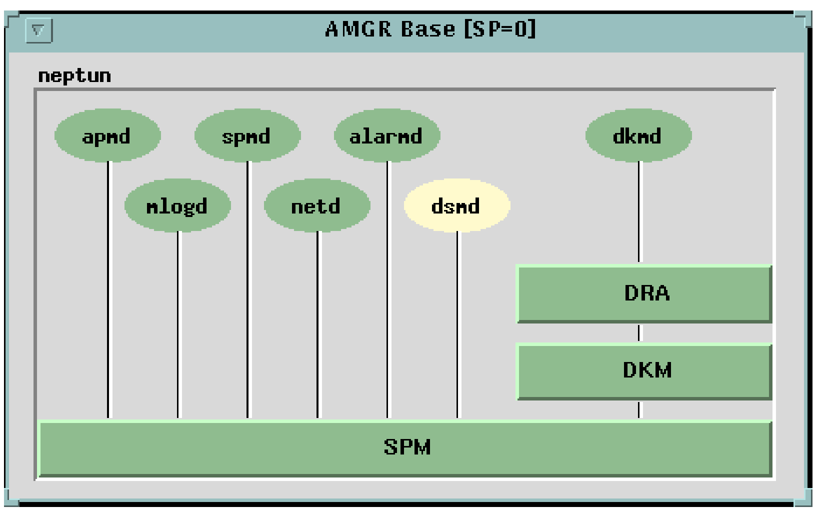


Figure 6-24: AccessMonitor DISTRIBUTED7 Base layer window

The window representation for other layers are the same. The only difference is that each of these layers monitors the daemon processes and STREAMS components it is interested in, the other processes and/or STREAMS components are hidden. In the following figures, the difference between SCCP layer window (Figure 6-25) and TCAP layer window (Figure 6-

26) can be seen (i.e., SCCP layer is active whereas the TCAP layer is not active at all in Figure 6-25 and partially active in Figure 6-26 .)

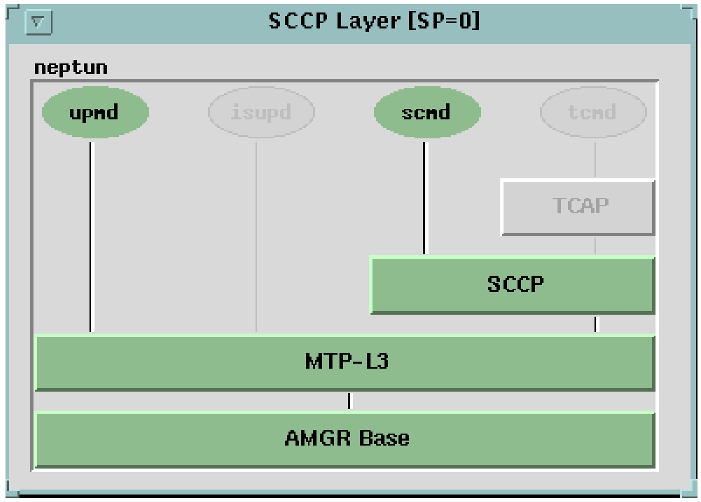


Figure 6-25: AccessMonitor SCCP layer window

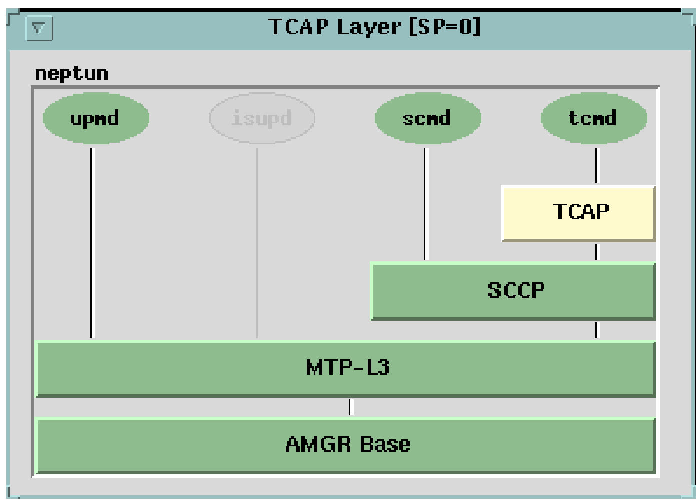


Figure 6-26: AccessMonitor TCAP layer window

6.7.3.1 TCP/IP Connections

The health of the TCP/IP connections among individual hosts is also monitored by AccessMonitor on a continued basis. There is a main LAN connection and all hosts are connected to it. If dual LAN is currently in use, two distinct LAN connections are displayed. If the heartbeat on a specified TCP/IP connection is okay, the connection is drawn as a straight line (Figure 6-22) with extensions made to the individual hosts involved. Otherwise, it is drawn as a dashed line. If Distributed7 is stopped on a host, the host figure is hidden and TCP/IP connection is cleared. For example, in Figure 6-27 , Distributed7 at host *phantom* is stopped and as a result of that, *phantom* is hidden and TCP/IP connection between *neptun* and *phantom* is cleared.

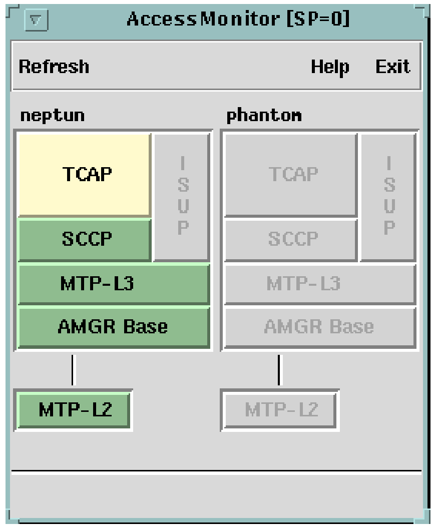


Figure 6-27: AccessMonitor TCP/IP Connections window

6.8 Using the Command File Navigator

6.8.1 Introduction

The Distributed7 Command File Navigator, a Graphical User Interface (GUI) application program, has been designed to simplify the usage of command-line utilities that require many command-line arguments. It is implemented using the TCL/TK script language. Input GUI files (.gui) are prepared for the following command-line utilities:

- Platform utilities
- Application Process Management (APM) utilities
- Distributed Shared Memory (DSM) utilities
- Distributed Kernel Memory (DKM) utilities
- Transaction Capabilities Application Part (TCAP) utilities

The Command File Navigator is started by invoking the **navigate.tcl** script from the command-line. Upon start-up, Command File Navigator brings up a session window (Figure 6-33), and a command file selection window (Figure 6-28). With the file selection window, the Navigator collects the command-line arguments for the selected command file, and executes the command using the parameter values collected.

6.8.2 Command File Selection

The user selects a command file with the help of a command file selection window (Figure 6-28 on page 6-68). This window displays the list of command files that are found under the current directory.

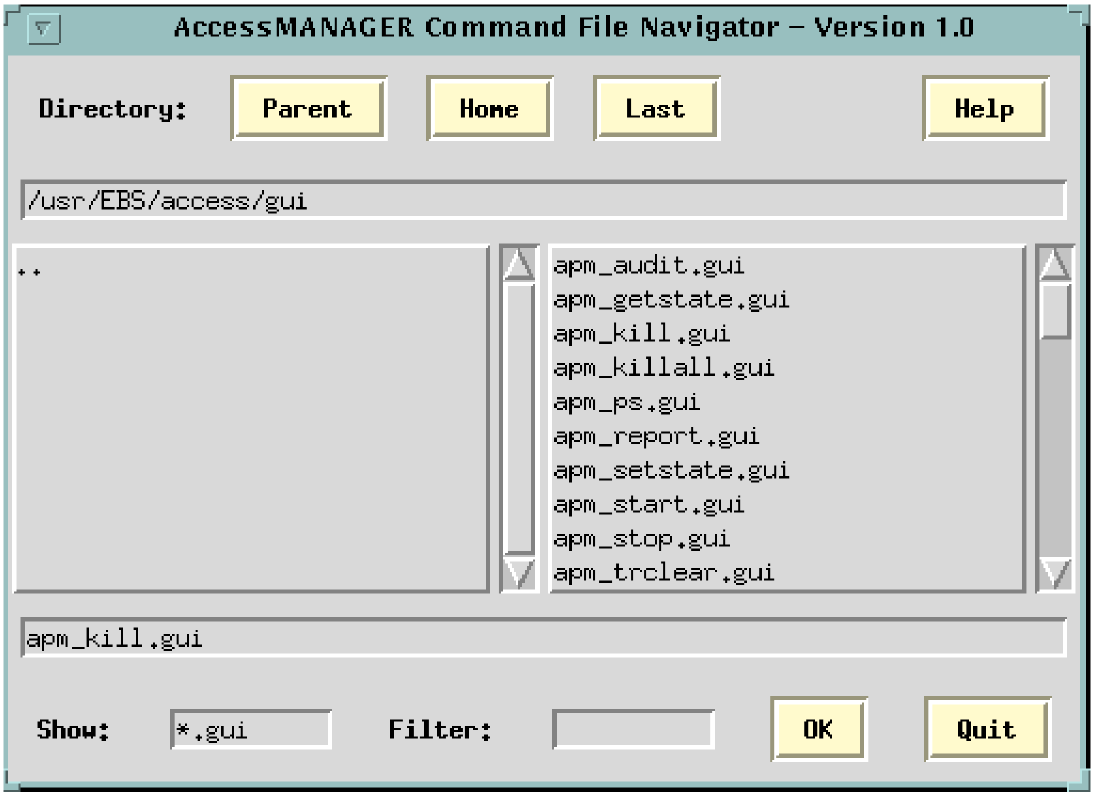


Figure 6-28: Command File Selection Window

The directory to be displayed can be chosen by typing the directory path in the top input line or by using the Directory buttons:

- *Parent*: Changes the current directory to the parent of it.
- *Home*: Changes the current directory to user home directory.
- *Last:* Changes the current directory to the last directory a command is executed.

The command file name can be chosen by selecting it from the list provided, or by typing it in the bottom input line. The input box labeled “Show:” is used to specify the file extension of the files to be listed. When a new string is typed in the “Show” input box, it creates a new list for the extension specified and replaces the old list with the new one. The input box labeled “Filter:” is used to specify a keyword—expected to be found within the information part of the input command files—for filtering the files. When a new keyword is typed for “Filter”, it searches all the files under the current directory and creates a new list by specifying the relative path for each file. Subsequently, it replaces the old list with the newly constructed list.

6.8.2.1 Control Buttons

The “OK” button is used to call the parameter collection window (Figure 6-29) for the selected command. This can also be done by double-clicking on the command file desired. At a given time, only one parameter collection window can exist and mouse interaction is only available on this window until operations for the selected command are cancelled.

The “Quit” button ends the program and “Help” button creates a new window which displays text that gives brief information about the Command File Navigator.

6.8.3 Parameter Collection

Parameter collection operation is done by filling in at least the mandatory entries among the activated ones on the parameter collection window (Figure 6-29 on page 6-70) that is called for the selected command. This window is created according to the parameter definitions taken from the input file for the selected command. Each window has a menu bar that includes an “Info” button to display the UNIX on-line manual page for the command. If the command requires interactive answers during execution, an additional menu button, “Session”, which enables the user to see the entire session, will be added. Brief information about the execution of the command is given in the “Action” section of the window. Control buttons for this window are “OK”, “Edit”, and “Cancel”.

6.8.3.1 Parameter Entries

The user is able to fill in only the activated parameters. To run a command, the user should fill in at least the mandatory parameters. Otherwise, a warning message will be displayed.

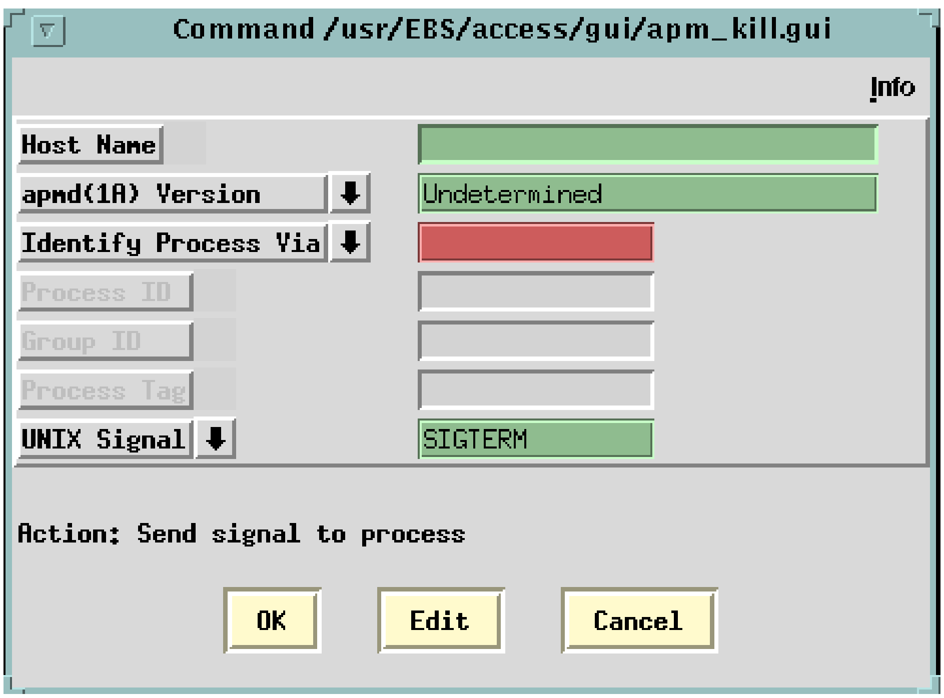


Figure 6-29: Parameter Collection Window

There are three types of parameter display mechanisms. The first one is to enter the values of the parameter by using a parameter entry space. The second method is to display the permissible values of the parameter using radio-buttons. The last one is to display parameters with two values as check-buttons.

In the first method, (Figure 6-29) a parameter entry may be in one of the two possible states. It may be enabled or disabled. An enabled entry has the following characteristics:

- The foreground color of the parameter label is black.

- The foreground color of the listing arrow is black.
- The background color of the parameter value entry space is red for mandatory ones and green for optional ones.

A disabled entry has the following characteristics:

- The foreground color of the parameter label is light grey.
- The foreground color of the listing arrow is light grey.
- The background color of the parameter value entry space is light grey.

If the range of the parameter is specified by giving the whole set of values in the input file, the entry space of the parameter is not able to be edited. Instead, a pop-up list can be activated by clicking on the arrow sign.

In the second method (Figure 6-30), the range of the parameter is restricted by a set of values given in the input file. Each value is represented as a radio-button and, at any given time, only one of them can be selected as the value of the parameter.

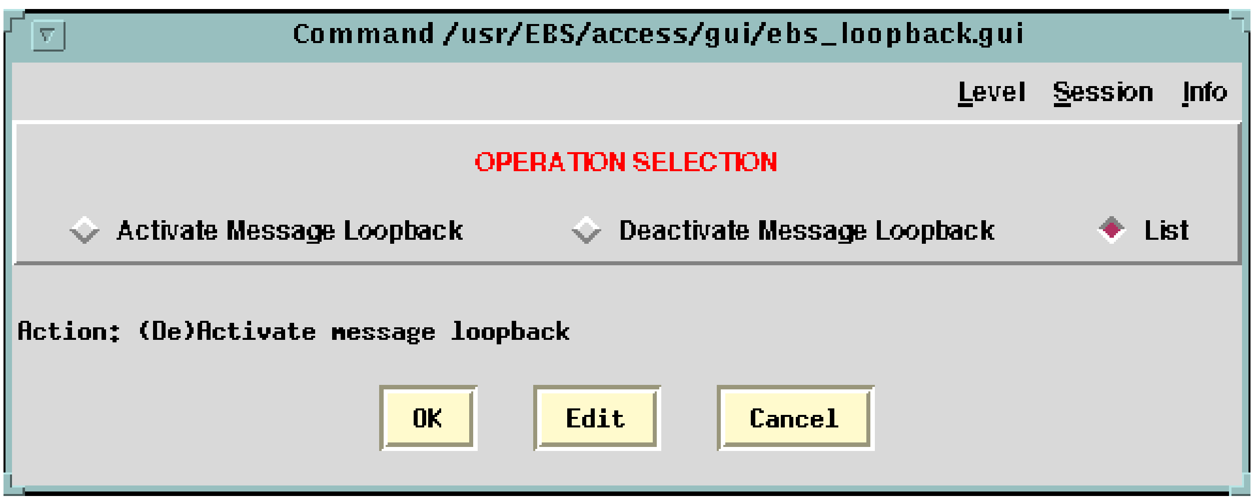


Figure 6-30: Alternate Parameter Collection Window

The last method is used for parameters with exactly two possible values. Here, the parameter is represented as a check-button and each of its values is bound to the corresponding state of the check-button. As can be seen in Figure 6-31 , the second parameter is represented as a selected check-button. The label of the check-button changes to a cross symbol when it is unselected.

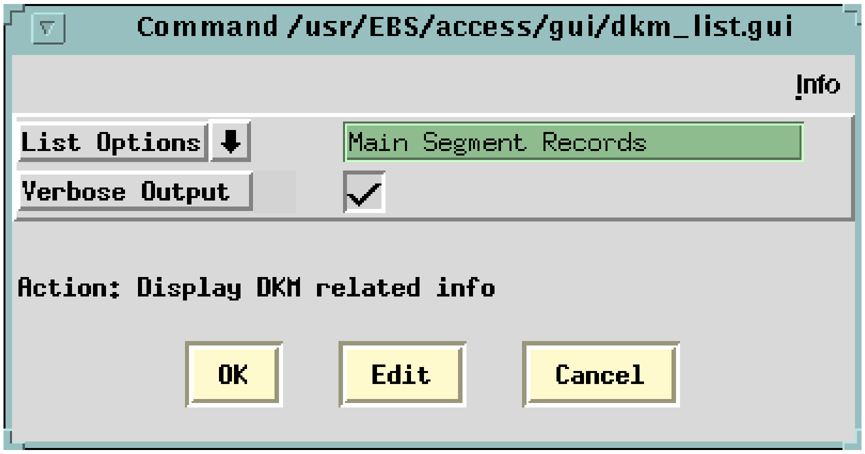


Figure 6-31: Second Alternate Parameter Collection Window

The range of some parameter values is specified in the ***$EBSHOME/access/gui/env.txt*** config file. If the user enters a value that does not suit the specified range, a warning message will be displayed. For the others, the user is responsible to enter the correct parameter values and the command is responsible for validating the parameters specified and print error messages if and when an invalid parameter is specified.

6.8.3.2 Control Buttons

There are three control buttons for each of the Parameter Collection Windows.

The "OK" button is used to run the command using the new command-line argument values.

The "Edit" button creates a new window (Figure 6-32) that displays the entire command line. In this Command Display window, the command is able to be edited, and the final version will be executed when the "OK" button is pressed. If the "Cancel" button is pressed, this window will be destroyed and the user will be able to make choices from the Parameter Collection window.

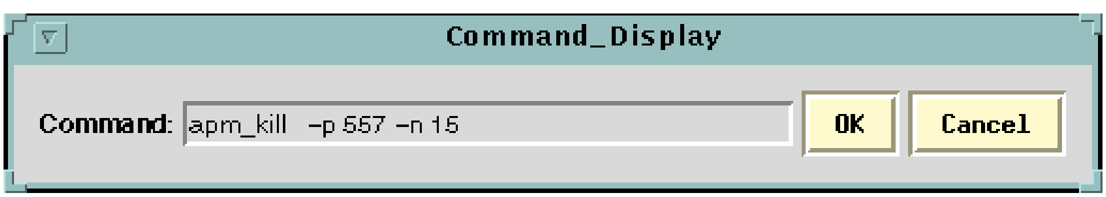


Figure 6-32: Command Display Window

The "Cancel" button destroys the parameter collection window and cancels all operations specified for the selected command.

If the command has no parameter, only the "OK" and "Cancel" control buttons and a text which provides a description of the command execution will be seen on the window.

6.8.4 Command Execution

The command is ready to run when the user fills in at least the all mandatory parameters. If some parameters are left empty, a warning message will be displayed and the user will be prompted to continue filling in the parameters. The results will be displayed on the session window (Figure 6-33). If the command requires some interactive inputs from the user during execution, the user will see the entire session, in addition to the results.

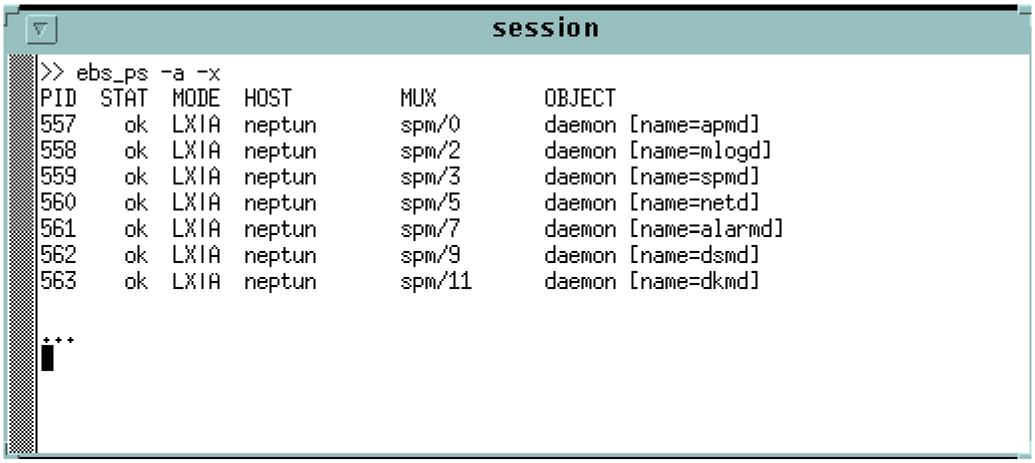


Figure 6-33: Session Window

6.9 Stand-alone Operation

This release of Distributed7 has been designed to extend the capabilities of the product to a distributed computing environment. While this is very functional, in some instances, it may be necessary or more desirable to run Distributed7 in a stand-alone mode. This release can be used for either distributed operations or stand-alone operations.

Distributed7 can be used "as is" for stand-alone operation, however, users of the stand-alone mode can further customize the product by modifying the contents of the *apmd* configuration file on their system as follows:

```
mlogd::failsafe::::::-1::::60:1::home:./bin/mlogd -x
spmd::failsafe::::::-1::::60:2::home:./bin/spmd
netd::failsafe::::::-1::::60:3::home:./bin/netd -s
alarmd::failsafe::::::-1::::60:4::home:./bin/alarmd
dsmd::failsafe::::::-1::::60:5::home:./bin/dsmd -s
dkmd::failsafe::::::-1::::60:6::home:./bin/dkmd -s -m dramod
```

Note that the ***-s*** command-line option specified for the *netd*, *dsmd*, and *dkmd* daemon processes indicate to the system that this host has been configured as a stand-alone machine; therefore, no effort should be spent by the *netd*, *dsmd*, and *dkmd* daemon processes to communicate with their counterparts on the remote host machines, if any. This is one certain way of optimizing the performance of the Distributed7 software on a host when running in the stand-alone mode.

The exact meaning of the stand-alone mode of operation for each of the above mentioned daemon processes is as follows:

- When invoked with the *-s* option, the *netd* daemon automatically assumes that no remote hosts are out there; therefore there can be no attempts to establish/remove TCP/IP connections through the local host. There is also no need to perform network clock synchronization.
- When invoked with the *-s* option, the *dsmd* daemon automatically assumes that no remote hosts are out there; therefore, a DSM segment is nothing more than an IPC shared memory segment allocated on the local host. Since no data replication is involved, the *dsmd* daemon has no need to interact with its counterparts on remote hosts when handling DSM related requests.
- When invoked with the *-s* option, the *dkmd* daemon automatically assumes that no remote hosts are out there; therefore, a DKM segment is nothing more than a kernel-resident memory segment allocated on the local host machine. Since no data replication is involved, DKM requests initiated by kernel threads can be serviced directly by the DKM multiplexer on the local host.

The upper layers of the Distributed7 system software may also feature command-line options that are intended to boost the performance of a particular subsystem when used in the stand-alone mode. The *isupd* daemon is one such example. The *tcm_tune* command-line utility is another example.

As a rule of thumb, users interested in exercising non-distribution related features of the Distributed7 product are recommended to consult with the manual pages of the daemon processes and/or command-line utilities involved first to find out whether these subsystems provide any additional support for stand-alone mode of operation.

6.10 Process Management

All Distributed7 system and application software should be started and stopped with the *apm_start* and *apm_stop* commands. These commands access a configuration file which specifies the order of process startup and other parameters associated with startup and management of the processes. The file is called:

$EBSHOME/access/RUN/config/PMGR/apmconfig

This file may need to be customized before the software is initially started. It is described in *Section 7.3.1 on page 7-38*. If a distributed network is being used, the file on each host may need to be customized.

The default contents of *apmconfig* specify start-up and termination procedures for SS7-specific system software for signalling points 0 through 7. These lines can be modified for your specific system setup. In addition, you will need to specify the rules for application processes that will be running on the host. Different states can be created to start and stop different groups of processes.

Once the configuration file changes are completed, the software can be started and the execution of all the processes will be managed through the configuration file and its defined states. Please read *Section 7.3.1* carefully.

The *apm_start* command puts the process manager (*apmd*) into the *init* state. The lines of the configuration file are executed based on this state. The configuration file is only executed when an event occurs or the state changes. The state can be changed from the command line with *apm_setstate*. The current state can be displayed with the *apm_getstate* command.

The configuration file can be modified while the system is running. After modifying the file, run the *apm_update* command. This command causes the process manager to re-read and run the configuration file based on the current state.

The process management commands are:

- apm_ps - report process status
- apm_getstate - retrieve apmd run state
- apm_setstate - manipulate apmd run state
- apm_kill - send signal to process and/or group of processes
- apm_killall - send signal to all processes
- apm_update - cause apmd to re-run apmconfig file because it has been changed

6.11 Configuration

This section describes the database configuration work that must be done for the Distributed7 system to function in the SS7 network. The configuration for each SP is held in configuration files located under the corresponding ***$EBSHOME/access/RUN#/DBfiles*** directory (where # is the SP number). Initially this directory will have an empty database because no default configuration exists. Once configuration information has been entered through MMI/MML or the AccessMOB GUI, it is automatically saved to the configuration files. Any time the Distributed7 software is to be started with an empty configuration, the files in the above directory must be deleted before the software is started.

The following subsections provide examples of how MMI/MML commands are used to configure the network shown in Figure 6-34. Each MMI/MML command is defined in *Chapter 9: Man-Machine Language Commands*.

Configuration can also be done through the Distributed7 AccessMOB GUI, which is described in *Using AccessMOB*. The same general steps that are described in this section would be used with the GUI.

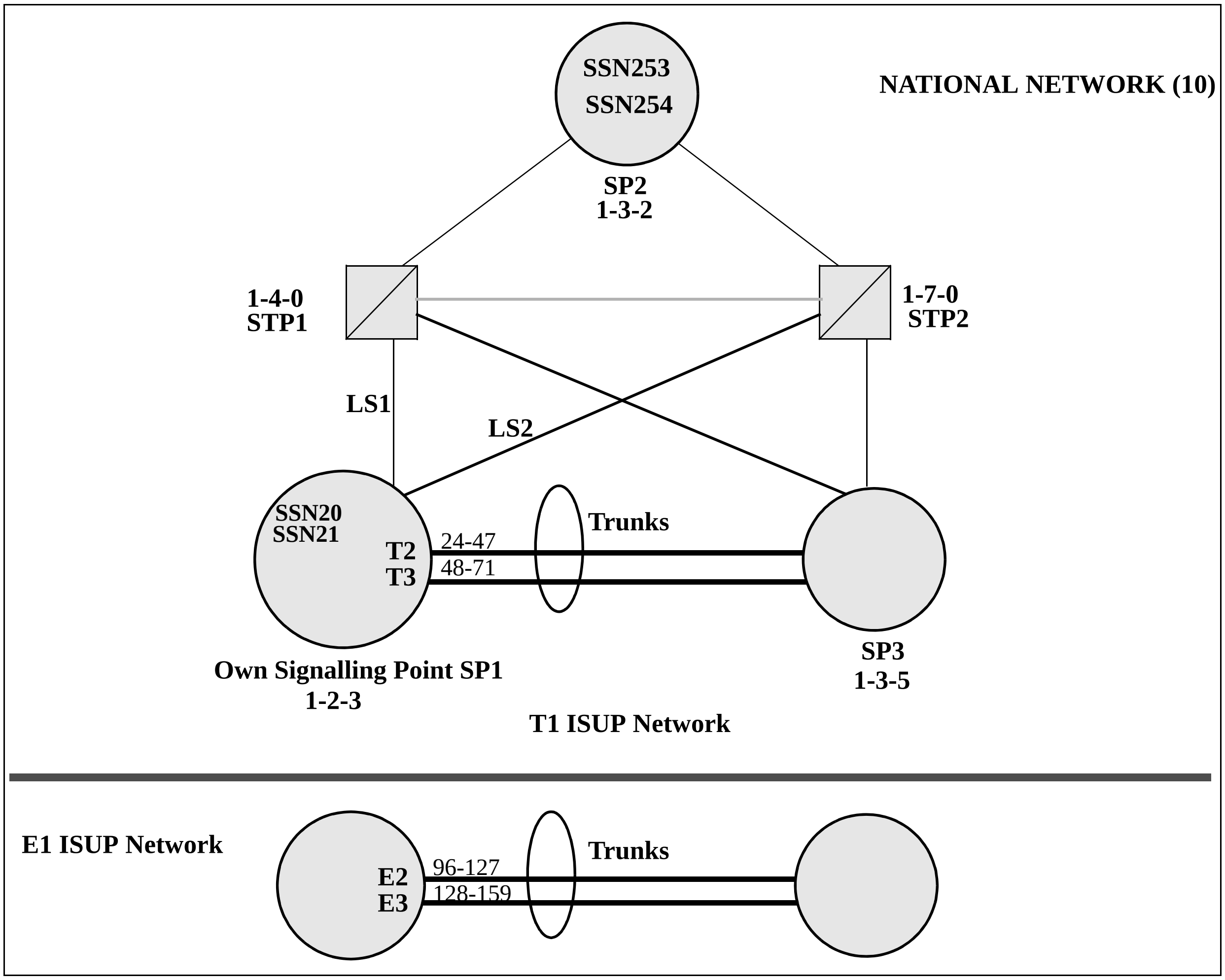


Figure 6-34: Sample Network

6.11.1 Configuring MTP

The MTP database must be configured prior to any other layer. Only by configuring MTP can the Distributed7 system be an SS7 node. The MTP layer identifies the pathways for transmitting SS7 messages to the other SS7 nodes.

6.11.1.1 Configuring Own Node

To become a node in the SS7 network, you must modify the MTP and SP managed object to provide your protocol and unique identity (point code). You must first add an instance of the MTP managed object in order to determine the protocol and pointcode size of the MTP/L3.

To configure the MTP MO, use the following command (*see Section 9.4.3 for details of the command*):

```
ADD-MTP:PROTOCOL=ANSI_92,PCSIZE=24_BIT;
```


This command will set the protocol of the MTP to ANSI (1992) and the pointcode size to 24 bits. It also creates an SP managed object entry with the default values. You must modify the SP MO in order to have a unique pointcode in the network. You must also modify the network type and node type as well.

To do this, use the following command (*see Section 9.4.15 for details of the command*):

```
MODIFY-SP:NAME=SP1,SPC=1-2-3,NI=NATIONAL;TYPE=SEP;
```

6.11.1.2 Adding Alias Point Code

Alias pint code is the second point code for the node. Use the following commands to create an alias point code (see Section 9.4.16 on page 9-76 for details of the command):

```
ADD-ALIAS: APC=2-3-4,OGCP=ON,INFLTR=APC,FLTRACT=UPU;
ADD-ALIAS: APC=3-4-5,OGPC=OFF,INFLTR=SPC, FLTRACT=UPU;
```

6.11.1.3 Adding Route Sets

First the RTSET MO has to be configured. In order to create route sets to the STPs and the remote SP, use the following commands:

```
ADD-RTSET:RTSET=RS_STP1,DPC=1-4-0;
ADD-RTSET:RTSET=RS_STP2,DPC=1-7-0;
ADD-RTSET:RTSET=RS_SP2,DPC=1-3-2;
```

6.11.1.4 Adding Link sets

After creating the route sets, you must create the linksets going to the adjacent STPs. Linksets are the signalling pathways to an adjacent destination. They consist of up to 16 links, which actually carry the signalling traffic. When connected to an STP pair, a linkset to each STP must be added. Linksets LS0 and LS1 of Figure 6-34 can be added with the following MML commands (*see Section 9.4.2 for details of the command*):

```
ADD-LSET:LSET=LS_STP1,DPC=1-4-0,LOADED=2,ACTIVE=2,TYPE=A;
ADD-LSET:LSET=LS_STP2,DPC=1-7-0,LOADED=2,ACTIVE=2,TYPE=A;
```

Note: *A linkset cannot be created without having a routset to the desired pointcode.*

In some networks, links may exist between two signalling end points (SEP) instead of between the SEP and the STPs. The following command would create a linkset between SP1 and SP3 (not shown in the figure):

```
ADD-LSET:LSET=LS_SEP3,DPC=1-3-5,LOADED=1,ACTIVE=1,TYPE=F;
```

6.11.1.5 Adding Routes

For all remote nodes, a route must be created and the corresponding linkset must be defined in this route set as the first route. To add the default routes for the adjacent STPs, the following commands are used (*see Section 9.4.5 for details of the command*):

```
ADD-ROUTE:RTSET=RS_STP1,LSET=LS_STP1;
ADD-ROUTE:RTSET=RS_STP2,LSET=LS_STP2;
```

If not provided, the priority of a route is set to the first available priority starting from the top priority value which is 0. If routes in a routeset have the same priority, these routes are called equal-priority routes (previously known as combined linksets) which will share the load over the linksets going to the same remote end.

For this sample configuration, let us define LS_STP1 and LS_STP2 as the equal priority routes for the remote destination. To do this, use the following MML command:

```
ADD-ROUTE:RTSET=RS_SP2,LSET=LS_STP1,PRIORITY=0;
ADD-ROUTE:RTSET=RS_SP2,LSET=LS_STP2,PRIORITY=0;
```

6.11.1.6 Adding Links

Links are the physical entities that carry the signalling traffic. The physical and agreed-upon logical parameters of each link must be identified to the system. Prior to adding a link instance, you must be sure that the corresponding SS7BOARD is configured and attached to the system. To do this, enter the following MML command to see if the SS7BOARD is set up correctly:

```
DISPLAY-SS7BOARD:;
```

If the board is correctly set up, add the links for each linkset with the following MML commands (*see Section 9.4.1 for details of the command*):

```
ADD-LINK:LINK=LS1L1,HOSTNAME=HOST_A,BOARDNM=sbs334,SLOT=2,PORT=1,
          LSET=LS_STP1,SLC=0,PRIORITY=0;
ADD-LINK:LINK=LS1L2,HOSTNAME=HOST_A,BOARDNM=sbs334,SLOT=2,PORT=2,
          LSET=LS_STP1,SLC=1,PRIORITY=1;
ADD-LINK:LINK=LS2L1,HOSTNAME=HOST_A,BOARDNM=sbs334,SLOT=1,PORT=1,
          LSET=LS_STP2,SLC=0,PRIORITY=0;
ADD-LINK:LINK=LS2L2,HOSTNAME=HOST_A,BOARDNM=sbs334,SLOT=1,PORT=2,
          LSET=LS_STP2,SLC=1,PRIORITY=1;
```

Section 6.11.1.4 showed an example of adding a linkset between two SEPs. To add a link to that linkset, the following command would be used:

```
ADD-LINK:LINK=LS3L1,HOSTNAME=HOST_A,BOARDNM=sbs332,SLOT=1,PORT=1,
          LSET=LS_SEP3,SLC=0,PRIORITY=0;
```

Links can be added to the system even if the HOSTNAME is not available. In this case MTP/L3 will save the ADD operation and apply the request to the system when the host becomes available.

6.11.1.7 Activating Links and/or Linksets

If the links are physically connected to the network or test equipment, they can be activated by the following commands (*see Section 9.4.11 and Section 9.4.12 for details of the commands*):

```
MODIFY-LINKSTAT:LINK=LS1L1,STATUS=SET_ACT;
MODIFY-LINKSTAT:LINK=LS1L2,STATUS=SET_ACT;
MODIFY-LINKSTAT:LINK=LS2L1,STATUS=SET_ACT;
MODIFY-LINKSTAT:LINK=LS2L2,STATUS=SET_ACT;
```

or

```
MODIFY-LSETSTAT:LSET=LS_STP1,STATUS=SET_ACT;
```



```
MODIFY-LSETSTAT:LSET=LS_STP2,STATUS=SET_ACT;
```

6.11.2 Configuring SCCP

The SCCP database only needs to be configured if SCCP or TCAP applications will be used to communicate with other SS7 nodes. For example, if the Distributed7 system will be an SCP or will be an SEP that queries and SCP, then the SCCP database must be configured for at least the SCCP nodes and subsystems.

6.11.2.1 SCCP Network Provisioning

SCCP functions are only available for the signalling points defined within the SCCP network. The SCCP network is a subset of the MTP network; therefore, each signalling point must be defined in the MTP network prior to its definition in the SCCP network. An error occurs when trying to add signalling points that do not have associated entries in the MTP database. The SCCP node, SP2, is added to the SCCP database with the following command (*see Section 9.5.6 on page 9-103 for details of the command*):

```
ADD-SNSP:SPC=1-3-2;
```

6.11.2.2 Subsystem Provisioning

SCCP users can only communicate with the subsystems provisioned in the SCCP network. In ITU WHITEBOOK networks, the SCCP management subsystem is created by SCCP and is always SSN=1. It gives the status of the remote SCCP. This management subsystem (SSN=1) cannot be modified.

In the figure, SP2 has two subsystems that will be accessed. These subsystems are added to the database with the following commands (*see Section 9.5.7 on page 9-105 for details of the command*):

```
ADD-SUBSYS:SPC=1-3-2,SSN=253;
ADD-SUBSYS:SPC=1-3-2,SSN=254;
```

6.11.2.3 Concerned SP Provisioning

You can provision concerned signalling points that are associated with the subsystems that will run on your node. A concerned signalling point will access the local subsystem and needs to know its status. SCCP management uses the concerned point code information to identify which signalling points must be notified of changes in the status of the local subsystems. A local subsystem cannot have its own signalling point code as a CPC.

For the figure, SP3 will be a concerned point code for the subsystems running on SP1. The CPC is identified in the SCCP database with the following commands (*see Section 9.5.1 on page 9-93 for details of the command*):

```
ADD-CPC:SPC=1-2-3,SSN=20,CPC=1-3-5;
ADD-CPC:SPC=1-2-3,SSN=21,CPC=1-3-5;
```

6.11.2.4 Global Title (GT) Database Provisioning

A global title is an alias address that can be translated to a point code, a point code and subsystem number, or a new global title. A global title address explicitly contains information that allows routing in the signalling network. The GT table is used to define the global title entries which will be translated in the GTENTRY table. The GT table is indexed by the key parameter GT and it is used to name translation types. A maximum of 131072 (256 translation type X 512 global title per translation type) instances can be entered.

```
ADD-GT:GT=GT1,GTIE=4,TRTYPE=0,ADDRINFO=12039251111;
```

Note: A wildcard cannot be used with the ADDRINFO parameter. To add all values, you must create 16 entries, each one with ADDRINFO equal to a different H' value, H'0 through H'f

6.11.2.5 Global Title Entry Table (GTENTRY) Provisioning

This table is used to introduce the global title translations to SCCP routing module (scrc). Two types of global titles are maintained for incoming and outgoing translation. The incoming global title is for translation on messages coming from the network and the outgoing one is for translation on messages going to the network. Cycles in global title translations are not allowed. A maximum of 131072 (256 translation type X 512 global title per translation type) instances can be entered.

ADD-GTENTRY:IO=INCOMING,GT=GT1,SPC=1-3-2,SSN=254;

6.11.2.6 Mated Subsystem Provisioning

SCCP subsystems can be mated with each other as pairs to provide redundancy in the case of failure. The signalling points, the subsystems, and their concerned point codes have to be designed prior to mating two subsystems.

As an example, the following command mates subsystem 21 and 253 (*see Section 9.5.4 on page 9-99 for details of the command*):

```
ADD-MATE:SPC=1-2-3,SSN=21,MSPC=1-3-2,MSSN=253;
```

6.11.3 Configuring ISUP

The ISUP database only needs to be configured when a call processing application is associated with the Distributed7 ISUP module for control of circuit-switched communications.

6.11.3.1 Configuring Remote ISUP Node

All nodes that your node will be exchanging ISUP messages with must be in the ISUP database. These nodes should already be in the MTP database. When adding nodes, you must associate the destination point code with a point code index number (PCNO). The PCNO is arbitrary and allows the administrator to identify a remote node by a simple number instead of the entire point code. An optional parameter, CICCONTROL, can be specified to identify which node will control which CICs of the trunks between them.

From the figure, trunks exist to SP3. Since this is the first ISUP node to be added, it will be assigned a point code number (PCNO) of 1.To add the node, enter the following command (*see Section 9.6.3 for details of the command*):

```
ADD-ISUPNODE:PCNO=1,DPC=1-3-5;
```

When the new node is introduced to the system, the default accessibility status is set to ACCESSIBLE (as would be seen in the DISPLAY-ISUPNODE output). The node is set to this initial status even if the links or routes to that destination are down when the node was created. The status is updated with the accurate value after further MTP_PAUSE and MTP_RESUME messages (link ups and downs) occur with the new node existing in the system.

6.11.3.2 Adding ISUP Circuit Groups

The trunk groups between two ISUP nodes that will be used must be identified. To add a circuit group, you must have the trunk group ID and a circuit group ID. You must also specify the number of circuits in the group. In ANSI releases, you may specify whether Software Carrier Group Alarm (SCGA) protection is on or off.

The trunk group ID (TRNKGRPID) is the designated trunk group number in the switch. Trunk group IDs are unique across the switch. They are used to identify groups of trunks (circuits) on the local end. A circuit identification code (CIC) is assigned to each trunk and is known at both ends of the trunk. CICs are unique between two SEPs. More than one trunk in a switch may have the same CIC, but each trunk that does have the same CIC must go to a different destination. CICs are composed of a circuit group ID and a circuit ID. For the example of this section, trunk group 3 of the figure will be used (T3/E3).

The circuit group ID (GRPID) is determined from the CIC of a trunk in the desired trunk group. It is determined differently for T1s than for E1s.

For a T1 (used in ANSI releases), the circuit group ID equals the modulus 24 of any CIC on the desired trunk group. This means you can pick CIC 71 from group 3 (T3), divide it by 24 and get the result of 2. (Any number from 48 to 71 can be used.) The circuit group ID equals 2.

For an E1 (used in CCITT releases), the circuit group ID equals the binary value of the seven most-significant bits of the CIC of the desired trunk group. This means you can use any binary CIC from group 3 (E3), such as 150. The binary value of CIC 150 is 000010010110. The binary value of the seven most-significant bits (0000100) is 4, which is also the circuit group ID.

To add trunk group 3 as an ISUP circuit group, the command is (*see Section 9.6.2 for details of the command*):

T1: `ADD-ISUPCGRP:PCNO=1,GRPID=2,CCTNUM=24,TRNKGRPID=3,SCGA=ON;`

E1: `ADD-ISUPCGRP:PCNO=1,GRPID=4,CCTNUM=32,TRNKGRPID=3;`

6.11.3.3 Adding ISUP Circuits

The circuits of the circuit group that will be available must be identified to ISUP in the ISUP database. ISUP does not assume all the circuits will be used. Individual circuits or a range of circuits may be added. Circuits are identified differently for T1 and E1 networks. Please see the MML command (*Section 9.6.1*) for more information on the command.

For a T1, the circuit number equals the remainder of the modulus 24 of the CIC. Generally, the first circuit will be 0 and the last 23. For example, in trunk group 3, CIC 71 divided by 24 has a remainder of 23, which is the circuit number. See *Section 9.6.1* for valid ranges of protocol variants.

For an E1, the circuit number equals the binary value of the five least-significant bits of the CIC. For CIC 150, the binary value of the five least-significant bits (10110) is 22 which is also the circuit number. While there are 32 circuits in an E1, from 0 to 31, circuit 0 is always reserved for synchronization and will not be used for voice traffic. In addition, one of the remaining circuits is considered a D-channel (signalling channel). This circuit cannot be included in the ISUP group. The rest of the circuits, a total of 30, are considered B-channels or voice channels.

To add single circuits, commands similar to the ones below can be used:

T1: `ADD-ISUPCCT:PCNO=1,GRPID=2,CCTNUM=23;`

E1: `ADD-ISUPCCT:PCNO=1,GRPID=4,CCTNUM=22;`

To add a sequential range of circuits, the range parameter is used. For a T1, the following command adds all circuits, numbered from 0 to 23.

T1: `ADD-ISUPCCT:PCNO=1,GRPID=2,CCTNUM=0,RANGE=24;`

For an E1, the following command adds 30 circuits, numbered from 1 to 30. The command assumes the D-channel is on circuit 31 or that no signalling link is in the group.

E1: `ADD-ISUPCCT:PCNO=1,GRPID=4,CCTNUM=1,RANGE=30;`

For an E1, if a signalling link (D-channel) is on circuit 13, you would need to enter the following commands to make all the other trunks into ISUP circuits:

```
ADD-ISUPCCT:PCNO=1,GRPID=4,CCTNUM=1,RANGE=12;
ADD-ISUPCCT:PCNO=1,GRPID=4,CCTNUM=14,RANGE=18;
```

Circuits 1 - 12 and 14 - 31 (a total of 30 circuits) will be added to the circuit group. Circuit 0 is excluded because it is used for synchronization. If circuit 0 was inadvertently added to the group, the switches on both sides automatically locally block it from any possible voice traffic.

6.11.4 Changing Initial Configuration

6.11.4.1 Modifying

Most MODIFY commands are used to change specific parameters that were set with an ADD command or to set certain parameters that were not available with an ADD command.

Signalling Point

Any of the parameters defined initially for the signalling point (MTP and SP managed objects) (Section 6.11.1.1) can be modified at a later time. If you modify the RESTART parameter of the SP MO, the MTP/L3 will perform the MTP-SP Restart procedure which will restart the node.

Alias Point Code

The Modify-alias MML command allows you to change the outgoing poinr code, incoming message filter, and filter action. See Section 9.4.16 on page 9-76 for the command.

Linksets

The MODIFY-LSET MML command allows you to change the number of active and loaded links in the linkset. See *Section 9.4.2 on page 9-42* for the command.

Linkset Status

The MODIFY-LSETSTAT (*Section 9.4.12 on page 9-69*) MML command allows you to change the administration state of a linkset between active and inactive. The automatic link activation setting can also be changed with this command.

Links

The MODIFY-LINK MML command allows you to change the priority of the link in a linkset (if there are more than one). See *Section 9.4.1 on page 9-37* for the command.

Link Status

The MODIFY-LINKSTAT (*Section 9.4.12 on page 9-69*) MML command allows you to change the status of the link to activate, deactivate, inhibit, or uninhibit.

ISUP Nodes

The MODIFY-ISUPNODE MML command allows you to change the destination point code of a point code index number (PCNO) or the type of control the node has on CICs. See *Section 9.6.3 on page 9-117* for the command.

ISUP Circuits

The MODIFY-ISUPCCT MML command allows you to change the operation state of a circuit or group of circuits. The command is described in *Section 9.6.1 on page 9-110*. Valid operation states are:

- reset a single circuit (RSC)
- block a circuit (BLO)
- unblock a circuit (UBL)
- reset a circuit group (group reset - GRS)
- hardware circuit group block (HCGB)
  In ANSI ISUP, this is a *block with immediate release*.
- hardware circuit group unblock (HCGU)
- maintenance circuit group block (MCGB)
  In ANSI ISUP, this is a *block with immediate release*.
- maintenance circuit group unblock (MCGU)
- stop all supervision events on a circuit

6.11.4.2 Deleting

The configuration can be changed by deleting instances of managed objects. However, most instances have dependencies so other commands are also needed. In general, when deleting any instance from the database, commands similar to those used to create the instance are required, but in the reverse order. The following examples are provided with this order.

ISUP Circuits

To delete an ISUP circuit or a range of circuits from a group, use the DELETE-ISUPCCT command similar to the following examples.

To delete all circuits from 0 to 31:

```
DELETE-ISUPCCT:PCNO=1,GRPID=4,CCTNUM=0,RANGE=32;
```

To delete circuit 31:

```
DELETE-ISUPCCT:PCNO=1,GRPID=4,CCTNUM=31;
```

ISUP Circuit Groups

To delete an ISUP circuit group, all circuits in the group must be deleted first. For example:

```
DELETE-ISUPCCT:PCNO=1,GRPID=4,CCTNUM=0,RANGE=32;
DELETE-ISUPCGRP:PCNO=1,GRPID=4;
```

ISUP Nodes

To delete a ISUP node, all circuits and circuit groups must be deleted first. For example:

```
DELETE-ISUPCCT:PCNO=1,GRPID=4,CCTNUM=0,RANGE=32;
DELETE-ISUPCGRP:PCNO=1,GRPID=4;
DELETE-ISUPNODE:PCNO=1;
```

Global Title Database Entries

Global title entries may be removed from the database. The command is described in *Section 9.5.3 on page 9-97*. A wildcard may be used for the ADDRINFO parameter to specify all addresses for a particular table, GTIE, and TRTYPE. The following is a sample command deleting one address:

```
DELETE-GTENTRY:IO=OUTGOING,GTIE=4,TRTYPE=0,ADDRINFO=H'12039251111;
```

Mated SCCP Subsystems

Two subsystems may be *unmated*. The subsystems remain functional, they simply will no longer be mates to each other. As an example, the following command ends the mate relationship between subsystem 21 and 253 (*see Section 9.5.4 on page 9-99 for details of the command*):

```
DELETE-MATE:SPC=1-2-3,SSN=21,MSPC=1-3-2,MSSN=253;
```

Concerned SPs

If you no longer want SCCP management to automatically notify a point code of the status of a subsystem, you remove the concerned point code (CPC) entry for it. The point code can still access the subsystem; nothing else changes. The command is described in *Section 9.5.1 on page 9-93*. The following command identifies that SP3 will no longer be a concerned point code for subsystem 21 (it stays concerned for SSN 20):

```
DELETE-CPC:SPC=1-2-3,SSN=21,CPC=1-3-5;
```

Subsystems

When a subsystem on a particular node will no longer be accessed, it can be removed from the local SCCP database. Subsystem 1 cannot be deleted because it is the SCCP management subsystem. If a subsystem is defined as a mate in the database, the mate definition must be deleted first. Any concerned point code definitions or global title entries related to the SSN must also be deleted from the database first. The command is defined in *Section 9.5.7 on page 9-105*.

As an example, subsystem number 253 of Figure 6-34 will be deleted:

```
DELETE-MATE:SPC=1-2-3,SSN=21,MSPC=1-3-2,MSSN=253;
DELETE-SUBSYS:SPC=1-3-2,SSN=253;
```

SCCP Nodes

An SCCP node entry in the database may be deleted if the node will no longer be accessed by your node. You must first delete all subsystems, including global title entries, related mates, and CPCs. The command is described in *Section 9.5.6 on page 9-103*.

The following example deletes SP2 from the SCCP portion of the database:

```
DELETE-MATE:SPC=1-2-3,SSN=21,MSPC=1-3-2,MSSN=253;
DELETE-SUBSYS:SPC=1-3-2,SSN=253;
DELETE-SUBSYS:SPC=1-3-2,SSN=254;
DELETE-SNSP:SPC=1-3-2;
```

Links

To delete a link, it must be deactivated. For example:

```
MODIFY-LINKSTAT:LINK=LS1LNK1,STATUS=CLR_ACT;
DELETE-LINK:LINK=LS1L1;
DELETE-LINK:LINK=LS1L2;
DELETE-LINK:LINK=LS2L1;
DELETE-LINK:LINK=LS2L2;
```

Routes

To delete a route from a route set, use the DELETE-ROUTE command as in the example:

```
DELETE-ROUTE:RTSET=RS_STP1,LSET=LS_STP1;
DELETE-ROUTE:RTSET=RS_STP2,LSET=LS_STP2;
DELETE-ROUTE:RTSET=RS_SP2,LSET=LS_STP1;
DELETE-ROUTE:RTSET=RS_SP2,LSET=LS_STP2;
```

Linksets

To delete an entire linkset, it must be deactivated and its links must be deleted, and all routes using this linkset must be deleted. In addition, its route set must also be deleted. If a route set is defined for a destination point code of a ISUP node, it must be deleted first (route set names and information can be retrieved with DISPLAY-RTSET). For example, to remove a linkset to an STP, the commands are:

```
MODIFY-LSETSTAT:LSET=LS_STP1,STATUS=CLR_ACT;
DELETE-ROUTE:RTSET=RS_STP1,LSET=LS_STP1;
DELETE-RTSET:RTSET=RS_STP1;
DELETE-ROUTE:RTSET=TO_CLUS,LSET=CLS1;
DELETE-LSET:LSET=LS_STP1;
DELETE-LSET:LSET=LS_STP2;
```

Note: There is no longer a route to SP2 or all other nodes having point codes starting with 1-3 until another route is defined.

Route Sets

To delete a route set, it must not have a destination point code for an active ISUP node (route set names and information can be retrieved with DISPLAY-RTSET). Also, all routes in this route set must be deleted, and no linksets going to the destination where the routeset is defined can exist. When deleting route sets for ISUP nodes, the ISUP node must be deleted first, which involves deleting the circuits and circuit groups.

The basic commands are:

```
DELETE-RTSET:RTSET=RS_STP1;
DELETE-RTSET:RTSET=RS_SP2;
```

For example, to remove a route set to a STP, the commands are:

```
MODIFY-LSETSTAT:LSET=LS1,STATUS=CLT_ACT;
DELETE-RTSET:RTSET=LS1STP;
```

To delete a route set to SP3, which happens to be an ISUP node, the commands are:

```
DELETE-ISUPCCT:PCNO=1,GRPID=4,CCTNUM=0,RANGE=32;
```



```
DELETE-ISUPCGRP:PCNO=1,GRPID=4;
DELETE-ISUPNODE:PCNO=1;
DELETE-RTSET:RTSET=SEPLS3;
```

If a route set is for a SCCP node, the node must be removed from the SCCP database first.

Note*: The default route set created automatically after adding a linkset can not be deleted with this command. You must use DELETE-LSET.*

Alias Point Code

Use the DELETE-ALIAS command to delete an alias point code, as in the following example:

DELETE-ALIAS: APC=2-3-4;

Signalling Point and MTP Layer

If the user wants to remove the MTP layer and reconfigure it, all links, linksets, routes and routesets must be deleted. After that, the MTP instance can be deleted. Use the following command to delete the MTP instance:

```
DELETE-MTP:SPNO=0;
```

6.11.5 Displaying the Configuration

To see current settings of managed objects, the DISPLAY commands should be used as shown in *Chapter 9: Man-Machine Language Commands*.

6.11.6 Changing Timers

All timers have default settings which do not need to be changed, but can be. MTP Level 2 timers are changed with the MODIFY-L2TIMER MML command (*Section 9.4.8 on page 9-58*). MTP Level 3 timers are changed with the MODIFY-L3TIMER MML command (*Section 9.4.9 on page 9-60*). ISUP-related timers are changed with the MODIFY-ISUPTMR MML command (*Section 9.6.5 on page 9-123*). These commands and the valid ranges for timers are given in *Chapter 9: Man-Machine Language Commands*.

6.11.7 Changing MTP Congestion Settings

Congestion settings for the links are changed with the MODIFY-L2FLOW MMI/MML command, which is described in *Section 9.4.7 on page 9-55*. Congestion occurs when incoming MSUs begin accumulating in the queue for MTP processing. Default values are provided and normally do not need to be changed. They are shown in Table 9-10 on page 9-55. The flow control settings are the same for all links on the node.

In networks with multiple congestion and priority levels, values for congestion onset, congestion abatement, discard onset, and discard abatement can be set for each of 3 different flow control levels. In standard CCITT networks, there is only one flow control level so only the congestion onset and abatement values may be set.

The congestion onset value specifies the number of waiting MSUs that should trigger MTP to send out Status Indication messages to other nodes indicating the link is in a congested state. MTP will continue to send out the indication messages until the number of MSUs in the queue goes below the congestion abatement value. The abatement value should be less than the congestion onset value.

The discard onset value specifies the number of waiting MSUs that should trigger MTP to start discarding MSUs that are received on the link. The value should be higher than the congestion onset value. MTP will continue to discard MSUs until the number of MSUs in the queue goes below the discard abatement value. The abatement value should be less than the discard onset value.

Please note that the Congestion Settings are defined for each SS7 link. The user can modify flow control values for a specific link. If the user wants to set a congestion value for all links, the corresponding command must be issued for each link.

6.11.8 Changing General ISUP Settings

The MODIFY-ISUP command allows you to change several general ISUP behavioral settings on a node. For any version of ISUP that is running on the node, this command will allow you to change to a different ISUP variant. The command and valid variants are described in *Section 9.6.4 on page 9-120*. It will also allow you to change the recovery mode of ISUP.

Important: *When a variant is changed, all the ISUPNODE, ISUPCGRP, ISUPCCT, and ISUPTMR configurations are erased. These objects must be configured again.*

For CCITT versions of ISUP, the command also allows you to change whether maintenance indication messages will be sent to the maintenance module and whether ISUP will limit outgoing calls to a congested destination.

6.11.9 Configuring the Display of Alarms

All alarms are displayed to the console by default. With the MODIFY-ALARM MMI/MML command (*Section 9.7.3 on page 9-133*), the display of alarms can be turned off for all or some alarms. Regardless of the display settings, all alarms will always be logged to the alarm log files in ***$EBSHOME/access/RUN/alarmlog*** (*see the Installation and Maintenance Manual for more information on these files*). The following subsections describe the different levels of display.

6.11.9.1 Turn Display of Alarms On or Off

The MODIFY-ALARM:DISPLAY=OFF; command turns the display of all alarms off, both at the HICOM service terminal and at a UNIX console connected to the Distributed7 unit. No alarms will be displayed, regardless of other settings. The MODIFY-ALARM:DISPLAY=ON; command turns the display on. The types of alarms that are displayed depend on the settings described in Section 6.11.9.2 and Section 6.11.9.3.

6.11.9.2 Display Only Alarms with Certain Severities

The alarm display can be filtered to only send alarms of a certain severity and higher to the console or to an external alarm interface can be limited by their severity. By setting a minimum severity threshold, only those alarms with that severity or a higher one will be sent by the alarm process. A severity can be set for all Distributed7 alarms, or different minimum severities can be set for each alarm group. The severities to be used are INFO, MINOR, MAJOR, CRITICAL or FATAL.

The thresholds are set separately for the console and the external alarm interface. The MODIFY-ALARM command is used to set these thresholds for all alarms on the system while the MODIFY-ALMGRP command is used to set thresholds for a particular alarm group. If the overall threshold set with the MODIFY-ALARM is higher than the threshold set for a group using MODIFY-ALMGRP, then the overall setting will be used. The alarm groups are:

- ISUP- ISUP management (ANSI and ITU/CCITT)
- ISUPMOD - ISUP management
- NIMOD - TCP/IP connection management
- OMAP - operation, maintenance, and administration part
- SCCP - service connection control part management
- SPM - signalling point (SP) management
- TCAP - TCAP transaction layer management
- TCMOD - TCAP over TCP/IP connection management
- TRMOD - translation module
- UPM - MTP user part management
- DKM - distributed kernel memory management
- ETMOD - ethernet test module
- ISUPMOD - ISUP module

- APM - application process management
- MTPL1 - MTP Level 1
- MTPL2 - MTP Level 2

The thresholds for display to the console are set using the CONS_THRS parameter of the MODIFY-ALARM or MODIFY-ALMGRP command. The thresholds for alarms sent to the external alarm interface are set using the USER_THRS parameter of each command. An external alarm interface is a user-developed process that is designed to receive and process alarm messages that are sent by the system alarm process. Figure 6-35 depicts the relationship of the processes and use of these parameters.

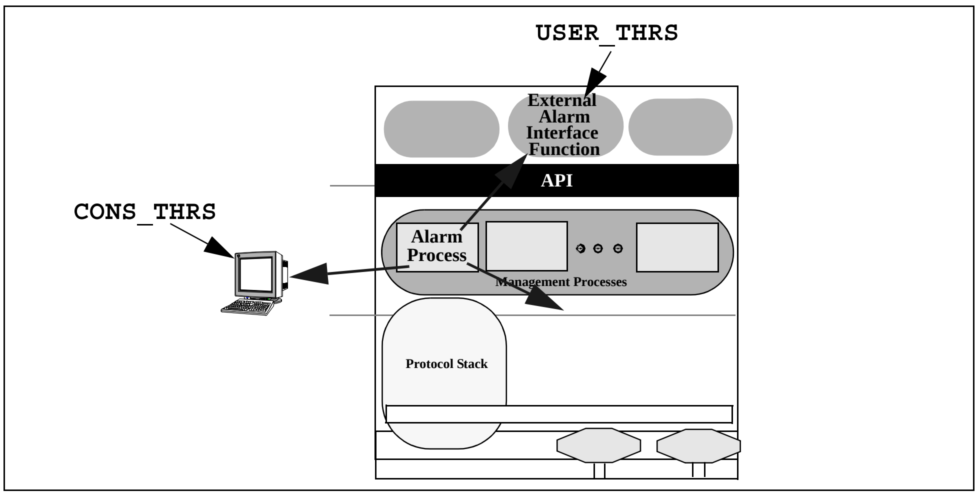


Figure 6-35: Alarm Display Thresholds

For example, to set a minimum severity of MINOR for <u>all</u> alarms to be displayed on the console, use the command:

```
MODIFY-ALARM:CONS_THRS=INFO;
```

To set a minimum severity of MINOR for <u>all</u> alarms to be sent to the external alarm interface, use the command:

```
MODIFY-ALARM:USER_THRS=INFO;
```

To set a minimum severity of MAJOR for the ISUP alarm group to be displayed to the console, use the command:

```
MODIFY-ALMGRP:GROUP=ISUP,CONS_THRS=MINOR;
```

To set a minimum severity of MAJOR for the ISUP alarm group for the external alarm interface, use the command:

```
MODIFY-ALMGRP:GROUP=ISUP,USER_THRS=MINOR;
```

6.11.9.3 Limiting the Display of Repeated Alarms

An alarm may repeat several times without any other alarms occurring. For this case, a threshold is available so that the alarm is displayed with a message indicating the number of times it was repeated instead of displaying the alarm each time it occurs. The default for this threshold is 3.

As an example, when the threshold is 3, the first alarm is displayed. If the same alarm occurs three more times, consecutively, a message will display saying the alarm was repeated 3 times.

The command to change this setting to 5 is:

```
MODIFY-ALARM:REPEAT=5;
```

An example is shown in the listing for MODIFY-ALARM in *Section 9.7.3 on page 9-133*.

6.11.10 Redundant LAN Configurations

The Distributed7 system software provides built-in support for dual-LAN configurations. In a dual-LAN configuration, each host is connected to the others by two physically separated LAN connections. The standard MMIs can be used to select between single and dual-LAN configurations.

***NOTE**: To upgrade from an existing single-LAN configuration, or to downgrade to a single-LAN configuration, see Single-LAN to Dual-LAN Configuration and Dual-LAN to Single-LAN Configuration below.*

From an implementation point of view, a dual-LAN configuration requires a total of two TCP/IP connections to be established between any two specified pair of hosts, as depicted in Figure 6-36. These connections provide alternate communication paths between the corresponding hosts in the case of LAN failures.

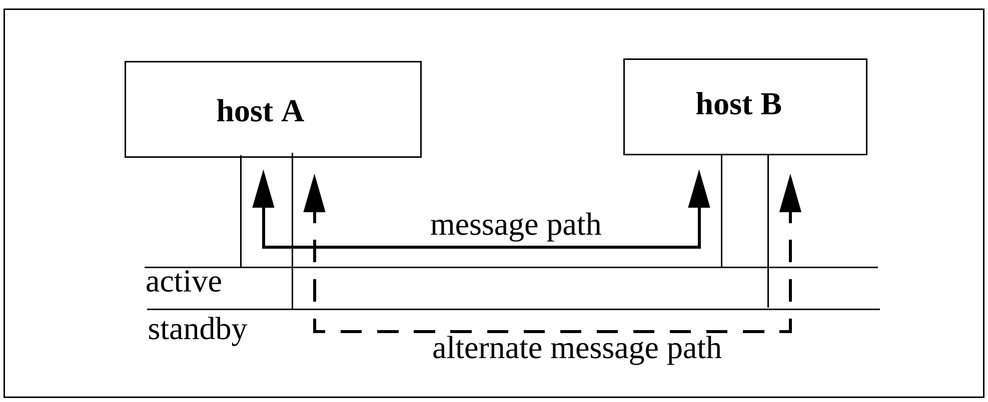


Figure 6-36: Message Exchange in a Dual-LAN Configuration

Multiple TCP/IP connections between two hosts always operate in *active/standby* mode. The *active* connection is used to transport messages. The *standby* connection is ready to take over if a failure occurs on the *active* connection. This is critical for ensuring orderly delivery of inter-host messages between any two hosts. The switchover from *active* to *standby* will occur under the following circumstances:

- When the *active* TCP/IP connection is removed.
- When the Distributed7 heartbeat check mechanism fails on the *active* TCP/IP connection.

The *active*/*standby* mode of operation of a particular TCP/IP connection is only meaningful to the Distributed7 host. The mode specifies whether the Distributed7 host can send inter-host messages to the remote host across that connection. It can only send messages on the *active* connection. If the remote host sends messages to the Distributed7 host across a TCP/IP connection that is marked as *standby*, the Distributed7 system will process these messages as if they were sent across an *active* connection. However, *heartbeat response* messages are always returned over the same TCP/IP connection that the corresponding *heartbeat request* messages have been received through.

EXAMPLE

Configuring HOST MO to connect host-A, host-B, host-C, and host-D to each other under a distributed environment where Dual-LAN is in use (i.e., where the NTWK MO on all hosts is modified to run in the Dual-LAN mode). Alternate host names are assumed to be AltHost-A, AltHost-B, AltHost-C, and AltHost-D, respectively.

1. Introducing other hosts (i.e., host-B, host-C, and host-D) to host-A under a Dual-LAN configuration

MODIFY-NTWK:hostname=Host-A,dualhost=AltHost-A;

ADD-HOST:hostname=host-A,rmthost=host-B,alias=AltHost-B,conf=ON;

ADD-HOST:hostname=host-A,rmthost=host-C,alias=AltHost-C,conf=ON;

ADD-HOST:hostname=host-A,rmthost=host-D,alias=AltHost-D,conf=ON;

2. Introducing other hosts (i.e., host-A, host-C, and host-D) to host-B under a Dual-LAN configuration

MODIFY-NTWK:hostname=host-B,dualhost=AltHost-B;

ADD-HOST:hostname=host-B,rmthost=host-A,alias=AltHost-A,conf=ON;

ADD-HOST:hostname=host-B,rmthost=host-C,alias=AltHost-C,conf=ON;

ADD-HOST:hostname=host-B,rmthost=host-D,alias=AltHost-D,conf=ON;

3. Introducing other hosts (i.e., host-A, host-B, and host-D) to host-C under a Dual-LAN configuration

MODIFY-NTWK:hostname=host-C,dualhost=AltHost-C;

ADD-HOST:hostname=host-C,rmthost=host-A,alias=AltHost-A,conf=ON;

ADD-HOST:hostname=host-C,rmthost=host-B,alias=AltHost-B,conf=ON;

ADD-HOST:hostname=host-C,rmthost=host-D,alias=AltHost-D,conf=ON;

4. Introducing other hosts (i.e., host-A, host-B, and host-C) to host-D under a Dual-LAN configuration

MODIFY-NTWK:hostname=host-D,dualhost=AltHost-D;

ADD-HOST:hostname=host-D,rmthost=host-A,alias=AltHost-A,conf=ON;

ADD-HOST:hostname=host-D,rmthost=host-B,alias=AltHost-B,conf=ON;

ADD-HOST:hostname=host-D,rmthost=host-C,alias=AltHost-C,conf=ON;

The example below shows a simulated two-host distributed product configuration where the dual LAN interfaces on each host are assigned hostnames other than the official hostname of that machine (i.e., gunes_1/gunes_2 for gunes and neptun_1/neptun_2 for neptun).

A total of 3 LAN interfaces on each host are necessary for private and public LAN access (i.e., two for private LAN access and one for public LAN access).

EXAMPLE

- Modify `/etc/hosts` file on gunes & neptun as follows:

192.203.250.120 gunes # official hostname

192.203.250.121 neptun # official hostname

129.205.250.34 gunes gunes_1 # to simulate private lan 1

129.206.250.34 gunes gunes_2 # to simulate private lan 2

129.205.250.33 neptun neptun_1 # to simulate private lan 1

129.206.250.33 neptun neptun_2 # to simulate private lan 2

- Modify `/etc/hostname.le?` files on *gunes* as follows:

/etc/hostname.le0 -> gunes_1

/etc/hostname.le1 -> gunes_2

- Modify `/etc/hostname.le?` files on *neptun* as follows:

/etc/hostname.le0 -> neptun_1

/etc/hostname.le1 -> neptun_2

- Run *ebs_config* script on *gunes* and specify *gunes_1* as the Distributed7 hostname. Verify that a file named `/etc/amgrhost` gets created and *gunes_1* is listed in this file.
- Run *ebs_config* script on *neptun* and specify *neptun_1* as the Distributed7 hostname. Verify that a file named `/etc/amgrhost` gets created and *neptun_1* is listed in this file.
- Sync/reboot *gunes* and *neptun* and wait until they come up. Start Distributed7 on both hosts subsequently using the *apm_start* command.
- Run the ``` `uname -n` ``` command and verify official host names.
- Use following MMI/MML commands on *gunes* to configure its network interfaces such that private dual LAN is in use:

  MOD-NTWK:hostname=gunes_1,dualhost=gunes_2;

  ADD-HOST:hostname=gunes_1,rmthost=neptun_1,alias=neptun_2, conf=ON;
- Use following MMI/MML commands on *neptun* to configure its network interfaces such that private dual LAN is in use:

  MOD-NTWK:hostname=neptun_1,dualhost=neptun_2;

  ADD-HOST:hostname=neptun_1,rmthost=gunes_1,alias=gunes_2, conf=ON;
- Run the *ebs_sysinfo* command on both hosts and verify its output. Mote that host names/ addresses on the private LAN should be the ones displayed and there should be no references to the official hostname or address.

Run the ***ebs_showlink*** command on both hosts and verify its output. Note that host names/ addresses on the private LAN should be the ones displayed, and there should be no references to the official hostnames or addresses.

6.11.10.1 Dual-LAN Subnet Configuration

When configuring a dual LAN in which subnet addressing is employed across the two distinct networks, the following parameters associated with the NTWK MO must be modified on each host machine prior to introducing any HOST MO instances to the MO database on that machine:

- NETMASK1: 32 bit mask—specified in hex format—used to extract the "network id" information on the primary network.
- NETMASK2: 32 bit mask—specified in hex format—used to extract the "network id" information on the secondary network, if any.

The default values for both masks are initialized on the basis of class type associated with the corresponding network, as follows:

Class Network	Initialized to Hex Format
Class A	7f000000
Class B	3fff0000
Class C	1fffff00

Note that default initialization of NETMASK1/NETMASK2 parameters is based on the standards associated with different classes of networks. It takes into account the fact that "network id" information for different classes of networks is stored in different bit locations and is of different sizes.

When subnet addressing is in use, default values of NETMASK1 and NETMASK2 need to be modified so that it is possible to extract the "extended network id" information from within an IP address in either network. The "extended network id" information includes "network id" and "sub-network id." Its length may vary from one subnet configuration to another.

EXAMPLE

Assume Class A addressing is used across both LAN interfaces, and the high 8 bits of the *host id* field within the 32 bit address are reserved for subnet addressing. The default settings of NETMASK1 and NETMASK2 parameters do not allow the sub-network id information to be extracted, since the mask applies to the most significant 8 bits of the address. Thus, it is only when the default values of NETMASK1 and NETMASK2 are changed to hex 7fff0000 that it becomes possible to extract the "extended network id" info for either network.

The MMI/MML command to perform this task is as follows:

MODIFY-NTWK:HOSTNAME=hostname,DUALHOST=dualhost,
NETMASK1=7fff0000,NETMASK2=7fff0000;

where "hostname" identifies the local host name to be used on the primary network, and "dualhost" identifies the alternate host name to be used on the secondary network.

6.11.10.2 Single-LAN to Dual-LAN Configuration

This procedure assumes an already-operational distributed Distributed7 environment comprising two or more host machines connected to each other. Note that the procedure described here must be done on *every* host machine.

1. Run ***ebs_stop*** to terminate Distributed7 software on the local host.
2. Run the ***ebs_config*** utility with the ***-u*** option to remove all local database files associated with NTWK, HOST, and TCPCO managed objects from the local host.
3. Run ***ebs_start*** to restart the Distributed7 software. When the ***netd*** daemon process runs, the system prompts you to configure the Distributed7 network.
4. Run ***mml 0*** to re-create the NTWK, HOST, and TCPCO managed objects on the local host.

5. In MMI/MML, modify the host and define all remote hosts.

EXAMPLE

To define a dual-LAN configuration for Host-A with an alias name AltHost-A, enter the following:
MOD-HOST:HOSTNAME=Host-A,DUALHOST=AltHost-A;
To define remote host entries for Host-B, Host-C, and Host-D with the alias names of AltHost-B, AltHost-C, and AltHost-D, respectively, enter the following:
ADD-HOST:HOSTNAME=Host-A,RMTHOST=Host-B,ALIAS=AltHost-B,CONF=ON;
ADD-HOST:HOSTNAME=Host-A,RMTHOST=Host-C,ALIAS=AltHost-C,CONF=ON;
ADD-HOST:HOSTNAME=Host-A,RMTHOST=Host-D,ALIAS=AltHost-D,CONF=ON;

***Important**: When configuring the NTWK managed object, make sure to specify the correct alias name for the local host. Also, for every remote host defined, there must be a corresponding alias name in the HOST managed object database. There is no need to create individual entries in the TCPCON managed object database because the ADD-HOST command generates all the appropriate TCPCON entries and populates them with the default settings.*

6.11.10.3 Dual-LAN to Single-LAN Configuration

This procedure assumes an already-operational distributed Distributed7 environment comprising two or more host machines connected to each other. Note that the procedure described here must be done on *every* host machine.

1. Run ***ebs_stop*** to terminate Distributed7 software on the local host.
2. Run the ***ebs_config*** utility with the ***-u*** option to remove all local database files associated with NTWK, HOST, and TCPCO managed objects from the local host.
3. Run ***ebs_start*** to restart the Distributed7 software. When the ***netd*** daemon process runs, the system prompts you to configure the Distributed7 network.
4. Run ***mml 0*** to re-create the NTWK, HOST, and TCPCO managed objects on the local host.
5. In MMI/MML, modify the network and define all remote hosts.

EXAMPLE

To define a single-LAN configuration for Host-A, enter the following:
MOD-NTWK:HOSTNAME=Host-A;
To define remote host entries for Host-B, Host-C, and Host-D, enter the following:
ADD-HOST:HOSTNAME=Host-A,RMTHOST=Host-B,CONF=ON;
ADD-HOST:HOSTNAME=Host-A,RMTHOST=Host-C,CONF=ON;
ADD-HOST:HOSTNAME=Host-A,RMTHOST=Host-D,CONF=ON;

***Important:** When configuring the NTWK managed object, make sure no alias name is specified for the local host. Also, make sure no alias names are specified for remote hosts defined in the HOST managed object database. There is no need to create individual entries in the TCPCON managed object database because the ADD-HOST command* *generates all the appropriate TCPCON entries and populates them with the default settings.*

6.12 Viewing the Status of System Processes

The Distributed7 software is made up of several processes. All processes that are registered to the Distributed7 environment running on the local host can be displayed with the *ebs_ps* command. Processes that are started and managed by the ***apmd*** daemon can be displayed with the *apm_ps* command. These processes may be on different hosts.

6.12.1 ebs_ps

The *ebs_ps* command (*Section 8.2.21*) displays a snapshot view of the active processes with their status and other identifying information. The command should be entered at the command line as follows:

```
ebs_ps
```

For each process the output provides the process ID (PID), status (STAT), mode, hostname, multiplexer (MUX), and type of object. A sample output is shown below:

PID	STAT	MODE	HOST	MUX	OBJECT
424	ok	LX|A	sparc4a	spm/0	daemon [name=apmd]
425	ok	LX|A	sparc4a	spm/2	daemon [name=mlogd]
426	ok	LX|A	sparc4a	spm/3	daemon [name=spmd]
427	ok	LX|A	sparc4a	spm/5	daemon [name=netd]
428	ok	LX|A	sparc4a	spm/7	daemon [name=alarmd]
429	ok	LX|A	sparc4a	spm/9	daemon [name=dsmd]
430	ok	LX|A	sparc4a	spm/10	daemon [name=dkmd]
2478	ok	LX|A	sparc4a	spm/11	nmdobj [name=ebs_ps]

The description of the columns is given in *Section 8.2.21 on page 8-33*.

6.12.2 apm_ps

The *apm_ps* command (*Section 8.3.5*) displays a snapshot view of the active processes that are started by the ***apmd*** daemon processes with their status and other identifying information. The command should be entered at the command line as follows:

```
apm_ps
```

For each process the output provides the process ID (PID), program ID (PROG), process tag (TAG), action mode, status, heartbeat status (HBSTAT), number of times process was restarted (RETRY), and execution state (ESTATE). A sample output is shown below:

```
PID  PROG  TAG             ACTION    STATUS  HBSTAT  RETRY  ESTATE
425  p001  mlogd@sparc4a   failsafe   ok      ok      0     all states
426  p002  spmd@sparc4a    failsafe   ok      ok      0     all states
427  p003  netd@sparc4a    failsafe   ok      ok      0     all states
428  p004  alarmd@sparc4a  failsafe   ok      ok      0     all states
429  p005  dsmd@sparc4a    failsafe   ok      ok      0     all states
430p006dkmd@sparc4a    failsafe   ok      ok      0     all states
```

The description of the columns is given in *Section 8.3.5 on page 8-65*.

6.13 Using OMAP

To use OMAP for gathering SS7 statistics, you must start the module with the *AccessOMAP* command that is described in *Section 7.2.3 on page 7-6*.

The OMAP module creates five files each day, one file each for MTP Level 2 measurements, MTP Level 3 ‘5 minute’ measurements, MTP Level 3 ‘30 minute’ measurements, SCCP measurements, and TCAP measurements. The files are stored in the ***$EBSHOME/access/RUN*/omaplog*** directory. Each file contains all the statistics gathered from the particular SS7 layer for that day, therefore, size varies. The files are kept as binary files in order to conserve space. AccessOMAP archives the logfiles accumulated in the ’omaplog’ directory every week by saving them in another directory at the same level with ’omaplog’.

Important*: OMAP log files continue to accumulate; older files, i.e., archived, omaplog directories should be deleted or moved elsewhere to free up space.*

Two methods exist to view the contents of the OMAP files. The first option involves developing a program using the OA&M API function calls. Another option entails using the *omap_report.c* source code file in the ***$EBSHOME/access/sample/omap*** directory. This file should be compiled with the Makefile contained in that directory. Then, the executable will display a report of the contents of all the OMAP files existing in the ***$EBSHOME/access/RUN*/omaplog*** directory.

Important*: The omap_report program performs a significant number of disk I/O operations; therefore is quite demanding on the CPU resources of the local host. It is highly recommended that this program is not executed while the system is handling live traffic.*

6.14 CompactPCI Hot-swap

Distributed7 provides basic CompactPCI hot-swap capability in systems where hot-swap is supported. It allows users to replace CompactPCI SS7 boards without stopping Distributed7 stack and shutting down the system.

6.14.1 Important Guidelines

It is extremely important to understand following guidelines before performing a hot-swap operation. The system may crash or become unstable if you do not follow them.

- Do not perform a hot-swap operation with an SS7 board if Distributed7 does not support it. Supported SS7 boards are listed below
- Do not perform a hot-swap operation in a system that does not support it
- Distributed7 saves critical system and board information before extraction. You must replace an SS7 board with an identical SS7 board so that the information that is saved before extraction can be loaded on to the new board after insertion
- Never leave a hot-swap operation uncompleted. If you cannot insert a new board, re-insert the old one and restore system s state
- Do not stop Distributed7 or shut down the system in the middle of a hot-swap operation
- Always insert the replacement board into the same slot where extracted board was installed

6.14.2 Supported SS7 Boards

Following boards are supported by Distributed7 for hot-swap operation:

- PMC8260/DS1: front and rear access PMC modules on MFIO-120 carrier (see section 2.3.8 in the Distributed7 Installation and Maintenance Manual)
- ARTIC1000: CompactPCI single slot, 6U form-factor SS7 board (see section 2.3.9.1 in the Distributed7 Installation and Maintenance Manual)

Important: *Distributed7 supports hot-swap for PMC8260/DS1 modules on an MFIO-120 carrier only. Never perform a hot-swap operation with these boards if they are installed into a PMC slot of a CPU board or on a different type of carrier board.*

6.14.3 Performing a CompactPCI Hot-swap

The Distributed7s hot-swap user interface is the SS7BOARD mml command. See section 9.4.6 for details of the command.

1. Before extracting a board, you must suspend all device-driver activities for that board instance. To do this enter the following mml command:

```
MODIFY-SS7BOARD: HOSTNAME=<hostname>, BOARDNM=<boardnm>, INST=<inst#>,
   conf=SUSPEND;
```

Important: *MFIO-120 carriers can hold up to two SS7boards. If you are going to extract a MFIO-120 carrier which holds two SS7 boards, you must suspend both of the board instances.*

The “SUSPEND” command must be completed successfully. If it returns an error, do not continue. The blue hot-swap LED on the CompactPCI carrier front panel turns on when the board is suspended successfully.

2. Remove the suspended board from its CompactPCI slot.
3. Insert the replacement board into the same CompactPCI slot.
4. Resume device-driver activities for the newly inserted SS7 board:

MODIFY-SS7BOARD: HOSTNAME=<*hostname*>, BOARDNM=<*boardnm*>, INST=<*inst#*>, conf=*RESUME*;

5. Once the board has successfully resumed device-driver activities, enter the following command to make it available for MTPL3:

MODIFY-SS7BOARD: HOSTNAME=<*hostname*>, BOARDNM=<*boardnm*>, INST=<*inst#*>, conf=*ON*;

There is no need to add SS7 links after the replacement is complete. Distributed7 software will restore board’s software state after the hot-swap operation is complete.

Chapter 7: System Processes

7.1 Chapter Overview

This chapter provides descriptions of the Distributed7 system processes (daemons) and their configuration files. The system processes are summarized in the table.

Table 7-1: Daemon Summary

New Command	Description
AccessAlarm	Starts the Alarm handling process
AccessMOB	Starts the Managed Object Browser graphical user interface
AccessOMAP	Starts the Operations, Maintenance, and Administration process (optional)
AccessSNMP	Starts the SNMP agent
AccessStatus	Starts monitoring SS7 links.
AccessMonitor	Starts the system software status monitor
apmd	Starts Application Process Manager (APM) daemon.
dkmd	Distributed Kernel Memory (DKM) manager daemon.
dsmd	Sets up the distributed shared memory manager system process.
isupd	Starts the ISUP user part
logd	Starts the log process (LOG_MNGR) for message logging capabilities
mlogd	Starts master event log daemon.
mmi	Starts the Man-Machine Language user interface
mml	Starts the Man-Machine Language user interface
netd	Sets up connections for inter-machine messaging in distributed environment
rtc_agent	Sets up and maintains remote TCAP connections in a distributed environment
scmd	Starts the SCCP user part
spmd	Starts the Distributed7 infrastructure
tcmd	Sets up the TCAP multiplexer.
upmd	Starts User Part Multiplexer
gsma	GSM A-Interface daemon

7.2 Daemon Listing

***Important**: Please refer to Chapter 2 for a list of the user commands with external dependencies to make sure your environment has the necessary software libraries.*

To use the Distributed7, set the $**EBSHOME** environment variable and include ***$EBSHOME/access/bin*** in the command path. The $**EBSHOME** environment variable should be set to the path where the Distributed7 software is installed.

To set the variable, use a C-shell command similar to this sample:

```
setenv EBSHOME /<samedir>/<mydir>/<mySS7>
```

***Important**: $EBSHOME must be less than or equal to 1024 characters.*

To add the Distributed7 ***bin*** directory into the command path, use the following command:

```
setenv PATH ${PATH}:$EBSHOME/access/bin
```

On-line reference manuals on all system processes are also available in the Distributed7 system. These reference manuals are provided in the form of manual pages so that the user can invoke the UNIX standard ***man(1)*** utility to review the information contained in them. The Distributed7 manual pages are provided within the ***$EBSHOME/access/manpages*** directory. Therefore, the user should set the **MANPATH** environment variable as follows:

```
setenv MANPATH ${MANPATH}:$EBSHOME/access/manpages
```


7.2.1 AccessAlarm

NAME

AccessAlarm or ***alarmd*** Starts the alarm handling process.

SYNOPSIS

AccessAlarm [-o] [-d dir***] [-n*** nfile***] [-m*** msize***]***

alarmd [-o] [-d dir***] [-n*** nfile***] [-m*** msize***]***

DESCRIPTION

AccessAlarm/alarmd This daemon is responsible for collecting, analyzing, logging, and displaying alarm messages that may be generated by the user/kernel-space components comprising the Distributed7 system software and/or user application programs running under the Distributed7 environment. A list of alarm conditions that may be encountered by the Distributed7 system software is provided in the form of a set of alarm text files that are all located in the ***$EBSHOME/access/RUN/config/ALARM*** directory. These files contain information about all alarm conditions that may be encountered by a particular software module and about the specifics of the individual alarm conditions, e.g., alarm module identifier, group identifier, severity, alarm text, and associated parameters. The severity and/or text of a particular alarm condition can be modified by the users of the Distributed7 product simply by editing the information contained in these files and re-starting the ***alarmd*** daemon. Care should be taken, however, not to change the module and group identifier information as well as the number of parameters supplied within individual alarm text messages, as this misleads the ***alarmd*** daemon.

A list of individual alarm groups is provided within the *alarmGroups* file under the ***$EBSHOME/access/RUN/config/ALARM*** directory. Also listed in this directory is the *alarmConfig* file, which contains configuration related information for use by the ***alarmd*** daemon (whether messages collected by the ***alarmd*** daemon should be displayed on the system console in addition to being logged for off-line analysis). Unless the ***-d dir*** command line option is in use, The log files maintained by the ***alarmd*** daemon are located under the ***$EBSHOME/access/RUN/alarmlog*** directory and they all start with the AccessAlarms prefix. Users interested in reviewing the contents of the alarm log files maintained on specified hosts within a Distributed7 environment can use the *ebs_report* command-line utility, i.e., for retrieving and displaying selected pieces of information stored in alarm log files.

While in operation, ***alarmd*** will be registered exclusively with the Distributed7 environment on the local host machine as a daemon object, under the name ALM_MNGR—a macro defined in the <***api_macro.h***> header file.

-o	This option is used to overwrite the time stamp information contained in the alarm message received. By default, the time stamp information is populated when the alarm message is submitted, i.e., at the point of origination. ***alarmd*** uses this information as is.
-d dir	Stores alarm log files on specified host machines. By default, alarm log files are located under the ***alarmlog*** directory in the ***$EBSHOME/access/RUN*** directory. If the ***dir*** is specified, the alarm log files are expected to be located under the ***dir/alarmlog*** directory. If ***dir*** directory (or ***alarmlog*** sub-directory) does not exist, ***alarmd*** will make an attempt to create all necessary directories.
-n nfile	Allows a maximum number of ***nfile*** log files to co-exist under the log directory at any time. When this count is exceeded, delete the oldest log file to create room for new log files. By default, ***alarmd*** allows up to 10 log files to co-exist, where each log file can grow up to ***msize*** kilobytes in size. The absolute maximum number of log files that can co-exist on a system is 100.
-m msize	Allows each log file in log directory to grow to ***msize*** kilobytes in size. By default, ***alarmd*** allows each log file to grow to 512 kilobytes in size before it starts writing into a new log file. When the total number of log files exceeds the limit set via the ***nfile*** argument, ***alarmd*** deletes the oldest log file before creating a new log file.

Note*: The alarm process is automatically started by the apmd process and does not have to be started separately. This command is only needed to activate the ALM_MNGR process if it is deactivated while the system is running.*

Important*: Refer to the Maintenance Manual for trouble reports and alarm definitions.*

FILES

$EBSHOME/access/RUN/config/ALARM/alarmGroups

$EBSHOME/access/RUN/config/ALARM/alarmConfig

$EBSHOME/access/RUN/alarmlog/AccessAlarms.*

SEE ALSO apmd, ebs_alarm, ebs_report

7.2.2 AccessMOB

NAME

AccessMOB Starts the graphical user interface.

SYNOPSIS

AccessMOB sp

mob sp

DESCRIPTION

AccessMOB/mob Starts the Managed Object Browser graphical user interface for node configuration of all managed objects of a node (instead of using MML).

The ***mob*** interface for a specified signalling point can be started automatically by the *apmd* daemon (at system start-up time) provided that there is a corresponding entry in the *apmd* configuration file. Alternatively, it can be started manually from the command line.

While in operation, ***mob*** will be registered exclusively with the Distributed7 environment on the local host machine as a daemon object, under the name GUI_MNGR(sp) (a macro defined in the <***api_macro.h***> header file).

sp The logical signalling point number used with *upmd*, or *ebs_start*.

Important*: The upmd process <u>must</u> be running before executing this command. Use ebs_ps to confirm that these processes exist.*

SEE ALSO mml, upmd

7.2.3 AccessOMAP

NAME

AccessOMAP or ***omapd*** Starts the SS7 OMAP process.

SYNOPSIS

AccessOMAP [-q report_frequency] sp

omap [-q report_frequency] sp

DESCRIPTION

AccessOMAP/omapd Starts the SS7 Operations, Maintenance, Administration Part (OMAP) process. The ***omapd*** daemon is responsible for collecting and storing various pieces of SS7-specific measurements data associated with a user-specified signalling point (SP). This data includes information about Message Transfer Part (MTP) Level 2 (collected at 30-minute intervals) and Level 3 measurements (collected at 5- and 30-minute intervals), Signalling Connection Control Part (SCCP) measurements (collected at 30-minute intervals), and Transaction Capabilities Application Part (TCAP) measurements (collected at 30-minute intervals).

The log files maintained by the ***omapd*** daemon are located under the ***$EBSHOME/access/RUN[0-7]/omaplog*** directory for the corresponding signalling point. Users interested in reviewing the contents of the log files maintained by the ***omapd*** daemon and generating summary reports can use the `omap_report` command-line utility (or customized versions of it) whose source code listing is given under the ***$EBSHOME/access/sample/omap*** directory. Alternatively, the *oam_retrieve()* function, which is part of the Distributed7 OAM API Library, can be used to retrieve measurements data collected by the AccessOMAP daemon.

The ***omapd*** daemon for a specified signalling point can be started automatically by the *apmd* daemon (at system start-up time) provided that there is a corresponding entry in the *apmd* configuration file. Alternatively, it can be started manually from the command line.

As an operational practice, the AccessOMAP daemon on each host needs to be started prior to starting the daemon processes associated with the MTP, SCCP, and TCAP layers on that host. If this practice is not followed, then the measurements data collected and stored by the AccessOMAP daemon do not include the first set of statistics reported by the individual SS7 protocol layers. Also, if any of these SS7 protocol layers are instantiated on multiple hosts, then the AccessOMAP daemon for the corresponding signalling point needs to be started on all such hosts in order to be able to collect measurements data through all involved hosts.

While in operation, ***omapd*** is registered exclusively with the Distributed7 environment on the local host machine as a daemon object, under the name OMAP_MNGR(sp) (a macro defined in the <***api_macro.h***> header file).

-q report_frequency This option is used to specify the time interval in minutes at which all measurements are collected. If this option is not specified, then the measurement interval defaults to 30 minutes

sp The logical signalling point number.

FILES

$EBSHOME/access/RUN[0-7]/omaplog/mtp2.mmddyy

$EBSHOME/access/RUN[0-7]/omaplog/mtp3.mmddyy

$EBSHOME/access/RUN[0-7]/omaplog/mtp3_5min.mmddyy

$EBSHOME/access/RUN[0-7]/omaplog/sccp.mmddyy

$EBSHOME/access/RUN[0-7]/omaplog/tcap.mmddyy

$EBSHOME/access/RUN/omaplog/tcap.mmddyy

$EBSHOME/access/sample/omap/omap_report.c

$EBSHOME/access/sample/omap/Makefile

Important*: The upmd process must be running before executing this command. Use ebs_ps to confirm that these processes exist.*

NOTES

When the TCAP over TCP/IP feature is in use, the TCAP layer will report its OMAP statistics to the AccessOMAP instance associated with the signalling point 0. Unless this instance of AccessOMAP is alive, no OMAP measurements data will be collected for TCAP [over TCP/IP] applications. Also note that since this data is not associated with any signalling point, per se, it will be saved under the ***$EBSHOME/access/RUN/omaplog*** directory as opposed to being saved in ***$EBSHOME/access/RUN[0-7]/omaplog*** directory. The ***omap_report*** utility contains built-in intelligence to search through all appropriate directories to locate the log files maintained by the AccessOMAP daemon.
The ***omapd*** daemon archives the log files under the *omaplog* directory every week by moving them to a *omaplog.mmddyy* directory that is at the same level as the *omaplog* directory. It is highly recommended that users check on the size of accumulated *omaplog* files on their system from time to time and clean up the old log files from their system to guard against excessive disk space consumption.

SEE ALSO oam_retrieve(), apmd

7.2.4 AccessSNMP

NAME

snmp_p Simple network management protocol interface.

SYNOPSIS

$EBSHOME/access/bin/snmp_p [-a] -v 1|2 sp

$EBSHOME/access/bin/AccessSNMP [-a] [-h] -v 1|2 sp

DESCRIPTION

AccessSNMP Starts the SNMP agent which replies to all the SNMPv1 and SNMPv2 requests that come from network managers. It responds according to the MIB view of Distributed7.

sp Identifies the signalling point of interest.

-h Displays message contents in hexadecimal format (default).

snmp_p This daemon is responsible for establishing a standardized interface between external network management entities and the Distributed7 platform using version 1 or version 2 Simple Network Management Protocol (SNMP). The selection of the SNMPv1 vs. SNMPv2 protocol is via the "-v" command line option provided to ***snmp_p*** at the time of program invocation.

The ***snmp_p*** interface for a specified signalling point can be started automatically by the *apmd* daemon (at system start-up time) provided that there is a corresponding entry in the *apmd* configuration file. Alternatively, it can be started manually from the command line. While in operation, ***snmp_p*** will be registered exclusively with the Distributed7 environment on the local host machine as a daemon object, under the name PRTCL_MNGR(sp) (a macro defined in the <***api_macro.h***> header file). This behavior of SNMP agent can be changed using the "-a" command line option. where SNMP agent will use the Distributed7 hostname assigned to a machine instead of using the official hostname for that machine.

Upon start-up, ***snmp_p*** will perform the following tasks:

- read through the SNMPv1 and SNMPv2 "*.conf" files located under the corresponding ***$EBSHOME/access/RUN*/config/SNMP*** directory
- establish a transport endpoint for communication with external management entities using the User Datagram Protocol (UDP)
- spawn the corresponding ***snmp_i*** daemon to be able to perform Managed Object related tasks if and when it becomes necessary
- wait for UDP datagrams from external network management entities or trap requests initiated by the Distributed7 system software

When the ***snmp_p*** daemon receives a UDP datagram from an external management entity, it will interact with the corresponding ***snmp_i*** daemon (i.e., in order to resolve and process the incoming request) and reply to the originating party with an appropriate response. Alternatively, when the ***snmp_p*** daemon receives a trap request via the ***snmp_i*** daemon, it will relay this request to the external network management entity via a datagram.

Note that it is also possible to define interfaces between the Distributed7 alarm handler and the ***snmp_i*** daemon process such that ***snmp_p*** is informed about selected alarm conditions that occur on the Distributed7 platform and it subsequently notifies the external network management entities to that effect by generating SNMP trap requests. This latter capability involves customizing of the *alarmd* daemon by manipulating the AlmExt.c file provided under the ***$EBSHOME/access/sample/alarm*** directory, re-compiling/linking the source/object codes in that directory, and subsequently replacing the default *alarmd* daemon executable under the ***$EBSHOME/access/bin*** directory with the newly constructed version of it. More information about this procedure can be found in the *Distributed7 User Manual*.

The operations of the ***snmp_p*** daemon are controlled via the SNMPv1 and SNMPv2 Management Information Base (MIB) text files and SNMP command table located under the ***$EBSHOME/access/RUN/config/SNMP*** directory. Following initial system software installation, both of these files can be edited to customize the operations of the Distributed7 SNMP agent (e.g., by modifying the access privileges, defining attributes for additional managed objects, modifying existing SNMP commands and/or introducing additional ones).

Important*: Following initial software installation, all configuration files listed under the* ***$EBSHOME/access/RUN/config/SNMP/etc*** *directory must be copied by the system administrator to all appropriate* ***$EBSHOME/access/RUN*/config/SNMP*** *directories with the "*.conf" extension and hand-edited to specify the Internet Protocol (IP) address of the external network management entity and setup-related system parameters. Once this step is successfully completed, the Distributed7 SNMP agent can be started properly. For further information about these configuration files and their exact use, refer to the Distributed7 documentation.*

FILES

$EBSHOME/access/RUN/config/SNMP/README

$EBSHOME/access/RUN/config/SNMP/mib_text.v1

$EBSHOME/access/RUN/config/SNMP/mib_text.v2

$EBSHOME/access/RUN/config/SNMP/snmp_cmnd.tbl

$EBSHOME/access/RUN/config/SNMP/etc/acl.ini

$EBSHOME/access/RUN/config/SNMP/etc/community.ini
$EBSHOME/access/RUN/config/SNMP/etc/context.ini
$EBSHOME/access/RUN/config/SNMP/etc/party.ini
$EBSHOME/access/RUN/config/SNMP/etc/trap.ini
$EBSHOME/access/RUN/config/SNMP/etc/view.ini
$EBSHOME/access/RUN<_ s_ p#>/config/SNMP/acl.conf
$EBSHOME/access/RUN<_ s_ p#>/config/SNMP/community.conf
$EBSHOME/access/RUN<_ s_ p#>/config/SNMP/context.conf
$EBSHOME/access/RUN<_ s_ p#>/config/SNMP/party.conf
$EBSHOME/access/RUN<_ s_ p#>/config/SNMP/trap.conf
$EBSHOME/access/RUN<_ s_ p#>/config/SNMP/view.conf
$EBSHOME/access/sample/alarm/AlmExt.c

SEE ALSO apmd, AccessAlarm, alm_trap()

7.2.5 AccessStatus

NAME

AccessStatus Monitors signalling point configuration, MTP level 2 and level 3 status, and traffic capacity utilization of SS7 links.

SYNOPSIS

AccessStatus sp

DESCRIPTION

AccessStatus Displays a scrollable *tcl* window with one row of information for each SS7 link defined in the corresponding Signalling Point (sp).
AccessStatus can be started on each host where the Distributed7 CORE system is running. This is to say that MTP/L2 or MTP/L3 is not necessary in order to start AccessStatus.
Upon start-up, AccessStatus gets the current link information from the MTP/L3 and displays information only for these links. If a link is added or deleted, MTP/L3 will inform all AccessStatus processes so that the correct link information can be displayed.

sp Signalling point number of the system.

DISPLAY

For each link entry, the following information is displayed:

LinkSet The linkset name of the link

Link The link name

SLC Signalling Link code of the link

L3State MTP/L3 status, can be one of the following:

- *failed* - link is unavailable
- *available* - link is available

Inhibit Inhibition, can be on of the following:

- *local* - link is locally inhibited
- *remote* - link is remotely inhibited
- *loc/rem* - link is locally and remotely inhibited

ProcOut Processor Outage, can be one of the following:

- *local* - local processor outage is set
- *remote* - remote processor outage is set
- *loc/rem* - local and remote processor outage is set

L2State MTP/L2 state of the link, can be one of the following:

- *pow_off* - power off
- *oos* - out of service

- *init_al* - initial alignment
- *alg_ready* - alignment ready
- *alg_not* - alignment not ready
- *is* - in service
- *proc_out* - processor outage

SueCnt	SUERM (Signalling unit Error Rate Monitor) counter
TxFrame	Number of transmitted frames per second
RxFrames	Number of received frames per second
TxBand	Transmit bandwidth usage in percentage
RxBand	Receive bandwidth usage in percentage
Baud Rate	Baud rate of the link, expressed in bits per second

EXAMPLE

AccessStatus 0

Starts up AccessStatus for signalling point 0.

7.2.6 AccessMonitor

NAME

AccessMonitor Starts the system software status monitor.

SYNOPSIS

AccessMonitor <sp#>

DESCRIPTION

AccessMonitor Provides the ability to monitor the status of the SS7 protocol stack running on multiple hosts within a Distributed7 environment, via a selected host, with a Graphical User Interface (GUI). It supports both stand-alone and distributed configurations. When executed under a distributed configuration, it also monitors the health of the kernel-level TCP/IP connections to all remote hosts on an on-going basis.

Upon start-up, AccessMonitor brings up a map of hosts that are currently configured and accessible through the local host. The main window gives general information about the current status of SS7 layers for each host and the TCP/IP connections among all active hosts on the system. By selecting each active layer from the main window, a new window for each layer will be created which displays further, more detailed, status information about daemon processes and STREAMS components.

sp Signalling point number of the system.

EXAMPLE

AccessMonitor 0

Starts up AccessMonitor for signalling point 0.

7.2.7 apmd

NAME

apmd Starts Application Process Manager (APM) daemon.

SYNOPSIS

apmd [-c|s|x] [-f cfgfile***]***

DESCRIPTION

apmd Starts the ***apmd*** daemon, which is a general application process manager. Its primary responsibility is to create and manage processes according to instructions specified in an *apmconfig* configuration file. It is also responsible for processing application requests (placed with APM library calls) for creation of a new process, communication between parent and child processes, detection of child process termination, and network-based inter-process communication via UNIX signals. The ***apmd*** is also capable of communicating with its peers on other hosts in a distributed processing environment (e.g., to spawn a process on a remote host or send signals to remote processes). This communication is message-based and uses the functionality provided by the Distributed7 SPM library.

-c Starts the AccessCRP (Call Routing Point) version of ***apmd***. The ***$PRODID*** and ***$RUNID*** environment variables must be set to appropriate values prior to program execution. This version supports multiple application domains on a single host. The settings of the above environment variables are used to determine the specific ***apmd*** instance to invoke.

-s Starts the AccessSERVICES version of ***apmd***. The ***$DOMID*** environment variable must be set to an appropriate value prior to program execution. This version supports multiple application domains on a single host. The current setting of the ***$DOMID*** environment variable is used to determine the specific ***apmd*** instance to invoke.

-x Starts the normal Distributed7 version of ***apmd***. The ***$EBSHOME*** environment variable must be set to an appropriate value prior to program execution. This version does not support multiple application domains on a single host.The ***apmd*** daemon executes in a local-exclusive mode, i.e., only one instance of this version of ***apmd*** may be executing on a host.

-f cfgfile Use the configuration file specified by ***cfgfile*** instead of the default. By default, ***apmd*** uses the *apmconfig* or ***apmconfig.old*** configuration file under an appropriate release directory. (See the FILES section.)

Important*: The IPC shared memory segment for the APM trace functionality must be created in advance using the apm_trinit utility. If it is not created, then* ***apmd*** *will terminate.*

The ***apmd*** supports different types of application environments, as described by the options (c, s, x). If the user does not explicitly specify one of the options in the command, ***apmd*** will determine the appropriate environment based on environment variable settings and the following logic:

- If ***$DOMID*** is set, it assumes an AccessSERVICES environment.
- If ***$DOMID*** is not set but ***$PRODID*** and ***$RUNID*** are set, it assumes an AccessCRP environment.
- If none of the above environment variables are set but ***$EBSHOME*** is set, it assumes a basic Distributed7 environment.

The AccessSERVICES and AccessCRP versions of ***apmd*** support multiple application domains on a single host. Multiple instances of either version of the ***apmd*** daemon can be invoked on a single host to create and manage independent sets of application processes. The basic Distributed7 version does not support multiple application domains. Only one instance of this version can exist on a host. However, all three versions of the daemon may coexist on a given host.

Similar to the UNIX *init(1)* daemon, ***apmd*** maintains an internal run state during its life cycle. This run state corresponds to a software configuration which specifies a group of processes that should be spawned by ***apmd***. The configuration of active processes for different run states is defined in the *apmconfig* file. The run state of ***apmd*** changes either when it executes a pre-defined scenario from the *apmconfig* file which puts it in a new state or when the *apm_setstate* utility is used to request state change from the command line.

Once started, ***apmd*** reads all entries listed in *apmconfig* and copies the information to an internal process control table. From then on, ***apmd*** will use the information stored in the internal process control table. It will not consult the *apmconfig* file until the execution of the *apm_update* utility causes it to re-read the configuration file.

The first run state is retrieved by ***apmd*** from *apmconfig* during start-up. When ***apmd*** starts, it searches the *apmconfig* file for an *initdefault* entry. If one exists, ***apmd*** uses the run state specified in that entry as the initial state to enter. If an *initdefault* entry does not exist in the *apmconfig* file, then ***apmd*** enters a run state as follows:

- In AccessCRP environments, it moves to the **D** state.
- In AccessSERVICES environments, it moves to the **A** state.
- In Distributed7 environments, it moves to the **init** state.

When ***apmd*** has its initial state, and whenever its run state changes, ***apmd*** scans all entries listed in its internal process control table and executes each line that has an execution state which matches its current state. If the execution state of a particular process entry does not match the current run state, this entry is completely ignored. When an entry instructs ***apmd*** to spawn a process, ***apmd*** reads the entry for the process first, and then identifies the host where the process should be created. If the host is not local, ***apmd*** contacts its peer on the appropriate remote host and submits the process entry to that ***apmd*** for processing. For entries specifying the local host, ***apmd*** creates a new child process and executes the user-specified program.

The ***apmd*** can be configured to do more than simply spawning and terminating processes. Each entry in the process control table also specifies the behavior of ***apmd*** during the start-up and life of a child process. For example, ***apmd*** can be programmed to wait a specified time for an acknowledgment that a process has started (or terminated) before executing the next entry. It can also be programmed to take certain actions if and when a child process terminates. During the life of a process, ***apmd*** can be programmed to continually monitor the process through *heartbeat request* messages, which require *heartbeat response* messages from the process.

An entry in the process table may also instruct the ***apmd*** to change its run state to a new state after executing the line. This action will cause ***apmd*** to again scan the process control table. But now the lines with execution states matching this new state will be executed, possibly executing new scenarios, e.g., spawning a new set of processes and/or terminating an existing set of processes.

During its operations, ***apmd*** reports all process-related activities and run state changes to the *mlogd* process. The *mlogd* process creates a permanent record of these activities in the master log files on the local host.

When ***apmd*** is requested to terminate processes, it takes the actions described in the *apm_stop*, *apm_kill*, and *apm_killall* utilities.

FILES

The default configuration file names are as follows:

Distributed7 version:	***$EBSHOME/access/RUN/config/PMGR/apmconfig***
AccessSERVICES version:	***$EBSHOME/access/RUN/config/PMGR/apmconfig.old***
AccessCRP version:	***$APPHOME/apmconfig.old***

SEE ALSO mlogd, apm_start, apm_stop, apm_kill, apm_killall, apm_update, apm_ps, apmconfig

7.2.8 dkmd

NAME

dkmd Distributed Kernel Memory (DKM) manager daemon.

SYNOPSIS

dkmd [-ps] [-m module(s)]

DESCRIPTION

dkmd This daemon is responsible for setting up, maintaining, and tearing down the Distributed Kernel Memory (DKM) and Distributed Record Access (DRA) infrastructures that are available as part of this release of the Distributed7 system software. The DKM infrastructure allows kernel-resident applications executing under a distributed Distributed7 environment to share kernel-space information in an efficient manner by reading/writing through local copies of kernel-space data that are replicated on all hosts involved. The DRA infrastructure builds upon the DKM and is intended to fulfill the needs of database-oriented kernel-resident applications. It is through the DRA framework, a kernel application can view its kernel-space data in the form of a distributed database and operate on it.

The ***dkmd*** daemon expects the *netd* daemon on the local host to be up and running when it starts its execution. The only exception to this rule is when the ***dkmd*** daemon is started in the stand-alone mode, using the ***-s*** command-line argument. The ***dkmd*** daemon on a particular host is normally started automatically by the *apmd* daemon on that host provided that there is an entry in the *apmd* configuration file. Alternatively, it can be started manually from the command-line.

While in operation, the ***dkmd*** daemon on each host will be registered exclusively with the Distributed7 environment on that host as a daemon object, under the name DKM_MNGR (a macro that is defined in the **<*api_macro.h*>** header file).

-p Collect performance statistics by running a small set of performance benchmark tests upon start-up. The collected statistics can be retrieved at a later time simply by sending a SIGUSR1 signal to the ***dkmd*** daemon.

-s Execute in standalone mode. Note that in this mode of operation, ***dkmd*** daemon acts as a lock manager only, which simply facilitates protected access to a kernel-resident memory space allocated on the local host.

-m module Push STREAMS modules specified over the DKM multiplexer in the specified order.

SEE ALSO apmd, netd

7.2.9 dsmd

NAME

dsmd Sets up the distributed shared memory manager system process.

SYNOPSIS

dsmd [-s]

DESCRIPTION

dsmd Allocates, maintains, and de-allocates resources associated with the distributed shared memory (DSM) subsystem that is available as part of the Distributed7 Release 1.0.0 system software. The ***dsmd*** process performs these functions on behalf of application programs that issue requests through the ***libdsm*** API library calls.

-s Executes the process in stand-alone mode. This mode of operation causes the ***dsmd*** process to act as a lock manager to facilitate protected access to the IPC shared memory segments allocated on the local host.

A distributed shared memory segment allows processes executing on the individual hosts of a Distributed7 environment to attach local copies of the segment to their virtual address space. In other words, UNIX IPC shared memory is replicated on each host. The processes share information by reading/writing from/to these segments. To prevent inconsistencies and/or collisions when reading/writing through a DSM segment, processes must always use synchronization variables (e.g. read/write locks) when reading/writing through local copies of the segment.

The ***dsmd*** process is normally started automatically by the *apmd* process on a particular host. To be started automatically, the command must be an entry in the *apmd* configuration file (e.g. *apmconfig* in *Section 7.3*) for the host. The ***dsmd*** process can also be started from the command-line. In either case, the *netd* process must already be running, unless the **-s** option is used. While running, the ***dsmd*** process on each host is registered exclusively with the Distributed7 environment on the local host as a daemon object called, DSM_MNGR [a macro defined in the **<*api_macro.h*>** header file].

Once the ***dsmd*** process starts, it contacts its peers across the network to find out if any DSM segment has already been created. If one has, ***dsmd*** will request information about all segments so it can synchronize itself. After it is synchronized, it starts servicing application requests placed by processes executing on the local host. Most DSM requests require communication between the ***dsmd*** processes on the individual hosts.

***Important**: The **dsmd** daemon features a built-in auditing mechanism that has been designed to ensure the consistency of various pieces of dynamic data that are associated with the DSM subsystem and maintained by the **dsmd** daemon on behalf of application processes. The frequency of these audits varies based on the nature of the data being audited (e.g., lock records are audited more frequently than other pieces of dynamic data in an effort to identify leftover locks as quick as possible). While this auditing mechanism is deemed as essential for correct operation of the DSM subsystem in unsupervised mode, users can still disable/re-enable this mechanism by sending a SIGUSR2 signal to the **dsmd** daemon process. Each time this signal is received, **dsmd** will toggle a software flag that controls whether automatic audits should be performed. By default, this flag is enabled. SIGUSR1 signal, on the other hand, causes the **dsmd** daemon to display the current values of its operational parameters. Both signals are instrumental for debugging purposes only; therefore, should not be used under normal circumstances. See dsm_audit on page 8-89 for more information on audits.*

SEE ALSO apmd, netd

7.2.10 isupd

NAME

isupd Starts the ISDN User Part daemon process.

SYNOPSIS

isupd sp# ***[-s] [-t]***

DESCRIPTION

isupd Sets up and configures the ISDN User Part (ISUP) for a particular signalling point under the local Distributed7 environment. The ***isupd*** daemon performs various Signalling Message Handling (SMH) functions required by the ISDN User Part.

sp# Identifies the signalling point on the host.

-s Activates the standalone mode of operation. When this option is specified, ***isupd*** will disable all DSM related activities, which are essential for maintaining the integrity of the ISUP user-space data in a distributed mode of operation, in order to achieve better performance.

-t Runs ***isupd*** in single-threaded mode. The default ***isupd*** mode is multi-threaded, but specifying this option will allow ***isupd*** to run in single-threaded mode.

The ***isupd*** daemon for a particular signalling point can be started automatically by the *apmd* daemon, if there is an entry for it in the configuration file. Alternatively, it can be started from the command line with this command. An ***isupd*** instance for each signalling point can exist on a host. The number of ***isupd*** instances that can coexist on a particular host machine varies, depending on the number of SP's configured on that machine.

While in operation, ***isupd*** will be registered exclusively with the Distributed7 environment on the local host as a daemon object for configuration message handling and as an SS7 object for protocol message handling. The registration name for ***isupd*** as a daemon object is ISUP_MNGR(*sp#*). Refer to the **<*api_macro.h*>** header file for a definition of this macro.

Important*: The upmd process must be running before executing this command. Use ebs_ps to confirm that these processes exist.*

FILES

$EBSHOME/access/RUN\fI<sp#>***\fR/DBfiles/isup.DB***

$EBSHOME/access/RUN\fI<sp#>***\fR/DBfiles/isupnode.DB***

$EBSHOME/access/RUN\fI<sp#>***\fR/DBfiles/isupcgrp.DB***

$EBSHOME/access/RUN\fI<sp#>***\fR/DBfiles/isupcct.DB***
$EBSHOME/access/RUN\fI<sp#>***\fR/DBfiles/isuptmr.DB***

SEE ALSO apmd, spmd, upmd, scmd

7.2.11 logd

NAME

logd Starts the Log process (LOG_MNGR).

SYNOPSIS

logd [-d] [-a|h|o] [-f logfile]

DESCRIPTION

logd Activates the LOG_MNGR process which outputs the messages logged from Distributed7 processes that have had the message logging capability activated with *ebs_log* or *spm_log()*. It is registered exclusively as a daemon object to the Distributed7 environment on the local host. By default, ***logd*** outputs the messages to the standard ouput, but can also send them to a file.

The information displayed for each message includes a message identifier, a date/time stamp, source and destination addresses, message type, message size, priority-level, and contents of the data portion of the message. The destination address information is *always* expressed in the L_IPCKEY format. The message size field contains information about both the overall message size and the size of the data portion of the message.

-d Displays additional information. An extra line of output is displayed which shows the origination point where message logging for a particular message took place, i.e., host and STREAMS multiplexer.

-a Displays message contents in ASCII format.

-h Displays message contents in hexadecimal format (default).

-o Displays message contents in octal format.

-f logfile Displays information to the standard output and also stores it in the file named ***logfile***. Without this option, ***logd*** will only display information to the standard output.
If ***logd*** is initially invoked with this option, output to the file can be enabled/disabled by sending a SIGUSR2 signal to the ***logd*** process. Each time this signal is received, ***logd*** will toggle its current log status.

SEE ALSO ebs_log, spm_log() from the API Reference Manual

7.2.12 mlogd

NAME

mlogd	Starts master event log daemon.

SYNOPSIS

mlogd [-c**|**s**|**x] [-v] [-d** dir**] [-m** msize**] [-a** asize**]

DESCRIPTION

mlogd	Starts the ***mlogd*** daemon which collects and processes log messages generated by system and application processes, messages are generated using the APM API library log macros. The ***mlogd*** daemon is normally started by the *apmd* daemon, but it could be started from the command line.
-c	Starts the AccessCRP version of ***mlogd***. The ***$PRODID*** and ***$RUNID*** environment variables must be set to appropriate values prior to program execution.
-s	Starts the AccessSERVICES version of ***mlogd***. The ***$DOMID*** environment variable must be set to an appropriate value prior to program execution.
-x	Starts the normal Distributed7 version of ***mlogd***. The ***$EBSHOME*** environment variable must be set to an appropriate value prior to program execution.
-o	This option is used to overwrite the time stamp information contained in the log message received. By default, the time stamp information is populated when the log message is submitted, i.e., at the point of origination. ***mlogd*** uses this information as is
-v	Runs ***mlogd*** in *verbose* mode. This mode causes ***mlogd*** to print the specifics of log messages received from other processes on the system console.
-d dir	Stores master/alternative log files on specified host machines. By default, master/alternative log files are stored under ***mlog*** and ***alog*** directories, respectively, in the ***$EBSHOME/access/RUN*** directory. If the ***dir*** is specified, the master/alternative log files are stored under the ***dir/mlog*** and ***dir/alog*** directories, respectively. If ***dir*** directory (or one of its sub-directories) does not exist, ***mlogd*** will make an attempt to create all necessary directories.
-m msize	Allows ***mlog*** directory to grow to ***msize*** kilobytes in size. By default, ***mlogd*** allows ***mlog*** directory to grow to 8192 kilobytes (i.e., 8 megabytes) in size before cleaning up dated ***mlog*** files.

-a asize	Allows ***alog*** directory to grow to ***asize*** kilobytes in size. By default, ***mlogd*** allows ***alog*** directory to grow to 8192 kilobytes (i.e., 8 megabytes) in size before cleaning up dated ***alog*** files.

The ***mlogd*** daemon also supports multiple versions of *apmd*. If the user does not explicitly specify one in the command, then ***mlogd*** determines the version by the following logic:

- If ***$DOMID*** is set, it assumes an AccessSERVICES environment.
- If ***$DOMID*** is not set but ***$PRODID*** and ***$RUNID*** are set, it assumes the AccessCRP (Call Routing Point) environment.
- If none of the above environment variables are set, but ***$EBSHOME*** is set, it assumes the basic Distributed7 environment.

The ***mlogd*** daemon is normally started by the *apmd* daemon as the very first process. Only if it is the first process can it collect and process ALL log messages generated by processes that are subsequently started. The ***mlogd*** analyzes the log options in the incoming log messages to send them to the specified destinations. It also formats each message. The destinations may include the master or alternate log files on the local host, the system console, the local printer, the local alarm handler daemon, and/or MMI agents such as MML and GUI. Log messages that carry the **E_MLog** and/or **E_ALog** options are stored in local log files permanently.

FILES

The master log files generated and maintained by ***mlogd*** are named according to the following convention:

$EBSHOME/access/RUN/mlog/MLog.mmddyy

The alternate [secondary] log files generated and maintained by ***mlogd*** are named as follows:

$EBSHOME/access/RUN/alog/ALog.mmddyy

The **mmddyy** string is the current month, day, and year, each expressed in a two-digit numeric format.

NOTES

mlogd will delete the oldest log file in the master log directory when the size of that directory reaches 8Mbytes to prevent excessive accumulation of log files on the system. Similarly, it will delete the oldest log file in the alternate log directory when the size of that directory reaches 4Mbytes. Users can overwrite these default settings by using the ***-m*** and/or ***-a*** command line options at start-up. For example, to accommodate 32MB of storage space for master log files and 16MB of storage space for alternate log files, one must invoke ***mlogd*** as follows:

mlogd -m 32768 -a 16384

SEE ALSO apmd, apm_logerr(), apmconfig

7.2.13 mml

NAME

mml Starts man-machine language interface

SYNOPSIS

mml [-d] [-f filename] sp

DESCRIPTION

mml Allows node/configuration management tasks on a user-specified signalling point (SP) using a customized version of the Man-Machine Language (MML) interface language. instead of the graphical user interface). When an ***mml*** session is started, the following prompt will be displayed by default: **`MML_TH>.`** The terminal handler prompts may be customized by setting the MML_PROMPT environment variable.

While in operation, ***mml*** will be registered exclusively with the Distributed7 environment on the local host machine as a daemon object, under the name MML_MNGR(sp) (a macro defined in the <***api_macro.h***> header file).

OPTIONS

-d Disable ***ksh*** technique that allows use of special keys (e.g., arrow keys) to recall previously entered MML commands at MML prompt line.

-f filename The batch process MML commands included in filename prior to returning control to user at the prompt line.

sp The logical signalling point number used with *upmd*.

FILES

$EBSHOME/access/RUN/config/MML/params.txt

$EBSHOME/access/RUN/config/MML/help.text

SEE ALSO apmd, AccessMOB, mmi

7.2.14 mmi

NAME

mmi Starts man-machine language interface

SYNOPSIS

mmi [-d] [-f filename]

DESCRIPTION

mmi Allows node/configuration management tasks on a user-specified signalling point (SP) using a customized version of the Man-Machine Language (MML) interface language. instead of the graphical user interface). ***mmi*** is a restricted version of *mml* that provides the same set of capabilities for managed objects that are not associated with a specific signalling point. Thus, one cannot use the ***mmi*** utility to perform SS7-related node/configuration management tasks. When an ***mmi*** session is started, the following prompt will be displayed by default: **`MMI_TH>.`** The terminal handler prompts may be customized by setting the MMI_PROMPT environment variable.

While in operation, ***mmi*** will be registered exclusively with the Distributed7 environment on the local host machine as a daemon object, under the name MMI_MNGR (a macro defined in the <***api_macro.h***> header file).

OPTIONS

-d Disable ***ksh*** technique that allows use of special keys (e.g., arrow keys) to recall previously entered MMI commands at MMI prompt line.

-f filename The batch process MMI commands included in filename prior to returning control to user at the prompt line.

FILES

$EBSHOME/access/RUN/config/MMI/params.txt

$EBSHOME/access/RUN/config/MMI/help.text

SEE ALSO apmd, AccessMOB, mml

7.2.15 netd

NAME

netd Starts TCP/IP network daemon.

SYNOPSIS

$EBSHOME/access/bin/netd [-d|n] [-i] [-s]

DESCRIPTION

netd Starts the ***netd*** daemon process, which sets up and maintains the STREAMS-based kernel-level interfaces from the local host's SPM to the SPMs of remote hosts in a distributed Distributed7 environment. The ***netd*** daemon process registers exclusively with the Distributed7 environment on the local host as the daemon object, NET_MNGR. This daemon process is required only for distributed configurations.

The kernel-level interface is built upon revision 1.5 of the Transport Provider Interface (TPI) specifications, and utilizes the connection-oriented TCP/IP protocol suite. This interface allows the processes that operate in the distributed network to exchange inter-machine messages in a reliable and high-performance manner.

This daemon is responsible for controlling the operations of distribution related NTWK, HOST, and TCPCON managed objects. The distributed processing environment, such as remote hosts and the TCP/IP connections between hosts, is defined by these managed objects. A distributed configuration can be a single TCP/IP connection between two hosts, or a maximum of eight host machines and/or redundant LAN configurations in which dual TCP/IP connections are set up between each and every host included in the network.

The ***netd*** daemon can be started automatically by the *apmd* daemon at system start-up time, provided that there is an entry in the ***$EBSHOME/access/RUN/config/PMGR/apmconfig*** file. Alternatively, it can be started manually from the command line.

The EBSHOME environment variable must be set before invoking this daemon as it makes use of this variable to locate various configuration files.

-d This option is used to disable the built-in LAN interface testing feature of the ***netd*** daemon. When disabled, Distributed7 software cannot effectively detect and/or act upon failures, such as those resulting from disconnected or cut LAN cables. By default, this feature is enabled.

i This function indicates that the Solaris IP Network Multipathing (IPMP) feature is in use; therefore, operations that involve LAN interfaces will need to be closely coordinated between Distributed7 and IPMP software (e.g., detection of failure or repair of a failed LAN interface).

The IPMP feature was introduced in Solaris 8 OE update 2 (10/00) release and it enables a host machine to have multiple network ports connected to the same subnet. This capability coupled with multiple network connections per subnet provide a host with one or both of the following advantages: [1] resilience from network adapter failure; [2] increased data throughput for outbound traffic. Since IPMP feature is intended to provide fully transparent recovery at TCP layer, when this option is specified, Distributed7 will not make any attempts to recover from single LAN interface failures (i.e., since IPMP software is expected to be in charge of "failover" and "failback" operations).

Note that one may need to fine tune the duration of "failure detection time" used by IPMP software and/or TCP/IP heartbeat interval in use by Distributed7 software across individual TCP/IP connections to ensure no heartbeat failure would occur during so-called "failover blackout" periods. This can be achieved by taking one of the following approaches: [1] reducing "failure detection time" value in use by IPMP software to a value less than five times the length of TCP/IP heartbeat interval in use; [2] increasing the length of TCP/IP heartbeat interval in use to be more than one fifth of the "failure detection time" value in use by IPMP software.

-n This option is used to enable optional ***ndd*** checks in an effort to detect dual-LAN cable disconnects more reliably and quickly. By default, Distributed7 relies on “cable disconnected/problem” messages reported through the STREAMS log to detect LAN cable disconnects. There are times, however, when dual-LAN cable disconnects are neither reliably nor quickly reflected in the log messages. This option instructs the ***netd*** daemon to conduct an additional set of checks using the ***ndd*** utility to detect dual-LAN cable disconnects more reliably and quickly.
Note that this option can be used only for newer type LAN interfaces, such as /dev/hme or /dev/qfe, and does not support older interface types, such as /dev/1e.

-s This option allows ***netd*** to execute in stand-alone mode. In this mode of operation, ***netd*** bars TCP/IP connections to remote hosts.

FILES

$EBSHOME/access/RUN/DBfiles/net_ntwk.DB

$EBSHOME/access/RUN/DBfiles/net_host.DB

$EBSHOME/access/RUN/DBfiles/net_tcpcon.DB

SEE ALSO spmd**,** apmd**,** ebs_sync, ebs_showlink, ebs_hbeat,

7.2.16 rtc_agent

NAME

rtc_agent Remote TCAP agent

SYNOPSIS

rtc_agent sp ssn

DESCRIPTION

rtc_agent This daemon process sets up and maintains remote TCAP connections under a Distributed7 environment, as necessitated by the *tcm_rmtopen()* function call.

The ***rtc_agent*** daemon process is always started indirectly, i.e., as a result of an indirect ***apm_spawn()*** call, on behalf of the TC application invoking the *tcm_rmtopen()* function call. It must therefore never be started manually from the command line.

On each front-end host machine, only a single instance of the ***rtc_agent*** daemon process may exist for a given SP/SSN pair, regardless of the number of remote TC applications running on the back-end machines. This means that the ***rtc_agent*** daemon process is spawned when the *tcm_rmtopen()* function is invoked by a remote TC application for the very first time, and that it is not terminated until all instances of that TC application terminate—at which time ***rtc_agent*** terminates automatically as well. Thus, it is perfectly normal for the ***rtc_agent*** daemon process on a front-end machine to serve the needs of multiple TC applications running on one or more back-end machines at a given time.

The primary function of the ***rtc_agent*** daemon process is to perform message routing between the TCAP layer software running on back-end machines and the SCCP layer software running on the front-end machine.

For outgoing messages, i.e., messages originated by remote TC applications and destined to the SS7 network, ***rtc_agent*** injects the message into the SCCP layer on the local host, from where it is transported to its final destination. In the reverse direction, ***rtc_agent*** is responsible only for relaying the message to the TCAP layer running on one of the back-end machines. The actual delivery of incoming messages to an appropriate TC application on an appropriate back-end machine remains as the responsibility of the TCAP layer. However, it is possible to establish some level of control over the message routing policy of the ***rtc_agent*** daemon process in the incoming direction, i.e., for messages received from the network, when invoking the *tcm_rmtopen()* call as follows:

- If the ***rtmuser*** field of the ***tcmopts_t*** data type is set to a value of L_TC_RMTUSER_PRIMARY, the registering TC application is considered a primary candidate to receive incoming TCAP messages through the ***rtc_agent*** daemon process.

- If the ***rmtuser*** field of the ***tcmopts_t*** data type is set to a value of L_TC_RMTUSER_SECONDARY, the registering TC application is considered a non-primary candidate, and is not normally delivered any incoming TCAP messages through the ***rtc_agent***.

Note that no restrictions are imposed by Distributed7 on the maximum number of primary and/or secondary remote TC applications that can co-exist at a given time. Thus, it is up to the TC application to impose any such restrictions.

The specifics of the message routing policy employed by the ***rtc_agent*** are as follows:

- Search for a primary TC application first.
- If more than one primary instance exists, use round-robin selection criteria to select which primary instance receives the incoming TCAP message.
- If no primary instance exists, search for a secondary instance.
- If more than one secondary instance exists, use round-robin selection criteria to select which secondary instance receives the incoming TCAP message.

While in operation, ***rtc_agent*** is registered exclusively with the Distributed7 environment on the local host machine as a daemon process object under the name RTC_MNGR—a macro defined in the **<*api_macro.h*>** header file.

sp This argument identifies the signalling point associated with the remote TC application invoking the *tcm_rmtopen()* call.

ssn This argument identifies the SCCP subsystem number associated with the remote TC application invoking the *tcm_rmtopen()* call.

SEE ALSO apmd, apm_spawn(), tcm_rmtopen(), **rtc_dump,** and **rtc_stat**

7.2.17 scmd

NAME

scmd Initializes the Signalling Connection Control Part (SCCP) multiplexer.

SYNOPSIS

scmd sp#

DESCRIPTION

scmd Starts the SS7 daemon process, ***scmd***, which initializes and administers the SCCP for a signalling point on the local system.

sp# Identifies the logical signalling point number of the node on the system to be configured; should be the same one used with the related *upmd* command.

The ***scmd*** daemon registers exclusively with the Distributed7 environment on the local host machine as the daemon object, SCM_MNGR(*sp#*). An ***scmd*** instance for each signalling point can exist on a host.

The ***scmd*** daemon initializes the SCCP multiplexer, links it to the UPM, and downloads it with the configuration information from the configuration database files located in the ***$EBSHOME/access/RUN\fI***<sp#>***\fP/DBfiles\fR*** directory. If no pre-configured data exists in the database, the following warning message occurs:

There is no spc in the database

The SCCP multiplexer provides routing control, connectionless control, connection-oriented control, and management functions required by the SCCP protocol layer.

The ***scmd*** daemon for a particular ***sp#*** can be started automatically by the *apmd* by making an entry in the associated configuration file (such as *apmconfig*). It can also be started from the command line.

Important: *The upmd process must be running before executing this command. Use ebs_ps to confirm that they exist.*

NOTES

The ***scmd*** daemon requires the ***upmd*** daemon for the corresponding signalling point to be up and running. It also requires the MTP protocol for that signalling point to be configured using the Distributed7 Managed Object interface, i.e., by manipulating the MTP managed object parameters. If the MTP protocol is not configured when the ***scmd*** daemon is started, ***scmd*** will suspend its execution indefinitely until this operation is completed before it proceeds with its normal start-up procedures. Note that under such circumstances, it will not be possible to see the ***scmd*** entry in the ***ebs_ps*** listing until the MTP protocol is configured.

FILES

$EBSHOME/access/RUN<sp#>***/DBfiles/snsp.DB***

$EBSHOME/access/RUN<sp#>***/DBfiles/subsys.DB***

$EBSHOME/access/RUN<sp#>***/DBfiles/cpc.DB***

$EBSHOME/access/RUN<sp#>***/DBfiles/mate.DB***

$EBSHOME/access/RUN<sp#>***/DBfiles/gtentry.DB***

SEE ALSO apmd, upmd

7.2.18 spmd

NAME

spmd Starts the Distributed7 infrastructure.

SYNOPSIS

spmd

DESCRIPTION

spmd Initializes and sets up the foundation of the multi-layered STREAMS architecture required by the Distributed7 infrastructure.

The ***spmd*** daemon configures the device drivers, TRMOD module, and Service Provider Module (SPM, also known as the signalling point multiplexer). The ***spmd*** daemon registers exclusively with the Distributed7 environment on the local host as a daemon object called SPM_MNGR.

On start-up, the spmd daemon first opens and initializes the SPM multiplexer to establish soft links to the SS7 signalling board device drivers (sbs334, pci334, pci3xpq, pci3xapq, cpc3xpq, pmc8260, artic8260, adaxm, and pmc4539) as they are configured using the MMI/NMI interfaces. The ***spmd*** also assumes the responsibility of removing these links during system software shutdown.

The ***spmd*** daemon can be started automatically by the *apmd* process at system start-up time provided that there is an entry in the ***$EBSHOME/access/RUN/config/PMGR/apmconfig*** file. It can also be started from the command line.

Note: The apmd process is now responsible for process management.

FILES

$EBSHOME/access/RUN/DBfiles/spm_ss7board.DB

$EBSHOME/access/RUN/DBfiles/spm_class.DB

$EBSHOME/access/RUN/DBfiles/spm_port.DB
$EBSHOME/access/drv/sal.*.rel
$EBSHOME/access/drv/mtpl2.*.rel

SEE ALSO apmd, upmd, netd, apm_start, scmd

7.2.19 tcmd

NAME

tcmd Sets up the TCAP multiplexer.

SYNOPSIS

tcmd

DESCRIPTION

tcmd Sets up and configures the Transaction Capabilities Application Part (TCAP) multiplexers under the local Distributed7 environment. The TCAP multiplexer provides dialogue-oriented functions for use by the Transaction Capability (TC) application processes.

The TCAP multiplexer of Distributed7 provides run-time support for different TCAP variants (ANSI or CCITT) and allows TCAP applications to exchange TCAP messages using SCCP or TCP/IP protocols. The TCAP multiplexer must be installed and the ***tcmd*** process must be running to use the capabilities of the Distributed7 TCAP library, ***libtcap.a***. This library also supports both TCAP variants and both the SCCP and TCP/IP transport service providers. It replaces the TCAP libraries of earlier releases.

The ***tcmd*** process can be started from the command line or automatically by the *apmd* system process. To start ***tcmd*** automatically, the command must be an entry in the *apmd* configuration file (e.g. *apmconfig* in Section 7.3). While running, the ***tcmd*** process is registered exclusively with the Distributed7 environment on the local host as a daemon object called, TCM_MNGR [a macro defined in the **<*api_macro.h*>** header file].

Important*: The previous version of the TCAP library (**libatcap.a** and **libctcap.a**) is still supported for backward compatibility, only. It may be discontinued in future releases of the Distributed7 product.*

7.2.20 upmd

NAME

upmd Sets up the User Part Multiplexer.

SYNOPSIS

upmd sp#

DESCRIPTION

upmd Sets up and configures the User Part Multiplexer (UPM) for a particular Signalling Point (sp) under the local Distributed7 environment. The primary responsibility of the UPM is to perform various Signalling Message Handling (SMH) functions and Signalling Network Management (SNM) functions required by the Message Transfer Part (MTP) Layer 3 protocol.

sp# Signalling point number of the logical node. It is usually 0 for a single or first node, but can be 1, 2, 3, 4, 5, 6, or 7. When the INE feature is being used, up to eight logical nodes can be started and configured, but all must have different signalling point numbers.

On start-up, the ***upmd*** daemon opens and initializes the appropriate UPM multiplexer and links it to the bottom of the previously initialized SPM.

After start-up, ***upmd*** interacts with the other ***upmd*** instances across the network, if there are any, in an effort to synchronize its local database files so that the signalling points on individual machines start from the same copy of the database.

The ***upmd*** subsequently continues its life by monitoring various events associated with the corresponding UPM, handling MMI/NMI requests that may be initiated by the users, and taking appropriate actions.

While in operation, the ***upmd*** daemon on each host registers exclusively with the Distributed7 environment on that host as the daemon object, UPM_MNGR(*sp#*) (a macro defined in <***api_macro.h***>). A ***upmd*** instance for each signalling point can exist on a host.

The ***upmd*** daemon for a particular ***sp#*** can be started automatically by the *apmd* by making an entry in the associated configuration file (such as *apmconfig*). It can also be started from the command line.

Important: *The **$EBSHOME** environment variable must be set before invoking this daemon because the variable is used to locate the MTP database files.*

FILES

$EBSHOME/access/RUN<sp#>***/DBfiles/mtp_l3timer.DB***

$EBSHOME/access/RUN<sp#>***/DBfiles/mtp_link.DB***

$EBSHOME/access/RUN<sp#>***/DBfiles/mtp_linkstat.DB***
$EBSHOME/access/RUN<sp#>***/DBfiles/mtp_lset.DB***
$EBSHOME/access/RUN<sp#>***/DBfiles/mtp_lsetstat.DB***
$EBSHOME/access/RUN<sp#>***/DBfiles/mtp_route.DB***
$EBSHOME/access/RUN<sp#>***/DBfiles/mtp_rsidx.DB***
$EBSHOME/access/RUN<sp#>***/DBfiles/mtp_rtset.DB***
$EBSHOME/access/RUN<sp#>***/DBfiles/mtp_sltimer.DB***
$EBSHOME/access/RUN<sp#>/Dbfiles/mtp_alias.DB
$EBSHOME/access/RUN<sp#>***/DBfiles/mtp_sp.DB***

SEE ALSO apmd, scmd

7.3 Configuration Files

7.3.1 apmconfig

NAME

apm_config apmd configuration file

SYNOPSIS

$EBSHOME/access/RUN/config/PMGR/apmconfig

DESCRIPTION

tag:estate:action:sstate:fstate:hstate:astate:nstate:acktime:retrycnt:retryint:
retrydel:hbeatint:progid:groupid:priclass:clparams:dirpath:process

Configuration file which controls the operations of the *apmd* daemon process. It is composed of individual entries with the position-dependent syntax defined in the synopsis. The fields of the entry must be separated by a **:** character, with no white space in the line. Each entry is delimited by a *newline*. A maximum of 512 characters are permitted for each entry. Comments may be inserted as separate lines using the convention for comments described in *sh(1)*. A sample of the default file is listed on page 7-46.

Important*: The first statement in the file must be* ***version=1.0.0*** *or version 1.2.0. The apmd process must know that the entries in the file follow Release 1.x.y conventions.*

When creating and/or modifying the file, the following guidelines should be observed:

- Each line must contain the correct number of fields and must be formatted correctly. If a line is incorrect, then *apmd* ignores the entries in the file. However, it logs the line number at which an error or inconsistency was encountered.
- The UNIX path names for the executables are valid and correct.
- The start-up and steady-state information provided for individual entries must not conflict with other entries, which could cause undesired loops in processing.
  The apmd process is as intelligent as the logic provided in the apmconfig file!
- Shell scripts which are invoked from an entry must have a statement in the first line that specifies which shell version to be invoked (e.g., #!/bin/sh to invoke plain UNIX shell).
- After making any changes to the contents of the file while the system is running, you must issue the *apm_update* command to notify the *apmd* daemon to re-read and execute the file.

There are no limits on the number of entries. The entry fields are defined as follows:

tag An alphanumeric string used to uniquely identify a process in the **id@host** format when operating in a network of hosts. The **id** is an alphanumeric string that uniquely identifies the process on the host machine identified by **host**. The **@host** portion may be omitted for processes on the local host because the system will automatically substitute this information. The total size of the **id@host** string cannot exceed 24 characters.

estate The state(s) that the *apmd* must be in for this entry to be executed. If the *apmd*'s run state while executing this file is among the states specified in this field, the entry is executed. Otherwise, it is ignored. A maximum of 16 execution states may be specified for an individual entry. Multiple execution states must be separated from each other with only the **|** character - no *white space* should exist. If no states are defined for **estate**, then the entry will be executed every time *apmd* executes the file. Execution states are defined by the developer and may have names that are up to 4 alphanumeric characters long.

action The action(s) that *apmd* should take on the process identified in the process field. Several key words exist that are recognized by *apmd*. Actions are only taken if *apmd*'s run state matches a state in the **estate** field. Valid actions are:

- **initdefault**: An entry with this action is scanned only once when *apmd* is initially invoked. This entry is used to determine the initial run state for *apmd*. This initial run state is set to the first execution state specified in the **estate** field of this entry. If the ***apmconfig*** file does not contain an entry with the *initdefault* action type, *apmd* determines its initial run state as follows:
  for the AccessSERVICES version, state **A**
  for the AccessCRP version, state **D**
  for the basic Distributed7 version, state **init**
- **once**: If the process named in the **process** field is not running, *apmd* should start it and proceed to the state specified in appropriate state field. The *apmd* should not wait for the process's termination and it should not restart the process if the process terminates. After starting the process, *apmd* will go to the state specified in one of the following *state* fields. The *state* field it will access depends on the contents of **acktime** and actions of the process. Based on this combination of results, each state field should hold an appropriate *apmd* run state.
  astate - 1) **acktime** field holds a non-zero value and a positive acknowledgment was received from the process
  2) no acknowledgment is expected (**acktime** is 0 or empty)
  nstate - 1) negative acknowledgment was received from process
  2) no acknowledgment was received within the time specified in **acktime**
  sstate - 1) process terminates with an exit code of 0, at any time
  fstate - 1) process terminates with a non-zero exit code, at any time
  2) process is killed by a signal it could not handle
- **auto**: If the process named in the **process** field is not running, *apmd* should start it and proceed to the state specified in appropriate state

field. The *apmd* should not wait for the process's termination. From then on, provided the process terminates its execution within **retrydel** seconds, restart the process at periodic intervals as specified via the **retrydel** setting and repeat this pattern forever. A missing or zero value of **retrydel** makes this action type equivalent to "once" and disables the periodic start-up capability. Similarly, at any point in time, if the process fails to terminate its execution within **retrydel** seconds, the periodic start-up capability is automatically disabled. After starting the process, *apmd* will go to the state specified in one of the following *state* fields. The *state* field it will access depends on the contents of **acktime** and actions of the process. Based on this combination of results, each state field should hold an appropriate *apmd* run state.

<u>astate</u> - 1) **acktime** field holds a non-zero value and a positive acknowledgment was received from the process
2) no acknowledgment is expected (**acktime** is 0 or empty)

<u>nstate</u> - 1) negative acknowledgment was received from process
2) no acknowledgment was received within the time specified in **acktime**

<u>sstate</u> - 1) process terminates with an exit code of 0, at any time

<u>fstate</u> - 1) process terminates with a non-zero exit code, at any time
2) process is killed by a signal it could not handle

The exit code of the process at termination time has no impact on the periodic start-up capability. As long as the process somehow manages to terminate its execution within **retrydel** seconds, an attempt will be made by *apmd* to re-start it at the end of **retrydel** seconds.

- **failsafe**: If the process named in the **process** field is not running, *apmd* should start it and proceed to the state specified in appropriate state field. The *apmd* should not wait for the process's termination but it should restart the process if start-up fails (fstate). The process should not be restarted if it exits with a zero exit code. After initially starting the process, *apmd* will go to the state specified in one of the following *state* fields. The *state* field it will access depends on the contents of **acktime** and actions of the process. Based on this combination of results, each state field should hold an appropriate *apmd* run state.

<u>astate</u> - 1) **acktime** field holds a non-zero value and a positive acknowledgment was received from the process
2) no acknowledgment is expected (**acktime** is 0 or empty)

<u>nstate</u> - 1) negative acknowledgment was received from process
2) no acknowledgment was received within the time specified in **acktime**

<u>sstate</u> - 1) process terminates with an exit code of 0, at any time (do not restart process)

fstate - 1) process terminates with a non-zero exit code (at any time)
2) process is killed by a signal it could not handle
The *apmd* should keep attempting to restart the process until it starts or until the number of attempts that occur in **retryint** seconds exceeds **retrycnt**. A delay of **retrydel** seconds should occur between attempts.

hstate - 1) number of attempts to restart process in the last **retrydel** seconds has exceeded the value in **retrycnt**
The cycle of restart attempts described in *fstate* should not begin for 60 seconds.

- **respawn**: If the process named in the **process** field is not running, *apmd* should start it and proceed to the state specified in the appropriate state field. The *apmd* should not wait for the process's termination but it should restart the process if start-up fails (fstate). The process should not be restarted if it exits with a zero exit code. After initially starting the process, *apmd* will go to the state specified in one of the following *state* fields. The *state* field it will access depends on the contents of **acktime** and actions of the process. Based on this combination of results, each state field should hold an appropriate *apmd* run state.

  astate - 1) **acktime** field holds a non-zero value and a positive acknowledgment was received from the process
  2) no acknowledgment is expected (**acktime** is 0 or empty)

  nstate - 1) negative acknowledgment was received from process
  2) no acknowledgment was received within the time specified in **acktime**

  sstate - 1) process terminates with an exit code of 0, at any time (do not restart process)

  fstate - 1) process terminates with a non-zero exit code (at any time)
  2) process is killed by a signal it could not handle
  The *apmd* should keep attempting to restart the process until it starts or until the number of attempts that occur in **retryint** seconds exceeds **retrycnt**. A delay of **retrydel** seconds should occur between attempts.

  hstate - 1) number of attempts to restart process in the last **retrydel** seconds has exceeded the value in **retrycnt**
  No further attempts to restart the process will be made.

- **wait**: If the process named in the **process** field is not running, *apmd* should start it and wait for it to terminate. After the process terminates, it can proceed to the state specified in the appropriate state field. (No other activities will occur until the process terminates.) The *state* field it will access depends on the contents of **acktime** and actions of the process. Based on this combination of results, each state field should hold an appropriate *apmd* run state.

astate - 1) **acktime** field holds a non-zero value and a positive acknowledgment was received from the process
2) no acknowledgment is expected (**acktime** is 0 or empty)
nstate - 1) negative acknowledgment was received from process
2) no acknowledgment was received within the time specified in **acktime**
sstate - 1) process terminates with an exit code of 0, at any time
fstate - 1) process terminates with a non-zero exit code, at any time
2) process is killed by a signal it could not handle

- **off**: If the process named in the **process** field is currently running, generate a SIGTERM signal to terminate it. If the process does not terminate within the next 3 seconds, send a SIGKILL signal to terminate it. Then, switch to state specified in **sstate** field. If the process is not running, ignore this entry.
- **setstate**: Change the current run state of *apmd* to the state specified in the **sstate** field.

sstate The success state that *apmd* will be set to when the process terminates with a zero exit code. Two separate states, start-up and steady-state, may be defined in this field. The start-up success state is only used when *apmd* starts up, i.e., the ***apmconfig*** file is being executed for the first time. From then on, only the steady-state success state is used.
Each state is defined by the developer and may have a name that is up to 4 alphanumeric characters long. The first state in the field is the start-up state, followed by the steady-state. The two success states must be separated from each other by the **-** character, with no white space. A single state in the field implies both the start-up state and steady-state are the same. Either or both fields may be set to *don't care*. To set the start-up state to *don't care*, this field must contain the **-** character followed by the steady-state success state. To set the steady-state success state to *don't care*, this field must contain the start-up state followed by the **-** character. If this field is empty, then both start-up and steady-state success states are *don't care*.

fstate The failure state that *apmd* will be set to when the process terminates with a non-zero exit code or the process terminates because of a signal it could not handle. Two separate states, start-up and steady-state, may be defined in this field. The start-up failure state is only used for failures when *apmd* initially starts up, i.e., first time ***apmconfig*** file is executed. From then on, only the steady-state failure state is used.
Each state is defined by the developer and may have a name that is up to 4 alphanumeric characters long. The first state in the field is the start-up state, followed by the steady-state. The two failure states must be separated from each other by the **-** character, with no white space. A single state in the field implies both the start-up state and steady-state are the same. Either or both fields may be set to *don't care*. To set the start-

up state to *don't care*, this field must contain the **-** character followed by the steady-state failure state. To set the steady-state failure state to *don't care*, this field must contain the start-up state followed by the **-** character. If this field is empty, then both start-up and steady-state failure states are *don't care*.

hstate The hopeless state that *apmd* will be set to when **retrycnt** successive attempts to restart the process fail within the **retryint** interval. Two separate states, start-up and steady-state, may be defined in this field. The start-up hopeless state is only used when *apmd* initially starts up, i.e., first time ***apmconfig*** file is executed. From then on, only the steady-state hopeless state is used.

Each state is defined by the developer and may have a name that is up to 4 alphanumeric characters long. The first state in the field is the start-up state, followed by the steady-state. The two hopeless states must be separated from each other by the **-** character, with no white space. A single state in the field implies both the start-up state and steady-state are the same. Either or both fields may be set to *don't care*. To set the start-up state to *don't care*, this field must contain the **-** character followed by the steady-state hopeless state. To set the steady-state hopeless state to *don't care*, this field must contain the start-up state followed by the **-** character. If this field is empty, then both start-up and steady-state hopeless states are *don't care*.

astate The positive acknowledgment state that *apmd* will be set to when a positive acknowledgment is received from the process during the acknowledgment interval or when no acknowledgment message is expected. Two separate states, start-up and steady-state, may be defined in this field. The start-up state is only used when *apmd* initially starts up, i.e., first time ***apmconfig*** file is executed. From then on, only the steady-state acknowledgment state is used.

Each state is defined by the developer and may have a name that is up to 4 alphanumeric characters long. The first state in the field is the start-up state, followed by the steady-state. The two states must be separated from each other by the **-** character, with no white space. A single state in the field implies both the start-up state and steady-state are the same. Either or both fields may be set to *don't care*. To set the start-up state to *don't care*, this field must contain the **-** character followed by the steady-state acknowledgment state. To set the steady-state acknowledgment state to *don't care*, this field must contain the start-up state followed by the **-** character. If this field is empty, then both start-up and steady-state acknowledgment states are *don't care*.

nstate The negative acknowledgment state that *apmd* will be set to when a negative acknowledgment is received from the process or when the acknowledgment interval expires without an acknowledgment being received. Two separate states, start-up and steady-state, may be defined in this field. The start-up state is only used when *apmd* initially starts up, i.e., first time ***apmconfig*** file is executed. From then on, only the steady-state negative acknowledgment state is used.

Each state is defined by the developer and may have a name that is up to 4 alphanumeric characters long. The first state in the field is the start-up state, followed by the steady-state. The two states must be separated from each other by the **-** character, with no white space. A single state in the field implies both the start-up state and steady-state are the same. Either or both fields may be set to *don't care*. To set the start-up state to *don't care*, this field must contain the **-** character followed by the steady-state negative acknowledgment state. To set the steady-state negative acknowledgment state to *don't care*, this field must contain the start-up state followed by the **-** character. If this field is empty, then both the start-up state and steady-state are *don't care*.

acktime The time interval, in seconds, during which *apmd* should wait for a positive or negative acknowledgment from the process before moving to a run state. A value of 0 or an empty field means that *apmd* will automatically move to the run state specified in the **astate** field and will not wait for an acknowledgment from the process. A value of -1 means that *apmd* must wait indefinitely for an acknowledgment.

retrycnt The number of times *apmd* should try to restart the process within **retryint** seconds. Restarts are attempted only if the **action** field is *respawn* or *failsafe* and the process has terminated with a non-zero exit code or was killed by an unexpected signal. If this field is left blank, the default value of 3 is used.

retryint The time interval, in seconds, during which *apmd* should try to restart the process, up to **retrycnt** number of times. Restarts are attempted only if the **action** field is *respawn* or *failsafe* and the process has terminated with a non-zero exit code or was killed by an unexpected signal. If this field is left blank, the default value of 60 seconds is used.

retrydel The time delay, in seconds, that should occur between attempts to restart the process. Restarts are attempted only if the **action** field is *respawn* or *failsafe* and the process has terminated with a non-zero exit code or was killed by an unexpected signal. If this field is left blank, the default value of 1 second is used.

hbeatint The time interval, in seconds, that should exist between heartbeat request messages sent to the process by *apmd*. If no value or a value of 0 is specified in this field, the heartbeat feature is disabled for the process. The heartbeat feature sends periodic heartbeat messages to the process if it is enabled. If the process fails to respond to 3 consecutive heartbeat request messages, apmd terminates the process by sending it a SIGKILL signal. Then, when the process terminates, the *apmd* will go to the state specified in the **fstate** field.

progid The program ID associated with the process. It can be an integer value between 0 and 255. If this field is empty, the default value of 0 will be assigned to the process. The program ID is used to identify program output in trace and log records. It should be the same as the one specified in the program using the ***apm_init()*** function call because the value in this field will overwrite the one specified by the function.

groupid The group ID associated with the process. Group IDs are non-negative integer values that define *functional* process groups. If this field is empty, *apmd*'s UNIX group ID is assigned to the process by default.
The group ID allows a group of processes to be targeted by the ***apm_kill()*** function and ***apm_kill()*** utility. They can send UNIX signals to all the processes of a group at once instead of having to send to each process separately.

priclass The priority class associated with the process. When either of the **real-time** or **time-sharing** classes is selected, *apmd* daemon will invoke the ***priocntl*** system call to adjust the scheduling class and class-specific scheduling parameters of the process spawned as specified via ***priclass*** and ***clparams*** fields.
This ***priclass*** optional field allows one of the following scheduling classes to be selected by the process:

- **rt**
  The real-time scheduling class.
- **ts**
  The time-sharing scheduling class.

clparams The individual parameters associated with the ***priclass*** field. This optional field allows the individual parameters associated with a specific priority class (i.e., **real-time** or **time-sharing**) to be initialized. Multiple parameters must be separated from each other using the '|' character, with no white space left in between.

- When ***priclass*** is set to **real-time**, the ***clparams*** field must contain, in specified order, the rt_pri|rt_tqsecs|rt_tqnsecs parameter values.
- When ***priclass*** is set to **time-sharing**, the ***clparams*** field must contain, in specified order, the ts_uprilim|ts_upri parameter values.

dirpath The UNIX path name of the executable program. This field is used with the **process** field to fully identify the executable. Either a keyword (home or run), a full UNIX path name (starting with the / character) or a UNIX environment variable that is part of the current execution environment must be in this field. The keywords have the following meanings:

- **home**
  For AccessSERVICES version: ***$EBSHOME/access***
  For CRP version: ***$CRPDIR***
  For basic Distributed7 version: ***$EBSHOME/access***

- **run**
  For AccessSERVICES version: ***$EBSHOME/access/RUN***
  For CRP version: ***$APPHOME***
  For basic Distributed7 version: ***$EBSHOME/access/RUN***

process The relative path name of the executable program and its command-line arguments, if any. This field is used with the **dirpath** field to locate the UNIX path name for the executable program, the executable name, and its arguments. The maximum number of command-line arguments that can be specified is 32.
Input/output redirection is not currently supported.

SEE ALSO apmd, apm_update, apm_getstate, apm_setstate, apmconfig.old, priocntl

SAMPLE FILE

```
#
# apmconfig(4A) - configuration file for apmd(1A) daemon
#
# entries in this file must comply with the following format:
#
# tag:estate:action:sstate:fstate:hstate:astate:nstate: \
# acktime:retrycnt:retryint:retrydel:hbeatint:progid:groupid:dirpath:process
#
# where the individual fields are defined as follows:
#
#      tag          - process identifier tag [must be unique]
#      estate       - execution state(s)
#      action       - action type
#      sstate       - success state(s)
#      fstate       - failure state(s)
#      hstate       - hopeless state(s)
#      astate       - ack received state(s)
#      nstate       - nak received state(s)
#      acktime      - time to wait for ack/nak [in seconds]
#      retrycnt     - # times to respawn within `retryint` interval
#      retryint     - respawn interval [in seconds]
#      retrydel     - delay before a respawn attempt [in seconds]
#      hbeatint     - heartbeat interval [in seconds]
#      progid       - process program id
#      groupid      - process group id
#      priclass     - priority class
#      clparams     - parameters of priority class
#      dirpath      - directory at which the process is located
#      process      - executable name & arguments
#
# for more info, pls refer to the apmconfig(4A) man page.
#

# specify apmcfgfile(4A) version
version=1.0
```

```
# specify apmd(1A) start-up run state
is:init:initdefault:

# specify rules for daemons that must exist at all times
mlogd::failsafe:::::-1::::60:1::home:./bin/mlogd -x
spmd::failsafe:::::-1::::60:2::home:./bin/spmd
netd::failsafe:::::-1::::60:3::home:./bin/netd
alarmd::failsafe:::::-1::::60:4::home:./bin/alarmd
dsmd::failsafe:::::-1::::60:5::home:./bin/dsmd
dkmd::failsafe:::::-1::::60:6::home:./bin/dkmd -m dramod
# change apmd(1A) run state from "init" to "safe"
sc:init:setstate:safe-:

#
# start daemon processes associated with signalling point 0
# when a run state of "sp0u" is explicitly specified by the user
#
upmd0:sp0u:respawn::sp0d:sp0d::sp0d:-1::::60:7:100:home:./bin/upmd 0
scmd0:sp0u:respawn::sp0d:sp0d::sp0d:-1::::60:8:100:home:./bin/scmd 0

#
# terminate daemon processes associated with signalling point 0
# when a run state of "sp0d" is explicitly specified by the user
# and/or a failure is encountered during the start-up phase
#
scmd0:sp0d:off::::::::::::home:./bin/scmd 0
upmd0:sp0d:off::::::::::::home:./bin/upmd 0

#
# start daemon processes associated with signalling point 1
# when a run state of "sp1u" is explicitly specified by the user
#
upmd1:sp1u:respawn::sp1d:sp1d::sp1d:-1::::60:7:101:home:./bin/upmd 1
scmd1:sp1u:respawn::sp1d:sp1d::sp1d:-1::::60:8:101:home:./bin/scmd 1

#
# terminate daemon processes associated with signalling point 1
# when a run state of "sp1d" is explicitly specified by the user
# and/or a failure is encountered during the start-up phase
#
scmd1:sp1d:off::::::::::::home:./bin/scmd 1
upmd1:sp1d:off::::::::::::home:./bin/upmd 1

#
# start daemon processes associated with signalling point 2
# when a run state of "sp2u" is explicitly specified by the user
#
upmd2:sp2u:respawn::sp2d:sp2d::sp2d:-1::::60:7:102:home:./bin/upmd 2
scmd2:sp2u:respawn::sp2d:sp2d::sp2d:-1::::60:8:102:home:./bin/scmd 2

#
```

```
# terminate daemon processes associated with signalling point 2
# when a run state of "sp2d" is explicitly specified by the user
# and/or a failure is encountered during the start-up phase
#
scmd2:sp2d:off:::::::::::::home:./bin/scmd 2
upmd2:sp2d:off:::::::::::::home:./bin/upmd 2

#
# start daemon processes associated with signalling point 3
# when a run state of "sp3u" is explicitly specified by the user
#
upmd3:sp3u:respawn::sp3d:sp3d::sp3d:-1::::60:7:103:home:./bin/upmd 3
scmd3:sp3u:respawn::sp3d:sp3d::sp3d:-1::::60:8:103:home:./bin/scmd 3

#
# terminate daemon processes associated with signalling point 3
# when a run state of "sp3d" is explicitly specified by the user
# and/or a failure is encountered during the start-up phase
#
scmd3:sp3d:off:::::::::::::home:./bin/scmd 3
upmd3:sp3d:off:::::::::::::home:./bin/upmd 3

#
# start daemon processes associated with signalling point 4
# when a run state of "sp4u" is explicitly specified by the user
#
upmd4:sp4u:respawn::sp4d:sp4d::sp4d:-1::::60:7:104:home:./bin/upmd 4
scmd4:sp4u:respawn::sp4d:sp4d::sp4d:-1::::60:8:104:home:./bin/scmd 4

#
# terminate daemon processes associated with signalling point 4
# when a run state of "sp4d" is explicitly specified by the user
# and/or a failure is encountered during the start-up phase
#
scmd4:sp4d:off:::::::::::::home:./bin/scmd 4
upmd4:sp4d:off:::::::::::::home:./bin/upmd 4

#
# start daemon processes associated with signalling point 5
# when a run state of "sp5u" is explicitly specified by the user
#
upmd5:sp5u:respawn::sp5d:sp5d::sp5d:-1::::60:7:105:home:./bin/upmd 5
scmd5:sp5u:respawn::sp5d:sp5d::sp5d:-1::::60:8:105:home:./bin/scmd 5

#
# terminate daemon processes associated with signalling point 5
# when a run state of "sp5d" is explicitly specified by the user
# and/or a failure is encountered during the start-up phase
#
scmd5:sp5d:off:::::::::::::home:./bin/scmd 5
upmd5:sp5d:off:::::::::::::home:./bin/upmd 5

#
```



```
# start daemon processes associated with signalling point 6
# when a run state of "sp6u" is explicitly specified by the user
#
upmd6:sp6u:respawn::sp6d:sp6d::sp6d:-1::::60:7:106:home:./bin/upmd 6
scmd6:sp6u:respawn::sp6d:sp6d::sp6d:-1::::60:8:106:home:./bin/scmd 6

#
# terminate daemon processes associated with signalling point 6
# when a run state of "sp6d" is explicitly specified by the user
# and/or a failure is encountered during the start-up phase
#
scmd6:sp6d:off::::::::::::home:./bin/scmd 6
upmd6:sp6d:off::::::::::::home:./bin/upmd 6

#
# start daemon processes associated with signalling point 7
# when a run state of "sp7u" is explicitly specified by the user
#
upmd7:sp7u:respawn::sp7d:sp7d::sp7d:-1::::60:6:107:home:./bin/upmd 7
scmd7:sp7u:respawn::sp7d:sp7d::sp7d:-1::::60:8:107:home:./bin/scmd 7

#
# terminate daemon processes associated with signalling point 7
# when a run state of "sp7d" is explicitly specified by the user
# and/or a failure is encountered during the start-up phase
#
scmd7:sp7d:off::::::::::::home:./bin/scmd 7
upmd7:sp7d:off::::::::::::home:./bin/upmd 7

# add application specific entries below this line
```

7.3.2 apmconfig.old

NAME

apmconfig.old *apmd* configuration file for previous versions

SYNOPSIS

$EBSHOME/access/RUN/config/PMGR/apmconfig.old

DESCRIPTION

**id:dirpath:execname:action:estate:hbeatind:sstate:fstate:hstate:astate:nstate:
retrycnt:retryint:retrydel:acktime:args**

Previous version of the configuration file which controls the operations of the *apmd* daemon process. It is composed of individual entries with the position-dependent syntax defined in the synopsis. The fields of the entry must be separated by a **:** character, with no white space in the line. Each entry is delimited by a *newline*. A maximum of 512 characters are permitted for each entry. Comments may be inserted as separate lines using the convention for comments described in *sh(1)*.

***Important**: Beginning with Distributed7, the apmconfig file should be used. The format in the **apmconfig.old** file has only been preserved to provide backward compatibility in this release. There is no guarantee that it will be supported in future releases of the product.*

When creating and/or modifying the file, the following guidelines should be observed:

- Each line must contain the correct number of fields and must be formatted correctly. If a line is incorrect, apmd will ignore the entries in the file. However, it will log the line number at which an error or inconsistency was encountered.
- The UNIX path names for the executables are valid and correct.
- The start-up and steady-state information provided for individual entries must not conflict with other entries, which could cause undesired loops in processing.
  The apmd process is as intelligent as the logic provided in the apmconfig file!
- Shell scripts which are invoked from an entry must have a statement in the first line that specifies which shell version to be invoked (e.g., #!/bin/sh to invoke plain UNIX shell).
- After making any changes to the contents of the file while the system is running, you must issue the *apm_update* command to notify the *apmd* daemon to re-read and execute the file.

There are no limits on the number of entries. The entry fields are defined as follows:

id An alphanumeric string used to uniquely identify an entry. The contents of this field are appended to the local host name to form the identity in the **id@host** format. The total size of the **id@host** string cannot exceed 24 characters.

dirpath The UNIX path name of the executable program. This field is used with the **execname** field to fully identify the executable. Either a keyword (home or run) or a full UNIX path name (starting with the / character) must be in this field. The keywords have the following meanings:

- **home**
  For AccessSERVICES version: ***$EBSHOME/access***
  For CRP version: ***$CRPDIR***
  For basic Distributed7 version: ***$EBSHOME/access***
- **run**
  For AccessSERVICES version: ***$EBSHOME/access/RUN***
  For CRP version: ***$APPHOME***
  For basic Distributed7 version: ***$EBSHOME/access/RUN***

execname The relative path name and file name of the executable program. This field is used with the **dirpath** field to locate the UNIX path name for the executable program and the executable name

action The action(s) that *apmd* should take on the process identified in the **process** field. Several key words exist that are recognized by *apmd*. Actions are only taken if *apmd*'s run state matches a state in the **estate** field. Valid actions are:

- **START**: If the process named in the **execname** field is not running, *apmd* should start it and not wait for the process's termination
- **KEEPALIVE**: If the process named in the **execname** field is not running, *apmd* should start it and proceed to the state specified in the appropriate state field. The *apmd* should not wait for the process's termination but it should restart the process if start-up fails. If the process terminates for any reason, the *apmd* should keep attempting to restart the process until it starts or until the number of attempts that occur in the last interval of **retryint** seconds exceeds **retrycnt**. A delay of **retrydel** seconds should occur between attempts.
  After initially starting the process, *apmd* will go to the state specified in one of the following *state* fields. The *state* field it will access depends on the contents of **acktime** and actions of the process. Based on this combination of results, each state field should hold an appropriate *apmd* run state.
  astate - 1) **acktime** field holds a non-zero value and a positive acknowledgment was received from the process
  2) no acknowledgment is expected (**acktime** is 0 or empty)
  nstate - 1) negative acknowledgment was received from process
  2) no acknowledgment was received within the time specified in **acktime**
  sstate - 1) process terminates with an exit code of 0, at any time
  fstate - 1) process terminates with a non-zero exit code (at any time)
  2) process is killed by a signal it could not handle
  hstate - 1) number of attempts to restart process in the last **retrydel** seconds has exceeded the value in **retrycnt**
  No further attempts to restart the process will be made.

- **WAIT**: If the process named in the **process** field is not running, ***apmd*** should start it and wait for it to terminate. After the process terminates, it can proceed to the state specified in the appropriate state field. (No other activities will occur until the process terminates.) The *state* field it will access depends on the contents of **acktime** and actions of the process. Based on this combination of results, each state field should hold an appropriate *apmd* run state.
  astate - 1) **acktime** field holds a non-zero value and a positive acknowledgment was received from the process
  2) no acknowledgment is expected (**acktime** is 0 or empty)
  nstate - 1) negative acknowledgment was received from process
  2) no acknowledgment was received within the time specified in **acktime**
  sstate - 1) process terminates with an exit code of 0, at any time
  fstate - 1) process terminates with a non-zero exit code, at any time
  2) process is killed by a signal it could not handle
- **OFF**: If the process named in the **process** field is currently running, generate a SIGTERM signal to terminate it. If the process does not terminate within the next 3 seconds, send a SIGKILL signal to terminate it. Then, switch to state specified in the **sstate** field. If the process is not running, ignore this entry.
- **CHNGSTATE**: Change the current run state of *apmd* to the state specified in the **sstate** field.

estate The state(s) that the *apmd* must be in for this entry to be executed. If the *apmd*'s run state while executing this file is among the states specified in this field, the entry is executed. Otherwise, it is ignored. A maximum of 16 execution states may be specified for an individual entry. Multiple execution states must be separated from each other with a comman (**,**); no white space should exist. If no states are defined for **estate**, then the entry is executed every time *apmd* executes the file. Execution states are defined by the developer and may have names that are up to four alphanumeric characters long.

hbeatind The heartbeat indicator. A value in this field enables the heartbeat feature, which causes *apmd* to send heartbeat request messages to the process every 5 seconds. If the **()** characters are in this field, the heartbeat feature is disabled for the process.

The heartbeat feature sends periodic heartbeat messages to the process if it is enabled. If the process fails to respond to 3 consecutive heartbeat request messages, apmd terminates the process by sending it a SIGKILL signal. Then, when the process terminates, the *apmd* will go to the state specified in the **fstate** field.

In the AccessCRP version, this field can also be used to specify the subsystem ID associated with the process by entering the non-negative integer subsystem ID.

sstate	The success state that *apmd* will be set to when the process terminates with a zero exit code. The state is defined by the developer and may have a name that is up to 4 alphanumeric characters long. If the field holds the \\ characters, then the state is set to *don't care*.
fstate	The failure state that *apmd* will be set to when the process terminates with a non-zero exit code or the process terminates because of a signal it could not handle. The state is defined by the developer and may have a name that is up to 4 alphanumeric characters long. If the field holds the \\ characters, then the state is set to *don't care*.
hstate	The hopeless state that *apmd* will be set to when **retrycnt** successive attempts to restart the process fail within the **retryint** interval. The state is defined by the developer and may have a name that is up to 4 alphanumeric characters long. If the field holds the \\ characters, then the state is set to *don't care*.
astate	The positive acknowledgment state that *apmd* will be set to when a positive acknowledgment is received from the process during the acknowledgment interval or when no acknowledgment message is expected. The state is defined by the developer and may have a name that is up to 4 alphanumeric characters long. If the field holds the \\ characters, then the state is set to *don't care*.
nstate	The negative acknowledgment state that *apmd* will be set to when a negative acknowledgment is received from the process or when the acknowledgment interval expires without an acknowledgment being received. The state is defined by the developer and may have a name that is up to 4 alphanumeric characters long. If the field holds the \\ characters, then the state is set to *don't care*.
retrycnt	The number of times *apmd* should try to restart the process within **retryint** seconds. Restarts are attempted only if the **action** field is *KEEPALIVE* and the process has terminated with a non-zero exit code or was killed by an unexpected signal. If this field is left blank, the default value of 3 is used.
retryint	The time interval, in seconds, during which *apmd* should try to restart the process, up to **retrycnt** number of times. Restarts are attempted only if the **action** field is *KEEPALIVE* and the process has terminated with a non-zero exit code or was killed by an unexpected signal. If this field is left blank, the default value of 60 seconds is used.
retrydel	The time delay, in seconds, that should occur between attempts to restart the process. Restarts are attempted only if the **action** field is *KEEPALIVE* and the process has terminated with a non-zero exit code or was killed by an unexpected signal. If this field is left blank, the default value of 1 second is used.
acktime	The time interval, in seconds, during which *apmd* should wait for a positive or negative acknowledgment from the process before moving to

a run state. A value of 0 or an empty field means that *apmd* will automatically move to the run state specified in the **astate** field and will not wait for an acknowledgment from the process. A value of -1 means that *apmd* must wait indefinitely for an acknowledgment.

args The command-line arguments, if any for the process. A maximum of 32 arguments can be specified.

***Important**: Input/output redirection is not currently supported.*

SEE ALSO apmd, apm_update, apm_getstate, apm_setstate, apmconfig

This page is intentionally blank.

Chapter 8: User Commands

8.1 Chapter Overview

This chapter provides descriptions of the Distributed7 platform Service Provider Module (SPM), Distributed Shared Memory (DSM), Distributed Kernel Memory (DKM), Transaction Capabilities Application Part (TCAP), Application Process Manager (APM) and Virtual Board (VB) user commands. They are summarized in the following table.

Table 8-1: User Command Summary

Command	Description
ebs_alarm	Trigger alarm condition.
ebs_apidemo	Demonstrates SPM API capabilities introduced with Distributed7.
ebs_brdfinfo	Get and clear SS7 board crash dump.
ebs_cfgbrd	Configure and unconfigure SS7 board.
ebs_dbconfig	Save or restore configuration data
ebs_dnlbrd	Reset and download SS7 board.
ebs_mngbrd	Start and stop SS7 board sanity checks.
ebs_oldapidemo	Demonstrates SPM API capabilities of releases prior to Distributed7.
ebs_audit	Audits dynamic data of a host.
ebs_config	Configure system operation mode.
ebs_explain	Retrieves detailed ***errno*** information.
ebs_hbeat	Sets up heartbeat mechanism with a remote host.
ebs_ipcbm	Benchmarks Distributed7 IPC system performance.
ebs_log	Activates and deactivates message logging capabilities for processes.
ebs_loopback	Activates and deactivates message loopback from a process to a fixed destination.
ebs_modinstall	Installs STREAMS drivers and drivers for boards.
ebs_modremove	Removes drivers.
ebs_modunload	Unload Distributed7 modules.
ebs_mtpglobal	Change the global instance of ***upmd*** in a distributed environment.
ebs_pkgrm	Removes Distributed7 packages.
ebs_ps	Displays information on registered processes.
ebs_qinfo	Retrieves STREAMS queue information (settings, sizes).
ebs_qlist	Displays queue addresses.
ebs_qstat	Retrieves STREAMS queue statistics.
ebs_report	Displays alarm report.

Table 8-1: User Command Summary (Continued)

Command	Description
ebs_setrelease	Activate specified Distributed7 release.
ebs_showlink	Displays link status for SS7 hardware and network interface.
ebs_shutdown	Shuts down applications and the Distributed7 software on remote hosts.
ebs_start	Starts the Distributed7 system software.
ebs_stop	Shuts down all applications and the Distributed7 software on the local host.
ebs_sync	Synchronizes dynamic data of hosts in a distributed environment.
ebs_sysinfo	Displays configuration information of the local host.
ebs_tasklist	Retrieves task list information.
ebs_tune	Tunes operating system parameters.
getcfg	Gets information about SS7 controllers in the system.
apm_audit	Audit *apmd* IPC resources.
apm_getstate	Retrieves current *apmd* run state.
apm_kill	Sends a signal to a process.
apm_killall	Sends a signal for all processes to terminate.
apm_ps	Reports process status.
apm_report	Generates a log report.
apm_setstate	Manipulates *apmd* run state.
apm_start	Starts the *apmd* daemon.
apm_stop	Terminates the *apmd* daemon.
apm_trcapture	Captures information used for tracing execution of application programs on the local host.
apm_trclear	Clears the contents of the trace shared memory.
apm_trgetmask	Retrieves trace mask settings.
apm_trinit	Initializes IPC shared memory.
apm_trsetmask	Sets a trace mask.
apm_trshow	Displays the trace output.
apm_update	Informs *apmd* of any changes in the configuration.
dsm_apidemo	Demonstrates the capabilities of the Distributed Shared Memory library functions.
dsm_audit	Audits Distributed Shared Memory dynamic data.
dsm_bm	Benchmarks Distributed7 DSM framework.
dsm_list	Displays Distributed Shared Memory information.
dsm_rm	Removes a DSM segment.
dsm_stat	Retrieves information about a DSM segment.
dkm_apidemo	Demonstrates the capabilities of the Distributed Kernel Memory library functions.
dkm_bm	Benchmarks Distributed7 DKM framework.
dkm_dump	Retrieves DKM segment contents.
dkm_list	Displays DKM related information.
dkm_rm	Destroys DKM segment and/or segment extension.
dkm_sar	DKM system activity reporter.
dkm_stat	Retrieves DKM block status information.

Table 8-1: User Command Summary (Continued)

Command	Description
dratest	Demonstrates Distributed Record Access capabilities.
rtc_dump	Retrieves RTCMOD information.
rtc_stat	Enables/disables RTCMOD measurements collection
tcm_apidemo	Demonstrates the capabilities of the TCAP library functions.
tcm_list	Display TCAP subsystem information.
tcm_stat	Enables and disables TCAP statistics collection
tcm_tune	Tune TCAP optional parameters
i_trace	Activates (or deactivates) the ISUP message trace.
vb_addhost	Used to add a host to an established virtual board environment.
vb_bridge	Establish a bridge for message transmission between two hosts.
vb_config	The user interface for the virtual board driver.
vb_connhosts	-ksh script that establishes connections between each pair of hosts.
vb_connports	Defines a link between two ports.
vb_discport	Breaks a link connection.
vb_lhosts	Lists host connections information for local host.
vb_lports	Lists host port information on local host.
vb_reset	Resets port and host connections on all hosts in the virtual board environment.
vb_startup	Virtual board environment configuration file.
snmptest	Communicates with a network entity using SNMP Requests
snmptrapd	Receives and prints SNMP traps.
snmpwalk	Communicates with a network entity using SNMP GET Next Requests
snmpget	Communicates with a network entity using SNMP GET Requests
db2date	Converts old database files to new database files
db2text	Converts all previous release ALARM, MML, NETWORK, SPM, MTP, SCCP, and ISUP configuration database files to text files containing the MML commands that created the configuration.

Important*: Please refer to Chapter 2 for a list of the user commands with external dependencies to make sure your environment has the necessary software libraries.*

To use the Distributed7 user commands, set the $**EBSHOME** environment variable and include ***$EBSHOME/access/bin*** in the command path. The $**EBSHOME** environment variable should be set to the path where the Distributed7 software is installed.

To set the variable, use a C-shell command similar to this sample:

```
setenv EBSHOME /<samedir>/<mydir>/<mySS7>
```

Important*: $EBSHOME can be up to 1024 total characters.*

To add the Distributed7 ***bin*** directory into the command path, use the following command:

```
setenv PATH ${PATH}:$EBSHOME/access/bin
```

On-line reference manuals on all utility programs and system processes are also available in the Distributed7 system. These reference manuals are provided in the form of manual pages so that the user can invoke the UNIX standard ***man(1)*** utility to review the information contained in them. The Distributed7 manual pages are provided within the ***$EBSHOME/access/manpages*** directory. Therefore, the user should set the **MANPATH** environment variable as follows:

```
setenv manpath ${manpath}:$EBSHOME/access/manpages
```


8.2 Platform Utilities

8.2.1 ebs_alarm

NAME

ebs_alarm Triggers a user-specified alarm condition.

SYNOPSIS

ebs_alarm ***[-i*** id ***] [-p*** pri ***] [-o*** opt_param(s) ***] [-d*** int_param(s) ***] [-s*** str_param(s) ***]*** ***[*** fmt ***]***

DESCRIPTION

ebs_alarm This utility triggers a user-specified alarm condition by issuing an appropriate request to the ***alarmd*** daemon on the local host. ***ebs_alarm*** is used in verifying the correct operations of the Distributed7 alarm subsystem at any time (e.g., when introducing a new group of user-specified alarms to the system and/or following a change to the contents of existing alarm groups and alarm text files associated).

Without any options, ***ebs_alarm*** will call the ***alm_report*** function using the following arguments:

alm_report(fd,
sp = L_SP_NA,
grp = L_ALMGRP_LOG, mod = 0, num = 0,
pri = L_ALMLVL_INFO,
param1 = L_NO_PARAM, param2 = L_NO_PARAM,
paramrest = NULL, char *fmt = NULL);

which will result in an informational ***alarmd*** condition with an alarm ID value of L_SYSDEF_ALARM_TYPE_BASE [$800000] to be triggered, with no additional parameters and/or alarm text.

NOTE: *It is possible, using the appropriate commandline options, to change any of these default settings (with the exception of the sp = L_SP_NA argument) to trigger specific alarm conditions at specific priority levels and/or to supply a list of parameters and user-specified format strings.*

-i id Use 6-digit ***id*** value specified in hexadecimal format. Alarm ID values are constructed on the basis of alarm group, module, and number information as follows:

id = (grp << 16) | (mod << 8) | num

-p pri Triggers an alarm condition at specified priority/severity level. The ***pri*** argument assumes a value from the following list:

1. Informational

2. Minor severity level

3. Major severity level

4. Critical severity level

5. Fatal severity level

By default, severity level will be set to *informational*.

-o opt_param Populate optional ***param1*** and ***param2*** parameters per values supplied via this argument.

Note*: To populate the* ***param2*** *parameter,* ***-o*** *option needs to be specified twice.*

-d int_param Populate arbitrary ***paramrest*** parameters per values supplied via the integer-type ***int_param*** argument.

Note*: To populate multiple integer-type* ***paramrest*** *parameters,* ***-d*** *option needs to be specified multiple times.*

-s str_param Populate arbitrary ***paramrest*** parameters per values supplied via the string-type ***str_param*** argument.

Note*: To populate multiple string-type* ***paramrest*** *parameters,* ***-s*** *option needs to be specified multiple times.*

EXAMPLES

To trigger a "critical" alarm message:

ebs_alarm -p 4 'system panic'

To trigger a "major" alarm for port 8 on board 2.

ebs_alarm -p 3 -o 2 -o 8 'board %d port %d failure'

To trigger an alarm using varying integer/string-type parameters and user-defined text format.

ebs_alarm -s "abc" -s "xyz" -d 123 'm:%s v:%s b:%d'

FILES

`$EBSHOME/access/RUN/config/ALARM/alarmGroups`

`$EBSHOME/access/RUN/config/ALARM/*_almTxt`

SEE ALSO alarmd**,** alm_report()**,** spm_alarm()

8.2.2 ebs_apidemo

NAME

ebs_apidemo Demonstrates new SPM library capabilities

SYNOPSIS

ebs_apidemo

DESCRIPTION

ebs_apidemo Starts a menu-driven program which demonstrates and exercises the basic set of capabilities provided with the Distributed7 Applications Programming Interface (API) SPM library (***libspm***).

The program allows the user to:

- establish and remove multiple service endpoints
- bind and unbind addresses at individual service endpoints
- perform local or network-based address binding
- perform exclusive or non-exclusive address binding
- select between active and standby mode of operation
- manipulate the current operation mode or service type
- create multiple instances of an object
- load-share among multiple instances of an object
- broadcast messages to selected instances of an object
- exchange messages in normal or expedited mode
- forward received messages to a new destination
- send messages in deferred mode
- retrieve messages in a selective manner
- activate and deactivate message logging at a service endpoint
- activate and deactivate message loopback at a service endpoint
- retrieve information about processes which are currently executing
- retrieve or manipulate water marks at the streamhead
- retrieve or manipulate queue management parameters
- generate alarm messages
- specify event handlers to do extended event management
- trigger user-specified events

SEE ALSO ebs_oldapidemo

8.2.3 ebs_brdfinfo

NAME

ebs_brdfinfo Get and clear SS7 board crash dump.

SYNOPSIS

ebs_brdfinfo [-glc devname]

DESCRIPTION

ebs_brdfinfo This utility gets and clears the crash dump of a user-specified SS7 signalling hardware—the SS7 board—on the local host machine. After an SS7 board crash is detected by the board sanity check mechanism, the SS7 board crash dump is copied from board-shared memory to a buffer area on the host. The ***ebs_brdfinfo*** utility allows the user to view and clear the latest board crash dump.

The ***ebs_brdfinfo*** utility is part of the set of programs called Distributed7 Configuration Utilities, which includes ***ebs_dnlbrd***, ***ebs_cfgbrd***, and ***ebs_mngbrd***. This set of utilities allows the user to configure an SS7 board without using the MML interface.

devname This argument specifies the board device driver and instance number of the SS7 board for which the user wants to view and clear the board crash dump. The ***devname*** argument can be a value from the following list:

- sbs334 -- The sbs334 device driver that supports SBS334, SBS370, and SBS372 boards.
- pci334 -- The pci334 device driver that supports PCI334, PCI370, and PCI372 boards.
- pci3xpq -- The pci3xpq device driver that supports PCI370PQ and PCI372PQ boards.
- cpc3xpq -- The cpc3xpq device driver that supports CPC370PQ and CPC372PQ boards.
- pmc8260 -- The pmc8260 device driver that supports the PMC8260 board.
- artic8260 -- The artic8260 driver that supports the ARTIC1000 and ARTIC2000 boards.
- vbrd -- The Distributed7 Virtual BoaRD (VBRD) device driver.

Use the ***getcfg*** command for a list of available SS7 boards and corresponding instance numbers.

-g Get (view) the crash dump of the specified SS7 board.

-c Clear the crash dump of the specified SS7 board, both on the host buffer and on the SS7 board shared memory.

EXAMPLES

To get (view) the crash dump of an sbs334 with instance 0 on local host Host-A:

host-A% ebs_brdfinfo -g sbs3340

To clear the crash dump of an sbs334 with instance 0 on local host Host-B:

host-A% ebs_brdfinfo -c sbs3340

SAMPLE OUTPUT

A sample crash dump output for an sbs334 with instance 0:

```
sbs334[0] crash log begin...
 crash log. . .
 pc = 00425D38
 sw = 2004 (trap)
 sw = 2004 (handler)
 fault vector = 0002 (2)
 fault format = c000
 a0:00FFF000 a1:00002700 a2:007206B2 a3:00720679
 a4:0072067A a5:0072067B a6:00720680 a7:00720634
 d0:00000064 d1:00000001 d2:00000004 d3:0000000F
 d4:00000005 d5:00000064 d6:FFFFFFFF d7:000002A0
code at fault address. . .
 00425D38  2F28 000C 206E FFFC 2F28 0008 206E FFFC
 00425D48  2F28 0004 206E FFFC 2F10 2F2E FFFC 487B
fault stack frame. . .
 00720634  2004 0042 5D38 C008 00FF F00C FFFF F000
 00720644  0042 5D38 0008 0045 0000 0004 0000 0001
 00720654  0000 0005 0000 0080 0072 06B2 0072 06F8
 00720664  0042 89F0 0072 06F0 0000 0001 0000 0000

sbs334[0] crash log end...
```

SEE ALSO spmd, apm_start, ebs_dnlbrd, ebs_cfgbrd, ebs_mngbrd, **streamio**

8.2.4 ebs_cfgbrd

NAME

ebs_cfgbrd Configure and unconfigure SS7 board.

SYNOPSIS

ebs_cfgbrd [-c|u|r [hostname:]devname]

DESCRIPTION

ebs_cfgbrd This utility configures and removes configuration from a user-specified SS7 signalling hardware—the SS7 board—on the local or remote host machine, following the start-up of the Distributed7 system software using the ***apm_start*** utility. The user can exchange SS7 messages through corresponding SS7 board devices only when the connection between the SPM and the board device driver for an SS7 board is in place, and when the SS7 board is configured. Configuring/removing configuration on the SS7 board with ***ebs_cfgbrd*** is done conceptually the same way as with the MODIFY-SS7BOARD command of Man Machine Language (MML) interface.

The ***ebs_cfgbrd*** utility is part of the set of programs called Distributed7 Configuration Utilities, which includes ***ebs_dnlbrd***, ***ebs_mngbrd***, and ***ebs_brdfinfo***. This set of utilities allows the user to configure an SS7 board without using the MML interface.

hostname This argument specifies the name of the host machine at which the SS7 board is physically located. It is an optional argument. When no host name is entered, the local host name is assumed.

devname This argument specifies the board device driver and instance number of the SS7 board for which the user is interested in viewing and clearing the board crash dump. The ***devname*** argument can be a value from the following list:

- sbs334 -- The sbs334 device driver which supports SBS334, SBS370, and SBS372 boards.
- pci334 -- The pci334 device driver which supports PCI334, PCI370, and PCI372 boards.
- pci3xpq -- The pci3xpq device driver which supports PCI370PQ and PCI372PQ boards.
- cpc3xpq -- The cpc3xpq device driver that supports CPC370PQ and CPC372PQ boards.
- pmc8260 -- The pmc8260 device driver that supports the PMC8260 board.
- artic8260 -- The artic8260 driver that supports the ARTIC1000 and ARTIC2000 boards.

- vbrd -- The Distributed7 Virtual BoaRD (VBRD) device driver.

Use the ***getcfg*** command for a list of available SS7 boards and corresponding instance numbers.

-c Configure the specified SS7 board, has the same effect with MODIFY-SS7BOARD: CONF=ON; command of MML.

-u Unconfigure the specified SS7 board, has the same effect with MODIFY-SS7BOARD: CONF=OFF; command of MML.

-r Recover the specified SS7 board which is in FAILED state, has the same effect with MODIFY-SS7BOARD: CONF=ON; command of MML when the SS7 board is in FAILED state. This option will fail if the board state is not FAILED.

EXAMPLES

To configure an sbs334 with instance 0 on local host Host-A:

```
host-A% ebs_cfgbrd -c host-A:sbs3340
```

or,

```
host-A% ebs_cfgbrd -c sbs3340
```

To configure an sbs334 with instance 0 on remote host Host-B:

```
host-A% ebs_cfgbrd -c host-B:sbs3340
```

To unconfigure an sbs334 with instance 0 on local host Host-A:

```
host-A% ebs_cfgbrd -u host-A:sbs3340
```

or,

```
host-A% ebs_cfgbrd -u sbs3340
```

To unconfigure an sbs334 with instance 0 on remote host Host-B:

```
host-A% ebs_cfgbrd -u host-B:sbs3340
```

To recover an sbs334 with instance 0 on local host Host-A:

```
host-A% ebs_cfgbrd -r host-A:sbs3340
```

or,

```
host-A% ebs_cfgbrd -r sbs3340
```

To recover an sbs334 with instance 0 on remote host Host-B:

```
host-A% ebs_cfgbrd -r host-B:sbs3340
```

SEE ALSO spmd, apm_start, ebs_dnlbrd, ebs_brdfinfo, ebs_mngbrd, streamio

8.2.5 ebs_dbconfig

NAME

ebs_dbconfig Save or restore configuration data.

SYNOPSIS

ebs_dbconfig -s*ave -re*store [-d*ir backup-dir] [-f*orce] [-ru*n run-list] [-p*attern pattern-list]

DESCRIPTION

ebs_dbconfig This utility is intended to save or restore Distributed7 configuration data. It can be instructed to use a particular backup directory, select a subset of the configuration directories and backup (or restore) only files matching the specified pattern(s). All options other than **save** and **restore** have default values. One and only one of **save** or **restore** options must be specified for a particular invocation of ***ebs_dbconfig***.

The requesting user must have read and read-write privileges to specified source and destination directories, respectively.

The ***ebs_dbconfig*** utility also checks the usage status of the files to be backed-up (restored). Unless the **force** (-force) option is specified, the backup (restore) action is rejected if files currently in use are being backed-up (restored).

For all options, * sign indicates the end of the mandatory option prefix.

OPTIONS

-s*ave Save existing Distributed7 configuration to the specified backup directory. This option cannot be specified together with the restore option.

-re*store Restore Distributed7 configuration from the specified backup directory. This option cannot be specified together with the save option.

-d*ir Specifies the backup directory. Defaults to ***$EBSHOME/access/BACKUP***.

-ru*n Used to qualify the configuration (RUN) directories to be backed-up (restored). Empty directories are skipped even if specified. Defaults to "RUN RUN0 RUN1 RUN2 RUN3 RUN4 RUN5 RUN6 RUN7"

-p*attern Specifies a list of shell style glob-patterns for selecting files to be backed-up or restored. Defaults to "*".

-f*orce Used to skip file usage check during the backup (restore) operation. When this option is used, backed-up files might contain inconsistent information.

EXAMPLES

The following example illustrates how ***ebs_dbconfig*** can be used to save the complete configuration to the default backup directory:

ebs_dbconfig -s

To restore the core configuration directory as well as SP1 and SP2 configuration directories from backup directory ***/home/config/D7***:

ebs_dbconfig -re -d /home/config/D7 -run "RUN RUN1 RUN2"

Finally, to save all SCCP and MTP configuration data to the default backup directory:
ebs_dbconfig -save -pattern "mtp* sccp*"

FILES

$EBSHOME/access/RUN/config/RUN/DBfiles

$EBSHOME/access/RUN/config/RUN[0-7]/DBfiles

SEE ALSO ebs_setrelease, netd, spmd, isupd, upmd, scmd, **mml**

8.2.6 ebs_dnlbrd

NAME

ebs_dnlbrd Reset and download SS7 board.

SYNOPSIS

ebs_dnlbrd [devname]

DESCRIPTION

ebs_dnlbrd This utility resets and downloads SAL/MTPL2 binaries to a user-specified SS7 signalling hardware—SS7 board—on the local host machine.

The ***ebs_cfgbrd*** utility is part of the set of programs called Distributed7 Configuration Utilities, which includes ***ebs_cfgbrd***, ***ebs_mngbrd***, and ***ebs_brdfinfo***. This set of utilities allows the user to configure an SS7 board without using the MML interface.

devname This argument specifies the board device driver and instance number of the SS7 board for which the user is interested in viewing and clearing the board crash dump. The ***devname*** argument can be a value from the following list:

- sbs334 -- The sbs334 device driver which supports SBS334, SBS370, and SBS372 boards.
- pci334 -- The pci334 device driver which supports PCI334, PCI370, and PCI372 boards.
- pci3xpq -- The pci3xpq device driver which supports PCI370PQ and PCI372PQ boards.
- cpc3xpq -- The cpc3xpq device driver that supports CPC370PQ and CPC372PQ boards.
- pmc8260 -- The pmc8260 device driver that supports the PMC8260 board.
- artic8260 -- The artic8260 driver that supports the ARTIC1000 and ARTIC2000 boards.
- vbrd -- The Distributed7 Virtual BoaRD (VBRD) device driver.

Use the ***getcfg*** command for a list of available SS7 boards with corresponding instance numbers.

EXAMPLES

To download an sbs334 with instance 0 on local host-A:

```
host-A% ebs_dnlbrd sbs3340
```

SEE ALSO spmd, apm_start, ebs_cfgbrd, ebs_brdfinfo, ebs_mngbrd, **streamio**

8.2.7 ebs_mngbrd

NAME

ebs_mngbrd Start and stop SS7 board sanity checks.

SYNOPSIS

ebs_mnglbrd [-o|f devname]

DESCRIPTION

ebs_mngbrd This utility starts and stops the sanity check on a user-specified SS7 signalling hardware—SS7 board—on the local host machine. SS7 board sanity check periodically tests the board state to detect software/ hardware problems. Starting the sanity check is normally done during SS7 board configuration. The ***ebs_mngbrd*** utility allows the user to take the SS7 board off line by stopping the sanity check to simulate a board crash.

The ***ebs_mngbrd*** utility is part of the set of programs called Distributed7 Configuration Utilities, which includes ***ebs_dnlbrd***, ***ebs_cfgbrd***, and ***ebs_brdfinfo***. This set of utilities allows the user to configure an SS7 board without using the MML interface.

devname This argument specifies the board device driver and instance number of the SS7 board for which the user is interested in viewing and clearing the board crash dump. The ***devname*** argument can be a value from the following list:

- sbs334 -- The sbs334 device driver which supports SBS334, SBS370, and SBS372 boards.
- pci334 -- The pci334 device driver which supports PCI334, PCI370, and PCI372 boards.
- pci3xpq -- The pci3xpq device driver which supports PCI370PQ and PCI372PQ boards.
- cpc3xpq -- The cpc3xpq device driver that supports CPC370PQ and CPC372PQ boards.
- pmc8260 -- The pmc8260 device driver that supports the PMC8260 board.
- artic8260 -- The artic8260 driver that supports the ARTIC1000 and ARTIC2000 boards.
- vbrd -- The Distributed7 Virtual BoaRD (VBRD) device driver.

Use the ***getcfg*** command for a list of available SS7 boards with corresponding instance numbers.

-o Start the sanity check on the specified SS7 board, in other words, take the SS7 board on-line.

-f	Stop the sanity check on the specified SS7 board, in other words, take the SS7 board off-line.

EXAMPLES

To start sanity on an sbs334 with instance 0 on local host Host-A:

host-A% ebs_mngbrd -o sbs3340

To stop sanity on an sbs334 with instance 0 on local host Host-B:

host-A% ebs_mngbrd -f sbs3340

SEE ALSO spmd, apm_start, ebs_cfgbrd, ebs_brdfinfo, ebs_dnlbrd**, streamio**

8.2.8 ebs_oldapidemo

NAME

ebs_oldapidemo Demonstrates SPM library capabilities.

SYNOPSIS

ebs_oldapidemo

DESCRIPTION

ebs_oldapidemo Starts a menu-driven program which demonstrates the basic set of Distributed7 SPM library API capabilities in the releases prior to 3.5.x. It also demonstrates the backward compatibility between Release 3.5.x and earlier releases.

Using this program, the user can:

- register and deregister objects with the environment
- send and receive messages
- send messages in deferred mode
- activate and deactivate message logging
- activate and deactivate message loopback
- retrieve information about other objects
- generate alarm messages

SEE ALSO ebs_apidemo

8.2.9 ebs_audit

NAME

ebs_audit Audits dynamic data.

SYNOPSIS

ebs_audit [hostname]

DESCRIPTION

ebs_audit Issues a manual request to audit the dynamic data tables maintained on a host machine which is operating under the Distributed7 environment. The host machine can be the local host or a remote host.

hostname Identifies which host to run audits on. If a host name is not specified, dynamic data on the local host machine will be audited.

The audit is an internal review and possible correction of all appropriate dynamic data on the specified host machine. Examples of dynamic data are the database tables for the objects that are executing under the Distributed7 environment and for the SS7 signalling link hardware that is available on the individual host machines.

Important*: Since the Distributed7 environment has an automatic mechanism to periodically audit and correct the dynamic data tables stored on the individual host machines, execution of this command is not normally required. This command simply provides a means to manually audit the dynamic data if the automatic auditing mechanism fails to operate properly.*

SEE ALSO ebs_ps, ebs_showlink, ebs_sync

8.2.10 ebs_config

NAME

ebs_config Configure system operation mode.

SYNOPSIS

ebs_config

DESCRIPTION

ebs_config Use ***ebs_config*** to configure the operation mode of Distributed7 system software on the local host as stand-alone or distributed. This script is only for modifying the operation mode that was specified during initial system software installation, i.e., from stand-alone to distributed, or vice versa.

ebs_config accesses information stored in the ***/etc/amgrmode*** file—created at Distributed7 installation time—to determine the current mode of operation on the local machine. It then replaces selected components of the base Distributed7 system software, such as executables and configuration files, with their appropriate versions. Finally, it updates the ***/etc/amgrmode*** file to reflect the new mode of operation, and removes all network related managed object database files.

The ***ebs_config*** script can also modify the default hostname setting that Distributed7 software uses. By default, the software uses the UNIX nodename set on the local host with the **`uname -n`** command as the official hostname. There are times, however, when the user may want to give the software a hostname other than the official UNIX nodename. For example, in a product configuration where a particular host machine is part of multiple networks, such as public and private networks, the user can reserve the official hostname of the machine for one network, and have Distributed7 software run on another network under a different hostname.

To determine the default hostname, the system software accesses existing hostname information stored in the ***/etc/amgrhost*** file. The user can modify the contents of this file with the ***ebs_config*** script. Note, however, that use of the ***/etc/amgrhost*** file is optional; this file does not get created during initial system software installation. In the absence of this file, the system software default is to the official UNIX hostname. Users interested in operating the software under a hostname that is different from the official hostname must run the ***ebs_config*** script to reset the hostname on that machine after the initial system software installation.

-u Upgrade option for switching LAN configuration from single to dual, or vice versa. Use this option only if you have already configured your system, i.e., specified operation mode and defined remote hosts, if any. Note that when this option is specified, all managed object (MO) database files associated with the NTWK, HOST, and TCPCON managed object are removed from the local host, and need to be re-entered.

*Note: You must have "root" privileges to execute the **ebs_config** script.*

WARNING: This script should be executed only when Distributed7 system software on the local host is NOT running. Parameters initialized/set by this script are used by the Distributed7 system software during system start-up time and thereafter.

ENVIRONMENT

The EBSHOME environment variable must be set before invoking this script.

FILES

```
/etc/amgrhost
/etc/amgrmode
$EBSHOME/access/drv/dramod
$EBSHOME/access/drv/.dramod.sa
$EBSHOME/access/drv/.dramod.dist
$EBSHOME/access/RUN/config/PMGR/apmconfig
$EBSHOME/access/RUN/config/PMGR/.apmconfig.sa
$EBSHOME/access/RUN/config/PMGR/.apmconfig.dist
$EBSHOME/access/RUN/DBfiles/net_tcpcon.DB
$EBSHOME/access/RUN/DBfiles/net_host.DB
$EBSHOME/access/RUN/DBfiles/net_ntwk.DB
```

SEE ALSO add_drv, ebs_modunload, ebs_modinstall, ebs_modunload, ebs_modremove, db2text, db2date

8.2.11 ebs_explain

NAME

ebs_explain Retrieves detailed ***errno*** information.

SYNOPSIS

ebs_explain errno

DESCRIPTION

ebs_explain Retrieves or displays information about a user-specified ***errno*** value that is encountered by a user-space application program when running under the Distributed7 environment. Among the information retrieved and displayed, are the general error category and a brief description of the error condition.

SEE ALSO spm_strerror(), spm_errgroup()

8.2.12 ebs_hbeat

NAME

ebs_hbeat Activates or deactivates the heartbeat mechanism to remote hosts

SYNOPSIS

ebs_hbeat [-x] [-a action ***] [-t*** time***]*** hostname

DESCRIPTION

ebs_hbeat Invokes the utility that activates or deactivates the heartbeat mechanism between the local host machine and a remote host machine which are both operating under the Distributed7 environment. This mechanism regularly monitors the accessibility and health of a remote host machine over the established TCP/IP link and takes the specified course of action if an abnormal situation develops.

The heartbeat mechanism is a kernel-level capability that can be controlled from the user-level. When the heartbeat mechanism on a host is enabled, the SPM on that machine will periodically generate *heartbeat request* messages and send them over a TCP/IP link to its peer on the remote host. The SPM on the remote host is expected to respond to each *heartbeat request* with a *heartbeat response* message. If the SPM on the local host does not receive a *heartbeat response* message, it marks the corresponding TCP/IP link as *heartbeat failed* and takes the action specified with the ***-a*** option. The system makes periodic attempts to restore links from a *heartbeat failed* state to *normal* state.

-x Indicates that the ***ebs_hbeat*** utility should not bind an address to the service endpoint associated with it (optional). Unless this option is specified, a named object entry for ***ebs_hbeat*** will be created in the process table of the local machine.

-a *action* Specifies the action to be taken when a *heartbeat response* message is not received (optional). Valid values for *action* are:

- **0**: No action should be taken.
- **1**: Remove the remote host's entries, i.e., for processes or SS7 link hardware, from the local machine's dynamic data table. The dynamic data tables of both hosts will be automatically synchronized when the heartbeat is restored. This value is the default, if a value is not specified.

-t *time* Specifies the length of the heartbeat interval in milliseconds. Omitting the argument or a value of zero (0) deactivates the heartbeat mechanism over the appropriate link.

hostname Identifies the remote host to activate or deactivate the heartbeat.

SEE ALSO netd, ebs_showlink, ebs_sync

8.2.13 ebs_ipcbm

NAME

ebs_ipcbm Benchmarks Distributed7 IPC system performance

SYNOPSIS

ebs_ipcbm -n -h hostname

DESCRIPTION

ebs_ipcbm Benchmarks the performance of the Distributed7 Inter-Process Communication (IPC) messaging mechanism when used for inter-process communication between application processes executing on the local host and application processes executing on different hosts. In either case, the application processes are assumed to be registered with the Distributed7 environment at the Service Provider Module (SPM) Multiplexer. The *apmd* and the *netd* processes on all involved hosts must be running.

When executed, ***ebs_ipcbm*** spawns an IPC message receiver process on an appropriate host (e.g., local host or the host identified via the hostname argument) and prompt the user for the specifics of the benchmark test to be performed (e.g., addressing method to be used during messaging, message size and priority). Note that the message receiver process spawned by the ***ebs_ipcbm*** has been pre-programmed to respond to the messages sent to it in a time-stamped manner. The ***ebs_ipcbm*** utility will exchange a total of 10,000 messages with the message receiver process and measure the round-trip delays involved in sending/receiving the individual messages.

After the specified number of IPC messages are exchanged, ***ebs_ipcbm*** will calculate the average round-trip delay for a single IPC message exchange and calculate the overall system performance in terms of the number of IPC messages [of specified size] per second. The results will be displayed to the user on *stdout*.

-n Skips the *spm_snd()* function call's sanity checks on destination address, resulting in a faster message exchange between the message sender and receiver processes.

-h hostname Indicates that the message receiver process should be executing on the host specified. By default, the message receiver process executes on the same host (local host) as the message sender process.

8.2.14 ebs_log

NAME

ebs_log Activates and deactivates message logging and lists logged processes.

SYNOPSIS

ebs_log [-l] [-do]

DESCRIPTION

ebs_log Invokes the utility which controls the Distributed7 message logging capabilities. The utility prompts for the information needed to activate or deactivate logging for a particular service endpoint, a user process or the link between any two adjacent STREAMS multiplexers. This command can also be used to display a list of all service endpoints that have message logging currently active.

When the message logging capability is activated, a copy of each message received or sent through the service endpoint is forwarded to either the standard Distributed7 LOG_MNGR daemon or the user process specified with the ***-o*** option. The logging process must be active and running during an entire log session. If the LOG_MNGR daemon terminates, message logging at all appropriate service endpoints will automatically be deactivated. Also, if a process being logged terminates, logging at all service endpoints associated with that process will automatically be deactivated.

-l Prints a list of any processes for which the message logging capability is currently active. (This option only provides the list, it does not prompt for information.) The QUEUE column indicates whether message logging is active at the read-side and/or write-side queue. See *ebs_ps* for a description of the columns that appear in the display.

-d Deactivates the message logging capability for a particular service endpoint. Deactivation of message logging at an endpoint where it is not currently active has no effect.

-o Enables the user to redirect the logged messages to a process other than the standard Distributed7 LOG_MNGR daemon. By default, messages are logged to the LOG_MNGR daemon. If a user-specified process is used, it must be designed to handle the messages it receives. Normally, a logger process will save the message contents to a file and/or display them to the standard output (see *logd*).

After entering the command to activate or deactivate logging, a prompt will appear for the object type - **named object**, **SS7 object**, **IPC key**, or **MUX object**. After selecting the type, prompts occur to uniquely identify the process of that type.

A named object is a process that does not directly send SS7 messages. An example is a Call Control application that interfaces with the ISUP process. A named object is identified by the name that the process provided in the *spm_open()* and *spm_bind()* functions at registration, up to 13 characters (the 14th is the null character to terminate the string).

An SS7 object is a process that directly communicates with an SS7 protocol multiplexer, such as a TCAP application.

- A TCAP application is uniquely identified by its user part number (3 for SCCP), logical signalling point number (SP), subsystem number (SSN), and instance number. The SP and SSN were specified by the process. The instance number is assigned by the system when the process registers with *tcm_open()*. The value is returned by the function. If only one application is registered with a particular SSN, then the instance number is 1.
- Applications associated with Signalling Network Management are uniquely identified by the logical signalling point number and the user part number of 0.
- The ISUP process is identified by the logical signalling point number and the user part number 5.

To select IPC key, the user must know the IPC key that the system assigned to the process when it registered with *spm_open()* and *spm_bind()*. The process would have retrieved the value from the **IPCkey** field of the **SPMreg_t** structure or by calling *spm_getusrinfo()*, and then would have had to create a way for the operator to access it.

A MUX object is a connection between two STREAMS multiplexers (e.g. UPM, MTP, SCCP, and those listed under *ebs_ps*). To identify a MUX object, the user is prompted for the multiplexer ID and the signalling point number. This information may be seen in the output of *ebs_ps*.

SEE ALSO logd, ebs_ps

8.2.15 ebs_loopback

NAME

ebs_loopback Activates/deactivates message loopback.

SYNOPSIS

ebs_loopback [-l] [-d]

DESCRIPTION

ebs_loopback Invokes the utility which controls the Distributed7 message loopback capabilities. The utility prompts for the information needed to activate or deactivate loopback for a particular service endpoint, a user process or the link between any two adjacent STREAMS multiplexers. This command can also be used to display a list of all service endpoints that have message loopback currently active.

When the message loopback capability at a particular service endpoint is activated, all messages sent out and/or about to be received through the endpoint will be routed to the user-specified process instead of being routed to their normal destinations, i.e., the destination specified within the message. The activation of message loopback at a particular service endpoint affects only the messages originated from the endpoint, not the messages destined for it.

-l Prints a list of those processes for which message loopback is currently active. (This option only provides the list, it does not prompt for information.) The QUEUE column indicates whether message loopback is active at the read-side and/or write-side queue. See *ebs_ps* for a description of the columns that appear in the display.

-d Deactivates the message loopback capability for a particular service endpoint. Deactivation of message logging at an endpoint where it is not currently active has no effect.

For the loopback utility to work successfully, the process which receives the messages must remain active and running during the entire time that loopback is enabled to it. If a process terminates, message loopback will automatically be deactivated at all service endpoints associated with that process as well as at all appropriate endpoints under the Distributed7 platform.

After entering the command to activate or deactivate loopback, prompts will appear for the object type of the endpoint that loopback will occur at and the process where the messages will be diverted. The types are **named object**, **SS7 object**, **IPC key**, or **MUX object**. After selecting the type, prompts occur to uniquely identify each process according to its type. See *ebs_log* for a description of the information required to identify the processes depending on their types.

SEE ALSO ebs_ps

8.2.16 ebs_modinstall

NAME

ebs_modinstall Installs Distributed7 modules.

SYNOPSIS

ebs_modinstall [-f]

DESCRIPTION

ebs_modinstall Installs the Distributed7 STREAMS components, i.e., multiplexers, modules, and device drivers. When executed, it copies the executables from an appropriate product directory to the ***/usr/kernel/drv*** and ***/usr/kernel/strmod*** directories. It also updates the various configuration files associated with the newly-introduced device drivers and creates special files associated with each Distributed7 multiplexer or device driver.

-f Force option. When multiple Distributed7 releases exist on a machine, the ***ebs_modinstall*** script cannot be used for installation. Rather, the ***ebs_setrelease*** script must be used to activate a particular Distributed7 release. The ***-f*** option allows the user to bypass checks regarding multiple Distributed7 releases, and performs the installation in an unconditional manner. Use of this option is restricted to reconfiguration, i.e., adding removing, or replacing the signaling hardware on the machine.

Important*: The $****EBSHOME*** *environment variable must be set before invoking this script, and you must have root privileges to execute this script.*

SEE ALSO ebs_modremove

8.2.17 ebs_modremove

NAME

ebs_modremove Removes Distributed7 modules.

SYNOPSIS

ebs_modremove [-f]

DESCRIPTION

ebs_modremove Removes the Distributed7 STREAMS components, i.e., multiplexers, modules, and device drivers. When executed, it cleans up the appropriate executables in the ***/usr/kernel/drv*** and ***/usr/kernel/strmod*** directories, updates various configuration files on the removed device drivers, and deletes all appropriate special files associated with the Distributed7 multiplexers and device drivers.

-f Force option. When multiple Distributed7 releases exist on a machine, the ***ebs_modremove*** script cannot be used for removal. Rather, the ***ebs_setrelease*** script must be used to deactivate a particular Distributed7 release. The ***-f*** option allows the user to bypass checks regarding multiple Distributed7 releases, and performs the removal in an unconditional manner. Use of this option is restricted to reconfiguration, i.e., adding removing, or replacing the signaling hardware on the machine.

Important*: The $**EBSHOME** environment variable must be set before invoking this script, and you must have root privileges to execute this script.*

SEE ALSO ebs_modinstall

8.2.18 ebs_modunload

NAME

ebs_modunload Unload Distributed7 modules

SYNOPSIS

ebs_modunload

DESCRIPTION

ebs_modunload This script is for unloading, from a Sun platform, the STREAMS components, i.e., multiplexers, modules, and device drivers, comprising the Distributed7 system software.

Note: You must have "root" privileges to execute this script.

SEE ALSO ebs_modremove

8.2.19 ebs_mtpglobal

NAME

ebs_mtpglobal Change the global instance of *upmd* in a distributed environment

SYNOPSIS

ebs_mtpglobal<sp_no><hostname>

DESCRIPTION

ebs_mtpglobal Used to force the system to change the host where the global instance of the *upmd* daemon is running.

By default the first started *upmd* becomes the global instance. Whenever the ***ebs_mtpglobal*** utility is used to change the global instance host, the global *upmd* instance closes its endpoint, where the global address of *upmd* is bound. Then the *upmd* instance on the requested host (***hostname***) registers as the global instance. The global instance of *upmd* can be viewed in the *ebs_ps* output by the **+** sign on the MODE column.

sp_no Specifies the signalling point that is of interest and may assume a value from 0 to 7.

hostname Specifies the hostname where the user wants the global instance of *upmd* daemon to be running.

NOTE: *The ebs_mtpglobal utility requires upmd daemon to be running on the host identified by the* **hostname** *parameter.*

SEE ALSO upmd, ebs_ps

8.2.20 ebs_pkgrm

NAME

ebs_pkgrm Removes the Distributed7 packages.

SYNOPSIS

ebs_pkgrm\ version

DESCRIPTION

The ***ebs_pkgrm*** script is for removing, from a Sun platform, all software packages associated with a user-specified ***version*** (e.g., 1.0.0.1) of the Distributed7 system software. Since Distributed7 product comprises a number of installable software packages, this script provides an alternative [as well as a short-cut] to the UNIX ***pkgrm()*** utility as it frees the user from knowing the names of the individual software packages and/or dependencies between them.

When executed, ***ebs_pkgrm*** script will search the list of all software packages installed on the local host and compile a list of all packages associated with the user-specified Distributed7 release. Subsequently, ***ebs_pkgrm*** will invoke the UNIX ***pkgrm()*** utility to remove these software packages in the appropriate order.

SEE ALSO pkgrm, ebs_setrelease

Note*: You must have "root" privileges to execute this script.*

8.2.21 ebs_ps

NAME

ebs_ps Reports process status.

SYNOPSIS

ebs_ps [-a|d|n|m|s] [-lqtx]

DESCRIPTION

ebs_ps Retrieves and displays a snapshot of information about active processes which are running under the Distributed7 environment. Since the environment is constantly changing, the information is only absolutely true for the instant when it was gathered.

Without options, ***ebs_ps*** displays information about the processes that have the same user ID or group ID as the user who issued the command. Without options, the output contains only the UNIX process ID, process status, operation mode, host machine identifier, STREAMS multiplexer, STREAMS queue identifiers, process type, and process name. Otherwise, the information to be displayed is controlled by the options.

The information displayed by ***ebs_ps*** is based on data stored in a process table on the local host machine. This table contains information about processes running on the local host machine and on all other machines in the distributed network. Individual machines within a network may have differences in the contents of their process tables due to processes that are only registered to the local host machine. The Distributed7 environment has a built-in mechanism which keeps the appropriate portions of the local process tables on the individual machines synchronized at all times.

-a Prints information about all active processes regardless of the process type, i.e., named object, SS7 object, and ownership, i.e., user ID, group ID. Without this option, only information for processes with the same user ID and/or group ID as the issuer of this command will be printed.

-d Prints information about daemon processes executing on any host within network.

-n Prints information about all active named object processes whose user ID or group ID are the same as that of the issuer of the command.

-m Prints information about all STREAMS multiplexers that are in use. STREAMS multiplexers implement the individual layers of the SS7 protocol stack, i.e., MTP, SCCP, TCAP, for individual signalling points. Process ID, user ID, and group ID fields for multiplexer objects are always set to 0 because they are kernel-level entities, not processes.

-s	Prints information about all active SS7 object processes whose user ID or group ID are the same as that of the issuer of the command.
-l	Prints additional information about each process including the assigned internal key and the service type, user ID, and group ID associated with it.
-q	Prints a complete list of the STREAMS read-side queue addresses associated with each process. This option is only meaningful when used with the ***-l*** option.
-t	Print the time of registration for each process. This option is meaningful only when used in conjunction with the -l option.
-x	Indicates that the ***ebs_ps*** utility should not bind an address to the service endpoint associated with it. Unless this option is specified, a named object entry for ***ebs_ps*** will be created in the process table of the local machine. This option allows ***ebs_ps*** to retrieve the contents of the local process table without disturbing it.

OUTPUT VALUES

The output of this command contains several columns of information, depending on the options used in the command. The following table contains the column headings of the output, the meaning of the column, and possible values that may be displayed. The fields that are displayed depend on the command options used.

Table 8-2: ebs_ps Output Description

Column Heading	Valid Values	Description
OBJECT		Object type.
	daemon	Process is a daemon process of the Distributed7 system software. The name associated with the process is printed within brackets.
	nmdobj	Process is a user process that is addressable as a named object. The name of the process is printed within brackets.
	ss7obj	User process that is addressable as an SS7 object. Brackets contain the signalling point and the MTP user part for the process. For SCCP and TCAP applications, the subsystem and instance number are also included.
	tcpobj	Identifies user processes that are addressable as TCP/IP objects (e.g.,TC applications that utilize the TCP/IP transport services). The TCP/IP port number as well as the instance information associated with the process will be printed within brackets.
	muxobj	A STREAMS multiplexer that is in use. Multiplexers implement the SS7 protocol stack for individual signalling points located on a host machine. All multiplexers other than the SPM may have multiple physical instances on a given host machine. There are 5 types:
		spm: Service Provider Module. (one physical instance per host machine)
		upm: Message Transfer Part (MTP) User Part Multiplexer. Used for implementing the MTP Signalling Message Handling (SMH) protocol.
		snm: MTP Signalling Network Management (SNM) Multiplexer.
		sccp: Signalling Connection Control Part (SCCP) Multiplexer.
		tcap: Transaction Capabilities Application Part (TCAP) Multiplexer.

Table 8-2: ebs_ps Output Description

Column Heading	Valid Values	Description
MODE		Operation mode. Combination of the following:
	A	Active mode. Process can receive and send messages.
	S	Standby mode. Process can send messages but cannot receive messages unless the message originator specifies its destination address in the L_IPCKEY format.
	L	Local. Process is addressable on the local machine only; other machines on the network do not have information about its existence. The lack of this flag indicates that the process is known and addressable across the network.
	X	Exclusive. No other process can bind with the address of this process. If L is shown, this restriction applies to processes on the local machine only. Otherwise, it applies to the network.
STAT		Current process status.
	ok	Process exists and operates in the normal state, i.e., it is neither blocked nor in a wait state.
	blkd	Process is in a blocked state. Its read-side STREAMS queue is full. This status may indicate a hung process and/or an overflow condition on the read-side STREAMS message queue.
	wait	Process is in a transient wait state while a software handshake procedure is underway to confirm its network-wide binding request with all machines in the network.
SRV		Service type of the process. Refer to the <***api.h***> header file for a list of service types available.
HOST		Name of the host machine on which the process is executing.
MUX		The STREAMS multiplexer used for establishing the service endpoint for the process on the host machine. Information displayed in this field identifies the multiplexer type, physical instance number [except for the SPM], and the upper stream number associated with the process. Only the stream number is shown for SPM since it only has one instance on a system.
	sccp/#/#	Signalling Connection Control Part (SCCP) Multiplexer.
	snm/#/#	MTP Signalling Network Management (SNM) Multiplexer.
	spm/#	Signalling Point Multiplexer.
	tcap/#/#	Transaction Capabilities Application Part (TCAP) Multiplexer.
	upm/#/#	Message Transfer Part (MTP) User Part Multiplexer. Used for implementing the MTP Signalling Message Handling (SMH) protocol.
PID		UNIX process ID assigned to the process on the host machine named in HOST.
KEY		Internal key assigned to the process on the local host machine. It identifies the slot allocated for the process in the local process table and is in the range from 0 to NMAXPROC.
GID		Group ID associated with the process, in decimal. For all multiplexer objects, this field is set to 0.
UID		User ID associated with the process, in decimal. For all multiplexer objects, this field is set to 0.
1STQADDR		Address, in hex, of the read-side STREAMS queue for the very first connection between the process and the STREAMS multiplexer.
SPMQADDR		Address, in hex, of the read-side STREAMS queue on the SPM multiplexer that should be used when routing messages to the process via the SPM. If equal to 1STQADDR, the process is readily connected to the SPM multiplexer.
UPMQADDR		Address, in hex, of the read-side STREAMS queue on the corresponding UPM multiplexer that should be used when routing messages to the process via the UPM. If equal to 1STQADDR, the process is readily connected to the UPM multiplexer.

Table 8-2: ebs_ps Output Description

Column Heading	Valid Values	Description
SNMQADDR		Address, in hex, of the read-side STREAMS queue on the corresponding SNM multiplexer that should be used when routing messages to the process via the SNM. If equal to 1STQADDR, the process is readily connected to the SNM multiplexer.
SCMQADDR		Address, in hex, of the read-side STREAMS queue on the corresponding SCCP multiplexer that should be used when routing messages to the process via the SCCP. If equal to 1STQADDR, the process is readily connected to the SCCP multiplexer.
TCMQADDR		Address, in hex, of the read-side STREAMS queue on the corresponding TCAP multiplexer that should be used when routing messages to the process via the TCAP. If equal to 1STQADDR, the process is readily connected to the TCAP multiplexer.
TIME		The time at which the process registered.

SEE ALSO ebs_sync, ebs_sysinfo, ebs_qlist

8.2.22 ebs_qinfo

NAME

ebs_qinfo Retrieves STREAMS queue information

SYNOPSIS

ebs_qinfo [-m|s] [-bx]

DESCRIPTION

ebs_qinfo Retrieves and displays a snapshot of information about all the STREAMS queues used by processes operating under the Distributed7 environment on the local host. Queue information from remote hosts is not displayed.

The basic information consists of the read/write queue sizes, byte counts, high water mark settings, and low water mark settings. In addition, an option is available to get the individual priority bands for each queue. Since the environment is constantly changing, the information is only absolutely true for the instant when it was gathered.

The Distributed7 system software allocates a unique pair of STREAMS queues to each process associated with a service endpoint The process exchanges application messages through these queues. The STREAMS queue used for receiving messages by the process is known as the read-side queue, while the queue used for sending messages is referred to as the write-side queue. The immediate pair of read/write queues that are used by the application to receive and send messages are known as the *streamhead* queues. These queues are connected to another pair of queues on the corresponding STREAMS multiplexer, which are called *multiplexer* queues.

-m Indicates that the information about the read/write queues located at the STREAMS multiplexer should be displayed instead of the streamhead.

-s Indicates that the information about the read/write queues located at the streamhead should be displayed. This is the default option.

-b Displays byte count and water mark settings for each individual priority band of the corresponding queue pair. Priority bands distinguish between the exchange of normal and expedited messages (either SS7 or IPC). When this option is specified, a detailed breakdown of the byte count and water mark settings for each priority band will be displayed on the screen. The output is displayed with a separate line for each priority band in the order of: normal SS7 messages (band 0), normal IPC messages (band 1), expedited SS7 messages (band 3), and expedited IPC messages (band 4).

-x	Indicates that ***ebs_qinfo*** should not bind an address to the service endpoint associated with it. Unless this option is specified, a named object entry for ***ebs_qinfo*** will be created in the process table of the local machine.

OUTPUT VALUES

The output of this command contains several columns of information, depending on the options used in the command. The following table contains the column headings of the output and the meaning of the column. The fields that are displayed depend on the command options used.

Table 8-3: ebs_qinfo Output Description

Column Heading	Description
MUX	The STREAMS multiplexer used for establishing the service endpoint for the process on the local host machine. Information displayed in this field identifies the multiplexer type, the physical instance number [for multiplexers other than the SPM], and the clone device (upper stream) number associated with the process. For processes associated with the SPM, the physical instance number is not displayed since SPM has only one instance.
RQSIZE	The total number of messages currently present in the read-side queue.
WQSIZE	The total number of messages currently present in the write-side queue.
RQCOUNT	The total number of bytes in the read-side queue or the corresponding priority band of the read-side queue. If -b option is not specified, the number displayed will be the sum of the byte counts for the individual priority bands.
WQCOUNT	The total number of bytes in the write-side queue or the corresponding priority band of the write-side queue. If -b option is not specified, the number displayed will be the sum of the byte counts for the individual priority bands.
RQHIWAT	The high water mark for the read-side queue or the corresponding priority band of the read-side queue. If -b option is not specified, the high water mark setting for priority band 0 will be displayed.
RQLOWAT	The low water mark for the read-side queue or the corresponding priority band of the read-side queue. If -b option is not specified, the low water mark setting for priority band 0 will be displayed.
WQHIWAT	The high water mark for the write-side queue or the corresponding priority band of the write-side queue. If -b option is not specified, the high water mark setting for priority band 0 will be displayed.
WQLOWAT	The low water mark for the write-side queue or the corresponding priority band of the write-side queue. If -b option is not specified, the low water mark setting for priority band 0 will be displayed.

SEE ALSO ebs_ps, ebs_qlist, ebs_qstat

8.2.23 ebs_qlist

NAME

ebs_qlist Retrieves STREAMS queue list.

SYNOPSIS

ebs_qlist [-x]

DESCRIPTION

ebs_qlist Retrieves and displays a snapshot list of all STREAMS queues that are currently in use by processes operating under the Distributed7 environment. Since the environment is constantly changing, the information is only absolutely true for the instant when it was gathered.

The Distributed7 system software allocates a unique pair of STREAMS queues to each process of a service endpoint for message exchange. The STREAMS queue used for receiving messages by the process is known as the read-side queue, while the queue used for sending messages is called the write-side queue.

The list produced by ***ebs_qlist*** includes the addresses for both read-side and write-side STREAMS queues associated with each process running on the local host machine. Information about STREAMS queues on remote host machines in a network is not displayed since queue addresses are only meaningful on the host machine of the queues.

-x Indicates that ***ebs_qlist*** should not bind an address to the service endpoint associated with it. Unless this option is specified, a named object entry for ***ebs_qlist*** will be created in the process table of the local machine.

DISPLAY

The output of this command contains several columns of information, depending on the options used in the command. The following table contains the column headings of the output and the meaning of the column. The fields that are displayed depend on the command options used.

Table 8-4: ebs_qlist Output Description

Column	Description
MUX	The STREAMS multiplexer used for establishing the service endpoint for the process on the host machine. Information displayed in this field identifies the multiplexer type, the physical instance number [for multiplexers other than the SPM], and the clone device (upper stream) number associated with the process. Since SPM handles all signalling points on a system, only its stream number is displayed.
1STQADDR	Address, in hex, of the read-side STREAMS queue for the very first connection between the process and the STREAMS multiplexer.
SPMQADDR	Address, in hex, of the read-side STREAMS queue on the SPM multiplexer that should be used when routing messages to the process via the SPM. If equal to 1STQADDR, the process is readily connected to the SPM multiplexer.

Table 8-4: ebs_qlist Output Description

Column	Description
UPMQADDR	Address, in hex, of the read-side STREAMS queue on the corresponding UPM multiplexer that should be used when routing messages to the process via the UPM. If equal to 1STQADDR, the process is readily connected to the UPM multiplexer.
SNMQADDR	Address, in hex, of the read-side STREAMS queue on the corresponding SNM multiplexer that should be used when routing messages to the process via the SNM. If equal to 1STQADDR, the process is readily connected to the SNM multiplexer.
SCMQADDR	Address, in hex, of the read-side STREAMS queue on the corresponding SCCP multiplexer that should be used when routing messages to the process via the SCCP. If equal to 1STQADDR, the process is readily connected to the SCCP multiplexer.
TCMQADDR	Address, in hex, of the read-side STREAMS queue on the corresponding TCAP multiplexer that should be used when routing messages to the process via the TCAP. If equal to 1STQADDR, the process is readily connected to the TCAP multiplexer.

SEE ALSO ebs_ps, ebs_qinfo, ebs_qstat

8.2.24 ebs_qstat

NAME

ebs_qstat Retrieves STREAMS queue statistics

SYNOPSIS

ebs_qstat -s -x

DESCRIPTION

ebs_qstat Retrieves and displays various pieces of statistical information about the messages exchanged by the individual processes operating under the Distributed7 environment and running on the local host. Since the environment is constantly changing, the information is only absolutely true for the instant when it was gathered. This information includes:

- total number of messages injected by a process
- total number of deferred messages originated by a process
- total number of messages submitted to a process
- total number of messages that have been subject to flow control prior to being submitted to a process
- total number of messages that have been discarded by the platform (e.g., to help prevent excess message accumulation in the read-side queues associated with the process)

Distributed7 initializes the measurement peg counts associated with each endpoint, i.e., STREAMS connection, when the endpoint is established during an *spm_open()* call and maintains them until the endpoint is removed.

Optionally, the ***ebs_qstat*** utility can be used to retrieve and display current settings of queue management parameters associated with the read-side STREAMS queues of individual processes operating on the local host. These parameters include:

- low/high queue sizes
- low/high age of congestion values

Note that a process can retrieve the current values of the queue management parameters using the *spm_getqparams()* function call. The process can manipulate these parameters using the *spm_setqparams()* function call.

-s Indicates that information on queue management parameters should be displayed. When this command-line option is specified, message statistics associated with the individual processes will not be displayed.

-x Indicates that the ***ebs_qstat*** utility should not bind an address to the service endpoint associated with it. Unless this option is specified, a named object entry for ***ebs_qstat*** will be created in the process table of the local machine.

OUTPUT VALUES

The output of this command contains several columns of information, depending on the options used in the command. The following table contains the column headings of the output and the meaning of the column.

Table 8-5: ebs_qstat Output Description

Field Name	Description
MUX	Specifies the STREAMS multiplexer used for establishing the service endpoint for the process on the local host machine. Information displayed in this field identifies the multiplexer type, the physical instance number [for multiplexers other than the SPM] and the clone device number associated with the process. For processes associated with the SPM, the physical instance number is not displayed since SPM has only one instance.
LOWQSIZE	Specifies the number of messages that should be buffered by the Distributed7 kernel-resident software prior to warning the process that its read-side queues are starting to fill up. At this point there is no room left at the streamhead read-side queue and messages are being buffered at the upper read-side of the STREAMS multiplexer. Note that the size of the streamhead read-side queue associated with a process is controlled by the read-side water marks and not by the LOWQSIZE parameter. See ***spm_stroptions()*** for more information. A ***LOWQSIZE*** value of -1 indicates that the user process will not be notified about message build-ups in its read-side queues until the MAXQSIZE parameter setting is reached.
MAXQSIZE	Specifies the maximum number of messages that should be buffered by the Distributed7 kernel-resident software prior to declaring that the read-side queues associated with the process are completely full. At this point there is no room left either at the streamhead read-side queue or at the upper read-side of the STREAMS multiplexer. When this limit is reached, Distributed7 software discards all subsequent messages going to the process and notifies the process each time a message is discarded. It also pegs an internal measurement count that keeps track of the total number of messages discarded by the platform. A ***MAXQSIZE*** value of -1 means that the Distributed7 software should not discard any messages going to the process and should keep buffering them as long as there is room at the upper-side of the STREAMS multiplexer.
LOWQTIME	Specifies [in terms of milliseconds] the current value of the minimum age of congestion parameter. When a message going to a user process remains in the upper read-side queue of the associated STREAMS multiplexer longer than the value specified by the ***LOWQTIME*** parameter, the Distributed7 software warns the process that messages on its read-side queues are becoming out-of-date. A ***LOWQTIME*** value of -1 indicates that the process will not be notified about out-of-date messages in its read-side queues until the MAXQTIME parameter setting is reached.
MAXQTIME	Specifies [in terms of milliseconds] the current value of the maximum age of congestion parameter. When a message going to a user process remains in the upper read-side queue of the associated STREAMS multiplexer longer than the value specified by the ***MAXQTIME*** parameter, the Distributed7 software determines that this message is out-of-date and discards it. Subsequently, it warns the user process and pegs an internal measurement count. A ***MAXQTIME*** value of -1 indicates that Distributed7 software should not discard messages going to a process and should keep buffering them indefinitely.
INJECTED	The total number of messages injected through the process's write-side STREAMS queue using function calls such as: *spm_snd()*, *spm_broadcast()*, *spm_forward()*. This count does not include the number of deferred messages originated by the process.
DEFERRED	The total number of deferred messages originated by the process using the *spm_tstart()* function call. This count does not include the number of injected messages.
SUBMITTED	The total number of messages on the streamhead read-side queue associated with the user process. These messages may or may not be retrieved by the user process.

Table 8-5: ebs_qstat Output Description

Field Name	Description
FLOWCNTLD	The total number of times messages have been subjected to the STREAMS flow control before being delivered to the streamhead read-side queue associated with the user process. A non-zero value of ***FLOWCNTLD*** indicates that the streamhead read-side queue associated with the process is becoming full and the Distributed7 software is buffering messages at the upper read-side queue of the associated STREAMS multiplexer. Increasing the high water marks associated with the streamhead read-side queue may eliminate the number of times messages experience flow control.
DISCARDED	The total number of messages going to the process but discarded by the Distributed7 platform. Messages are discarded to help prevent excess accumulation in the read-side queues associated with the process. These messages may be discarded on the basis of MAXQSIZE and/or MAXQTIME parameter settings associated with the process. If both parameters are set to -1, no discarding takes place even if the process cannot keep up with the message traffic.

SEE ALSO ebs_ps, ebs_qinfo, ebs_qlist, spm_qparams(), spm_stroptions()

8.2.25 ebs_report

NAME

ebs_report Generates an alarm report.

SYNOPSIS

ebs_report [-b mmddyy ***-e*** mmddyy ***-p pri -d*** dir hostname(s)***]***

DESCRIPTION

ebs_report Collects information from the ***alarmd*** log files stored on the individual host machines in the Distributed7 environment and creates a report. This utility organizes the records chronologically, searches the records for user-specified information, generates customized alarm reports, and displays the reports on the standard output. Without options, ***ebs_report*** generates a report that contains all alarm conditions existing for the local host up to the current point in time. Otherwise, the contents of the report depend on the options specified.

-b mmddyy Includes all alarm conditions that occurred on or after the specified date. The date is specified in the *mmddyy* format, with the month, day of the month, and year expressed in 2-digit numerals (as in the UNIX date(1) command).

-e mmddyy Includes all alarm conditions that occurred on or before the specified date. The date is specified in the *mmddyy* format, with the month, day of the month, and year expressed in 2-digit numerals (as in the UNIX date(1) command).

-p pri Includes alarm conditions with the specified priority levels only. Without this option, the default includes alarm conditions at all priority levels. The ***pri*** argument may contain any combination of the following values:

- 1:Informational messages.
- 2:Messages at minor priority level.
- 3:Messages at major priority level.
- 4:Messages at critical priority level.
- 5:Messages at fatal priority level.

-d dir Locates alarm log files on specified host machines. By default, alarm log files are located under the ***alarmlog*** directory in the ***$EBSHOME/access/RUN*** directory. If the ***dir*** is specified, the alarm log files are expected to be located under the ***dir/alarmlog*** directory.

hostname Identifies the host machine(s) whose alarm log files should be used for generating the report. If multiple hosts are specified, each hostname must be separated from the others by white space. If a hostname is not specified, the report will be generated for the local host only.

***Important**: The **$EBSHOME** environment variable must be set before invoking this utility.*

FILES

$EBSHOME/access/RUN/alarmlog

EXAMPLES

The following are example command lines for ***ebs_report***.

- To display all *critical* alarm messages generated on the local host, up to the current time:
  ebs_report -p 4
- To display all *major* and *critical* alarm messages generated on the host *phantom* since August 14, 1994.
  ebs_report -b 081494 -p 34 phantom
- To display all *minor*, *major*, and *critical* alarm messages generated on the hosts, *sun* and *mars*, between the dates September 3, 1994 and December 7, 1994.
  ebs_report -b 090394 -e 120794 -p 234 sun mars

***Important:** Refrain from executing the **ebs_report** utility on a live system (where several user-space application programs are running) as it may consume a large amount of CPU resources to search through and process the event log files accumulated on the system, which is likely to degrade the performance of user-space applications running on the system.*

SEE ALSO date(1), alarmd, apm_report

8.2.26 ebs_setrelease

NAME

ebs_setrelease Activate specified Distributed7 release.

*NOTE: This script replaces the **ebs_modinstall** script. While the **ebs_modinstall** script still exists, it cannot be used on machines where multiple versions of the Distributed7 software are installed.*

SYNOPSIS

ebs_setrelease version | -i

DESCRIPTION

ebs_setrelease This utility is used for installing, on a Sun platform, the STREAMS components, i.e., multiplexors, modules, and device drivers, associated with a user-specified version of the Distributed7 system software.
When executed, this utility locates the path for the user-specified Distributed7 release, create/update the /etc/amgrhome file if necessary, establish the $EBSHOME/access symbolic link, i.e., to point to the access directory of the specified Distributed7 release, copy the kernel components from under the /usr/kernel/strmod directories, update various configuration files regarding the newly introduced device drivers, and create a number of special device files associated with each Distributed7 multiplexor or device driver.
After copying the Distributed7 kernel components, this script converts the Distributed7 database files from the previous Distributed7 version, if any.

-i Causes ***ebs_setrelease*** to display information about the currently installed release, i.e., version and access directory path.

8.2.27 ebs_showlink

NAME

ebs_showlink Reports link status.

SYNOPSIS

ebs_showlink [-lx]

DESCRIPTION

ebs_showlink Retrieves and displays information about all connections to the SS7 boards and the Network Interface (NI) hardware. In a distributed Distributed7 environment, information on SS7 boards located at remote host machines can also be retrieved. That information includes the link number, link type, host, status, and last change in status. The SS7 and NI connections are established and maintained by the Service Provider Module (SPM) on the local host machine through appropriate STREAMS modules or drivers. STREAMS modules perform the message format translations for all messages flowing through the modules in both upstream and downstream directions.

- TRMOD STREAMS
  The TRMOD STREAMS module connects the SPM and the SS7 board device. SS7 boards provide access to the SS7 signalling link hardware on the local host, and are used to send and receive SS7 messages over the SS7 links. The *spmd* daemon establishes and maintains the SS7 device driver connections when Distributed7 system software is started on the local host machine with *ebs_start*. Once a connection to the SS7 board driver is made, it remains in that state until Distributed7 system software on the local host machine is stopped, or until a manual request is placed to reconfigure the corresponding SS7 device connection.
- NIMOD STREAMS
  The NIMOD STREAMS module connects the SPM and the TCP/IP protocol suite. The NI hardware establishes a reliable, connection-oriented, i.e., TCP/IP based, interface between individual machines in a distributed Distributed7 environment for inter-machine message exchange. The *netd* daemon establishes and maintains the TCP/IP connections to remote machines, and must communicate with its peers on the remote host machines to set the TCP/IP connections up. While a TCP/IP connection is in service, optional heartbeat messages can be exchanged periodically over that link to monitor its health. To end a TCP/IP connection, a disconnect request, a disconnect indicator, or an error message from the TCP/IP protocol suite must

occur to cause the *netd* daemon to tear down the corresponding connection.

-l Prints additional information about each link, including the internet address of the link and heartbeat-related information (see Table 8-6).

-x Indicates that ***ebs_showlink*** should not bind an address to the service endpoint associated with it. Unless this option is specified, a named object entry for ***ebs_showlink*** will be created in the process table of the local machine.

OUTPUT VALUES

The output of this command contains several columns of information, depending on the options used. The following table contains column headings of the output, the meaning of the column, and possible values that may be displayed. The fields that are displayed depend on the command options used.

Table 8-6: ebs_showlink Output Description

Column	Valid Values	Description
LINK		Lower stream number on the SPM multiplexer that is connected to an appropriate STREAMS driver. In current implementation, link numbers 0 through 7 are reserved for connections to the SS7 driver module and link numbers 8 through 15 are for connections to the TCP/IP protocol suite.
TYPE		Type of the connection, i.e., the hardware device associated with the link.
	sbs334	Indicates that the link interconnects the SPM multiplexer to the sbs334 device driver which supports SBS334, SBS370, and SBS372 boards.
	pci334	Indicates that the link interconnects the SPM multiplexer to the pci334 device driver which supports PCI334, PCI370, and PCI372 boards.
	pci3xpq	Indicates that the link interconnects the SPM multiplexer to the pci3xpq device driver which supports PCI370PQ and PCI372PQ boards.
	pci3xapq	Indicates that the link interconnects the SPM multiplexer to the pci3xapq device driver, which supports PCI370APQ and PCI372APQ boards.
	cpc37xpq	Indicates that the link interconnects the SPM multiplexer to the cpc37xpq device driver, which supports CPC370APQ and CPC372PQ boards.
	pmc8260	Indicates that the link interconnects the SPM multiplexor to the pmc8260 device driver which supports the PMC8260 board.
	artic8260	Indicates that the link interconnects the SPM multiplexor to the artic8260 device driver which supports the ARTIC1000 and ARTIC2000 boards.
	pmc4539	Indicates that the link interconnects the SPM multiplexer to the pmc4539 device driver, which support the PMC4539F board.
	vbrd	Indicates that the link interconnects the SPM multiplexer to the Distributed7 Virtual Board (VBRD) device driver.
	ecp	Indicates that the link interconnects the SPM multiplexer to an SS7 board device driver of unknown type, i.e., that does not belong to the aforementioned set of device drivers.
	tcp/ip	Indicates that the link interconnects the SPM multiplexer to the TCP/IP protocol suite; therefore, is used to communicate with a remote host machine on the network.
HOST		Name of the host machine associated with the link. For SS7 board connections, name identifies the host where the SS7 signalling hardware is physically located. For the TCP/IP connections, name identifies the remote host machine accessible via that link. Third-party hosts that are not equipped with the Distributed7 software are marked with a question mark tag (?) at the end.

Table 8-6: ebs_showlink Output Description

Column	Valid Values	Description
RMTHOST		Field that contains the name of the remote host machine that is accessible through the TCP/IP connection. Third-party hosts not equipped with Distributed7 software are marked with a question mark tag (?) at the end.
INETADDR		IP address of the host (in the dotted decimal notation).
STAT		Current status of the connection between the SPM and the STREAMS device driver.
	L	Linked. Connection is in place.
	U	Unlinked. Connection is not in place.
	A	Active mode of operation. Messages can be exchanged across the connection.
	S	Standby mode of operation. Connection is not being used for exchanging messages.
	B	Blocked. Connection cannot be used because it is not possible to exchange data across the TCP/IP STREAMS modules.
	I	Isolated, i.e., disconnected LAN interface
TIME		Last date and time that the link status changed between *linked* and *unlinked*.The time stamp is based on the system clock of the local host machine.
HBEAT		Current heartbeat status for the link.
	-	No heartbeat mechanism on the link.
	ok	Remote host is responding on time to heartbeat requests originated over the link by the local host.
	failed	Remote host machine has failed to respond on time to one or more heartbeat requests originated over the link by the local host.
HBACT		Action to be taken when the heartbeat mechanism over the link fails.
	0	No action. However, the link heartbeat status is changed to *failed*.
	1	Update the process table on the local host machine by removing all entries that belong to processes on the remote host machine that failed to respond to heartbeat messages. The process tables on both host machines will automatically be synchronized when the heartbeat mechanism over the link is restored.
HBINT		Length of the heartbeat interval in milliseconds.
HBSENT		Indicates the total number of heartbeat responses originated by the local host and sent across the link.
HBRCVD		Indicates the total number of heartbeat responses originated by the remote host, i.e., in response to heartbeat requests originated by the local hosts, and received across the link.
SEQNUM		The sequence number of the last message received and processed over the link

SEE ALSO netd, spmd, and ebs_hbeat

8.2.28 ebs_shutdown

NAME

ebs_shutdown Stops Distributed7 software, remotely

SYNOPSIS

ebs_shutdown [hostname(s) ... ***]***

DESCRIPTION

ebs_shutdown Requests the *spmd* daemon of the local host to send out a request to stop the Distributed7 system software and all related applications on one or more remote machines. The user will be prompted to confirm the execution of this action.

hostname Names of the hosts to be shutdown (including their applications). If no hostname is specified, the *spmd* daemon will relay this request to all host machines configured in the network, including the local host machine.

Note: User confirmation of the execution of this command is requested since this command will result in a non-functional Distributed7 environment.

SEE ALSO ebs_stop, apm_stop

8.2.29 ebs_start

NAME

ebs_start Starts Distributed7 software.

SYNOPSIS

ebs_start

DESCRIPTION

ebs_start Starts the Distributed7 system software on the local host and configures the system according to the instructions specified in the ***apmconfig*** input file.

On hosts equipped with the *apmd* daemon process, this utility executes *apmd* in the *exclusive* mode. The *apmd* process accesses the *apmconfig* file to determine which other processes to start.

On hosts where the *apmd* daemon is not available, this script will only execute the *spmd* daemon process.

NOTE: Starting with Distributed7 Release 1.0.0, the apmd daemon process is the only designated daemon for process creation and management. The spmd daemon can no longer start up processes from an input file.

Important*: The* ***$EBSHOME*** *environment variable must be set before invoking this utility.*

Important*: To add, remove, or reposition Sbus boards, follow the steps shown in the Installation Manual (also see ebs_modremove).*

SEE ALSO apmd, spmd, apm_start, ebs_shutdown, ebs_stop, apm_stop

8.2.30 ebs_stop

NAME

ebs_stop Stops Distributed7 software.

SYNOPSIS

ebs_stop

DESCRIPTION

ebs_stop Requests the *spmd* daemon to shut down the applications registered to Distributed7 and the Distributed7 system software on the local host machine.

The user will be prompted to confirm that the system should be stopped. Then, a signal is sent to all the user processes (see *spm_bind()* in the *API Reference Manual*). A process can receive the signal and perform cleanup operations before it is stopped. The utility returns control to the user after all system and user processes that are registered with the Distributed7 environment have terminated. Drivers and software components are disassembled and removed in the reverse order that they were assembled by *spmd* with *ebs_start*.

After executing this command, the Distributed7 environment will no longer be functional, until it is restarted with *ebs_start*.

Important*: The* ***$EBSHOME*** *environment variable must be set before invoking this utility.*

SEE ALSO ebs_ps, ebs_shutdown, ebs_start

8.2.31 ebs_sync

NAME

ebs_sync Synchronizes dynamic data across the network.

SYNOPSIS

ebs_sync [-x] hostname

DESCRIPTION

ebs_sync Issues a manual request to synchronize the dynamic data of a remote host to the local host or of the local host to all the remote hosts of an Distributed7 network. The Distributed7 environment automatically synchronizes the relevant portions of all dynamic data tables on the individual host machines. Under normal circumstances, this command is not needed. This command simply provides a means to manually synchronize the dynamic data if the automatic synchronization mechanism fails to operate properly.

Warning*: Use of this utility during moderate-to-heavy message traffic may result in the loss of a significant number of messages.*

In a distributed computing environment, or network, each host machine in the network must share the same view of the environment at all times. Each machine contains critical dynamic data which should be continuously available to all other machines in the network. Dynamic data includes information about objects executing on a particular host machine and about SS7 signalling link hardware existing on the machine. When this information is available to all machines within a network, objects on the individual host machines can communicate with each other and use the SS7 signalling link hardware on any host machine in the network.

The synchronization utility updates all appropriate dynamic database tables on a specified remote host with the information contained in the tables of the local machine. The remote host invalidates all appropriate entries in its local version of the tables and then replaces them with the new entries from the local machine. The local machine can also request that its tables be updated with information from all the remote hosts.

-x Inhibits ***ebs_sync*** from binding an address to the service endpoint associated with it. Unless this option is specified, a named object entry for ***ebs_sync*** will be created in the process table of the local machine.

hostname Identifies host whose tables will be synchronized.

- Remote host name: entries for the local host that are in the remote host's tables will be updated with the information from the local host's tables.

- Local host or no name: all entries for remote hosts that are in the local host’s tables will be updated with information sent from each respective remote host.

***Note**: Dynamic data synchronization is only relevant to distributed Distributed7 configurations. Using this command in a stand-alone configuration has no effect.*

SEE ALSO ebs_audit, ebs_hbeat

8.2.32 ebs_sysinfo

NAME

ebs_sysinfo Shows host machine information.

SYNOPSIS

ebs_sysinfo [-ax] [-l] [-m mode]

DESCRIPTION

ebs_sysinfo Obtains information about the current hardware and software configuration on the local host machine. The basic command, with no options, displays the system name, node name, operating system release, operating system version number, system kernel architecture, machine internet address, and operation mode. Alternate host names and the internet addresses associated with those names are also displayed for multi-homed hosts. All zero's will be displayed [within brackets] for the internet address if the internet address to be used by the kernel-level Distributed7 components has not been initialized to its proper value.

-a Initializes the internet address to be used by the kernel-level Distributed7 components on the local machine. This option is normally not needed because either *spmd* or *netd* usually initializes the internet address at system startup time.

-l This function is used to retrieve and display licensing information on the local host.

-x Indicates that ***ebs_sysinfo*** should not bind an address to the service endpoint associated with it. Unless this option is specified, a named object entry for ***ebs_sysinfo*** will be created in the process table of the local machine.

-m mode Specifies the current operation mode for the local host machine, either *server* or *client*. Currently, mode setting information is not used by the system. However, the *spmd* daemon initializes the operation mode to *server* at system startup time.

SEE ALSO showrev

8.2.33 ebs_tasklist

NAME

ebs_tasklist	Retrieves task list information.

SYNOPSIS

ebs_tasklist [-r] [-x]

DESCRIPTION

ebs_tasklist	Retrieves and displays information about the task lists that are created and maintained by the kernel-resident NewNet Communication Technologies, LLC Distributed7 system software. Task lists are used as a means of message-based communication between kernel-space threads running on a specified host.
r	Resets the number of counts, i.e., maximum number of entries and total number of entries, associated with each task list.
x	Prevents the ***ebs_tasklist*** utility from binding an address to the service endpoint associated with it. Default is for a named object entry for ***ebs_tasklist*** to be created in the process table of the local machine.

DISPLAY FORMATS

LISTSTAT	Specifies the current status, i.e., valid or invalid, of the task list.
LISTHEAD	Specifies the kernel-space address of the head of the linked list that is associated with the task list. If no entries are currently on the task list, then this field assumes a zero value.
LISTTAIL	Specifies the kernel-space address of the tail of the linked list that is associated with the task list. If no entries are currently on the task list, then this field assumes a zero value.
FREEFUNC	Specifies the kernel-space address of the clean-up function to be called when deleting entries listed on the task list at the time of task list destruction. If no clean-up function is specified, then this field assumes a zero value.
CNT	Specifies the number of entries currently available on the task list.
MAXCNT	Specifies the maximum number of entries that have accumulated on the task list since its creation.
TOTALCNT	Specifies the total number of entries that have enqueued on the task list since its creation.

NOTES

*Because the environment can change while **ebs_tasklist** is running, the snapshot it produces is valid only for a split second, and it may not be accurate by the time you see it.*

8.2.34 ebs_tune

NAME

ebs_tune Tunes operating system parameters.

SYNOPSIS

ebs_tune [-d]

DESCRIPTION

ebs_tune Modifies operating system parameters associated with the STREAMS subsystem and IPC semaphores, shared memory segments, and message queues for the operational needs of the Distributed7 environment on a Sun hardware platform. The parameters manipulated by this script are contained in the ***/etc/system*** configuration file and are used by the kernel during initialization of the system.

The ***ebs_tune*** script should only be executed once - following the execution of *ebs_modinstall* during installation of the Distributed7 system software.

When executed, ***ebs_tune*** creates a backup copy of the ***/etc/system*** file and names the backup, ***/etc/system.old***. Then, it reads the contents of the ***/etc/system*** file and appends to [or deletes from] it a set of instructions for the kernel parameters, if those parameters have not already been customized. If some of the parameters have already been customized, a message is displayed on the screen after manipulation of their current values. The system must be re-booted after executing this utility for changes to be made to certain kernel parameters.

-d Deletes changes introduced to the ***/etc/system*** configuration file by executing the ***ebs_tune*** script.

Important*: You must have root privileges to execute this script. The UNIX system must be re-booted after execution of this script for the changes to take effect.*

FILES

/etc/system

/etc/system.old

SEE ALSO ebs_modinstall, system(4), sysdef(1)

8.2.35 getcfg

NAME

getcfg Gets information about SS7 controllers in the system.

SYNOPSIS

getcfg

DESCRIPTION

getcfg Used to get information about the SS7 controllers in the system. It displays output in column format informing the user about the driver, type of board, instance number of the board, physical number of the slot carrying the board, and the physical slot information.

Note: *Use of the* ***ebs_modremove*** *and the* ***ebs_modinstall*** *commands (located under* ***$EBSHOME/access/install****) is required to get a correct* ***getcfg*** *output when any of the following actions are performed with the boards installed in the host system: removing a board from the system, adding a board to the system, replacing a board with another board of a different type.*

Below are the explanations for each of the columns displayed by the ***getcfg*** utility:

Driver - the name of the driver used to access the board. Driver name is one of the following:

- sbs334 - the sbs334 device driver that supports SBS334, SBS370, and SBS372 boards.
- pci334 - the pci334 device driver that supports PCI334, PCI370, and PCI372 boards.
- pci3xpq - the pci3xpq device driver that supports PCI370PQ and PCI372PQ boards.
- pci3xapq - the pci3axpq device driver that supports PCI370APQ and PCI372APQ boards.
- cpc3xpq - the cpc3xpq device driver that supports CPC370PQ and CPC372PQ boards.
- pmc8260 - the pmc8260 device driver that supports the PMC8260 board.
- artic8260 - the artic8260 driver that supports the ARTIC1000 and ARTIC2000 boards.
- pmc4539 - the pmc4539 device driver that supports the PMC4539F board.
- adaxm - the adaxm device driver that supports the HDCII-LPe board.

Board Type - the type of the board. Board type is one of the following:

- sbs334 - the sbus SS7 controller that supports up to four 64 Kbps links.
- sbs370 - the common name for sbus sbs370 (T1) and sbs372 (E1) SS7 controllers which support up to four 64 Kbps links over T1/E1 spans.
- pci334 - the pci bus SS7 controller that supports up to four 64 Kbps links.
- pci370 - the pci bus SS7 controller that supports up to four 64 Kbps links over T1 spans.
- pci372 - the pci bus SS7 controller that supports up to four 64 Kbps links over E1 spans.
- pci370pq - the pci bus SS7 controller which supports up to twenty-four 64 Kbps links over T1 spans.
- pci372pq - the pci bus SS7 controller which supports up to twenty-four 64 Kbps links over E1 spans.
- pci370apq - the pci bus SS7 controller which supports up to twenty-four 64 Kbps links over T1 spans.
- pci372apq - the pci bus SS7 controller which supports up to twenty-four 64 Kbps links over E1 spans.
- cpc370pq - the CompactPCI bus SS7 controller with 16 MB on board RAM which supports up to twentyfour 64 Kpbs links over T1 spans.
- cpc372pq - the CompactPCI bus SS7 controller with 16 MB on board RAM which supports up to twentyfour 64 Kpbs links over E1 spans.
- pmc8260 - the CompactPCI bus SS7 controller with 32 MB on board RAM which supports up to sixtyfour 64 Kpbs links over E1/T1 spans.
- pmc4539 - the PMC SS7 controller with 128 MB SDRAM which supports up to four High Speed Links or 128 64 Kbps links over E1/T1 spans.
- artic1000 - the CompactPCI bus SS7 controller with 32 MB on board RAM which supports up to sixtyfour 64 Kpbs links over E1/T1 spans.
- artic2000 - the PCI bus SS7 controller with 32 MB on board RAM which supports up to 64 Kpbs links over E1/T1 spans.
- HDCII-LPe - the PCIe bus SS7 controller which supports up to 124 links over E1/T1 spans.

Note: *Although the PCI3xPQ, PCI3xAPQ and CPC37xPQ boards allow configuration of up to 24 links, use of more than 16 links is not recommended for systems requiring full bandwidth on all configured links.*

Instance - instance number of the board among other boards of its type. Its value can be in the range of 0 to 7.

Slot - physical number of slot carrying the board. Its value depends on the hardware configuration of the host computer.

Slot Info - information about the physical PCI slot the board uses. It displays the clock speed, mechanical specification and electrical specification.

State - state of the device driver (applicable to AIX systems only).

SAMPLE OUTPUT

Solaris (CompactPCI bus)

```
Driver     Board Type  Instance  Slot   Slot Info
---------  ----------  --------  -----  --------------
artic8260  artic1000   0         2      33MHz-32/64bit
pmc8260    pmc8260     0         5      33MHz-32/64bit
```

Solaris (PCI bus)

```
Driver     Board Type  Instance  Slot   Slot Info
---------  ----------  --------  -----  ----------------------
pmc4539    pmc4539     0         5      33/66MHz-32/64bit-3.3V
pci3xapq   pci372apq   0         2      33/66MHz-32/64bit-3.3V
pci3xpq    pci370pq    0         1      33MHz-32/64bit-5V
```

AIX

```
Driver    Board Type  Slot  Instance   State
--------  ----------  ----  --------   ---------
pci334    pci370      1     0          Available
pci334    pci334      3     1          Available
pci3xpq   pci372pq    2     0          Available
pci3xapq  pci372apq   4     0          Available
```

8.3 APM Utilities

8.3.1 apm_audit

NAME

apm_audit Audit *apmd* IPC resources.

SYNOPSIS

apm_audit

DESCRIPTION

apm_audit Used to audit the UNIX Inter Process Communication (IPC) resources used by the *apmd* daemon. These resources include the message queues, shared memory segments, and semaphores acquired by *apmd* at start-up time and used by itself or with other UNIX processes attached to the *apmd* domain.

The auditing procedure conducted by the ***apm_audit*** utility simply involves a detailed listing of all the IPC resources allocated by the *apmd* daemon and the contents of information displayed is very much the same as that of the UNIX ***ipcs*** command. The ***apm_audit*** utility is intended to detect potential anomalies with IPC resources associated with the APM subsystem (e.g., accumulation in message queues). No corrective action is taken as part of the auditing procedure conducted by ***apm_audit***.

SEE ALSO apmd and apm_start

8.3.2 apm_getstate

NAME

apm_getstate Retrieves current *apmd* run state.

SYNOPSIS

apm_getstate [-c*|****s****|****x] [-h*** host***]***

DESCRIPTION

apm_getstate	Retrieves the current run state of the *apmd* daemon process on a specified host, operating in the Distributed7 environment.
-c	Retrieves run state of the AccessCRP version of *apmd*. The ***$PRODID*** and ***$RUNID*** environment variables must be set to appropriate values.
-s	Retrieves the run state of the AccessSERVICES version of *apmd*. The ***$DOMID*** environment variable must be set to an appropriate value.
-x	Retrieves the run state of the Distributed7 version of *apmd*. The ***$EBSHOME*** environment variable must be set to an appropriate value.
-h host	Retrieves the run state of *apmd* on a remote host identified by ***host***. The *apmd*s on both the remote and local host must be operational since the request is placed through the local *apmd*. If the option is not provided, the run state of the *apmd* on the local host will be retrieved.

Since *apmd* supports multiple versions, if the user does not explicitly specify one in the command, then *apmd* determines the version by the following logic:

- If ***$DOMID*** is set, it assumes an AccessSERVICES environment.
- If ***$DOMID*** is not set but ***$PRODID*** and ***$RUNID*** are set, it assumes the AccessCRP (Call Routing Point) environment.
- If none of the above environment variables are set, but ***$EBSHOME*** is set, it assumes the basic Distributed7 environment.

SEE ALSO apmd, apm_setstate

8.3.3 apm_kill

NAME

apm_kill Sends a signal to a process.

SYNOPSIS

apm_kill [-c|s|x] [-l] [-n signum***] [-h*** host***] -p|g|t pid|gid|tag***

DESCRIPTION

apm_kill Places a request to send a UNIX signal to a process or a group of processes executing in the Distributed7 environment.

-c Assumes the AccessCRP version of *apmd*. The ***$PRODID*** and ***$RUNID*** environment variables must be set to appropriate values.

-s Assumes the AccessSERVICES version of *apmd*. The ***$DOMID*** environment variable must be set to an appropriate value.

-x Assumes the basic Distributed7 version of *apmd*. The ***$EBSHOME*** environment variable must be set to an appropriate value.

-l Prints a list of symbolic signal names supported by ***apm_kill***. This list includes only commonly used signals and is only a subset of those supported by the UNIX *kill* command. The list shows the names without the SIG prefix.

-n signum Sends the signal identified in ***signum*** to the specified process(es). The valid entries for ***signum*** can be numeric or the symbolic names that are listed by the ***-l*** option. If no value is provided, the default signal, *SIGTERM*, is sent, which normally kills processes that do not catch or ignore the signal.

-h host Sends a signal to a process executing on a remote host identified by ***host***.The *apmd*s on both the remote and local host must be operational since the request is placed through the local *apmd*. If the option is not provided, the local host is the default.

-p pid Sends the signal to the process whose UNIX process ID is ***pid***.

-g gid Sends the signal to the processes whose group ID is ***gid***. This group ID is the one specified in the *apmd* configuration file, i.e. *apmconfig*, and it could be different from the process's UNIX group ID.

-t tag Sends the signal to the process identified by ***tag***. This tag is specified in the *apmd* configuration file, i.e., *apmconfig*. The tag of a process can be obtained by looking at the configuration file of the appropriate host or by executing *apm_ps*.

Since *apmd* supports multiple versions, if the user does not explicitly specify one in the command, then *apmd* determines the version by the following logic:

- If ***$DOMID*** is set, it assumes an AccessSERVICES environment.

- If ***$DOMID*** is not set but ***$PRODID*** and ***$RUNID*** are set, it assumes the AccessCRP (Call Routing Point) environment.
- If none of the above environment variables are set, but ***$EBSHOME*** is set, it assumes the basic Distributed7 environment.

SEE ALSO apmd, apm_stop, apm_killall, apm_kill

8.3.4 apm_killall

NAME

apm_killall Sends a signal for all processes to terminate.

SYNOPSIS

apm_killall [-c|s|x] [-h** host**]

DESCRIPTION

apm_killall Requests the *apmd* on a specific host to terminate all non-failsafe processes and then initialize its run state.

-c Assumes the AccessCRP version of *apmd*. The ***$PRODID*** and ***$RUNID*** environment variables must be set to appropriate values.

-s Assumes the AccessSERVICES version of *apmd*. The ***$DOMID*** environment variables must be set to an appropriate value.

-x Assumes the basic Distributed7 version of *apmd*. The ***$EBSHOME*** environment variable must be set to an appropriate value.

-h host Terminates processes executing on a remote host identified by the ***host*** argument. The *apmd*s on both the remote and local host must be operational since the request is placed through the local *apmd*. If the option is not provided, the local host is the default.

The *apmd* terminates a process by first sending a *SIGTERM* signal to it. If the process does not terminate within 3 seconds, *apmd* sends a *SIGKILL* signal. After all processes are terminated, *apmd* will change its run state to the default initialization state specified in the *initdefault* entry of the *apmd* configuration file. If an *initdefault* entry does not exist, then *apmd* enters a run state as follows:

- In AccessCRP environments, it moves to the **D** state.
- In AccessSERVICES environments, it moves to the **A** state.
- In Distributed7 environments, it moves to the **init** state.

Processes that are defined to operate in the failsafe mode (by their entry in the *apmd* configuration file) are not effected by the operations of this utility. Examples of fail-safe processes are *mlogd*, *spmd*, and *netd*.

Since *apmd* supports multiple versions, if the user does not explicitly specify one in the command, then *apmd* determines the version by the following logic:

- If ***$DOMID*** is set, it assumes an AccessSERVICES environment.
- If ***$DOMID*** is not set but ***$PRODID*** and ***$RUNID*** are set, it assumes the AccessCRP (Call Routing Point) environment.
- If none of the above environment variables are set, but ***$EBSHOME*** is set, it assumes the basic Distributed7 environment.

SEE ALSO apmd, apm_stop, apm_kill, apm_setstate

8.3.5 apm_ps

NAME

apm_ps Reports process status.

SYNOPSIS

apm_ps [-c|s|x] [-l]

DESCRIPTION

apm_ps Retrieves and displays information about active processes that were spawned by the *apmd* daemon on the local host machine. Only the processes spawned based on the configuration file are included in the output. Processes spawned dynamically through the *apm_spawn()* function are not part of the output. The elements included in the output are described in Table 8-7. This data is maintained by the *apmd* daemon and stored in a process table located on the local host machine.

-c Assumes the AccessCRP version of *apmd*. The ***$PRODID*** and ***$RUNID*** environment variables must be set to appropriate values.

-s Assumes the AccessSERVICES version of *apmd*. The ***$DOMID*** environment variables must be set to an appropriate value.

-x Assumes the basic Distributed7 version of *apmd*. The ***$EBSHOME*** environment variable must be set to an appropriate value.

-l Prints additional information about each process including the internal keys assigned to the process, its group ID, and various states that the *apmd* daemon should switch to based on process behavior.

Since *apmd* supports multiple versions, if the user does not explicitly specify one in the command, then *apmd* determines the version by the following logic:

- If ***$DOMID*** is set, it assumes an AccessSERVICES environment.
- If ***$DOMID*** is not set but ***$PRODID*** and ***$RUNID*** are set, it assumes the AccessCRP (Call Routing Point) environment.
- If none of the above environment variables are set, but ***$EBSHOME*** is set, it assumes the basic Distributed7 environment.

Note*: The output of* ***apm_ps*** *is a snapshot that is true only for a split-second because of the dynamic nature of the information. Therefore, it may not be completely accurate after it is displayed.*

OUTPUT VALUES

The output of this command contains several columns of information, depending on the options used in the command. The following table contains the column headings of the output, the meaning of the column, and possible values that may be displayed. The fields that are displayed depend on the command options used.

Table 8-7: apm_ps Output Description

Column Heading	Possible Values	Description
KEY		Internal key assigned to the process on the local host. Identifies the slot allocated for the process in the dynamic process table maintained by the *apmd* daemon.
RKEY		Registration related key assigned to the process on the local host. Identifies the slot allocated for the process in the Distributed7 internal registration table. It corresponds to the KEY field in the output of the *ebs_ps* command. For processes that are spawned by *apmd* but do not register with the Distributed7 environment, this field is set to 0.
PID		UNIX process ID assigned to the process on its host machine.
GID		Group ID assigned to the process in the *apmd* configuration file [if any]. This group ID could be different from the UNIX group ID of the process.
PROG		Program ID assigned to the process in the *apmd* configuration file.
TAG		Tag associated with the process. Constructed by combining the process tag information specified in the *apmd* configuration file with the node name of the host machine. All processes executing on a particular host must be assigned unique process tags.
ACTION		The action mode defined for the process in the *apmd* configuration file. Key words for this field are described in *apmconfig*.
STATUS		The status of the process. An `**\***` next to a value in this field indicates the *apmd* daemon is no longer executing. Information retrieved/displayed may not be accurate.
	ok	Process executing normally.
	busy	*apmd* is busy executing a scenario that involves the process.
	exited	Process terminated with a zero exit code.
	failed	Process terminated with a non-zero exit code. Depending on the ACTION field defined for the process in the *apmd* configuration file, process may be re-spawned by *apmd*.
	killed	Process terminated by an unexpected signal. Depending on the ACTION field defined for the process in the *apmd* configuration file, process may be re-spawned by *apmd*.
	stopped	Process has stopped.
HBSTAT		Heartbeat status.
	-	Process will not exchange heartbeat messages with *apmd*.
	ok	Process is responding to the heartbeat request messages generated by *apmd* on a regular basis.
	failed	Process failed to respond to the heartbeat request messages generated by *apmd* on a regular basis. Process will be killed by *apmd*.
	n/a	*apmd* unable to send heartbeat request messages to the process since process has not yet registered with the Distributed7 environment.
RETRY		Number of times process has been re-spawned by *apmd* following the initial start-up.
SSTATE		Start-up and steady-state success states for the process defined in the *apmd* configuration file. The two states are separated from each other by the ***:*** character. The **-** character is used to identify *don't care* states.
FSTATE		Start-up and steady-state failure states for the process defined in the *apmd* configuration file. The two states are separated from each other by the ***:*** character. The **-** character is used to identify *don't care* states.
HSTATE		Start-up and steady-state hopeless states for the process defined in the *apmd* configuration file. The two states are separated from each other by the ***:*** character. The **-** character is used to identify *don't care* states.

Table 8-7: apm_ps Output Description

Column Heading	Possible Values	Description
ASTATE		Start-up and steady-state positive acknowledgment states for the process defined in the *apmd* configuration file. The two states are separated from each other by the ***:*** character. The **-** character is used to identify *don't care* states.
NSTATE		Start-up and steady-state negative acknowledgment states for the process defined in the *apmd* configuration file. The two states are separated from each other by the ***:*** character. The **-** character is used to identify *don't care* states.
ESTATE		Execution states for the process defined in the *apmd* configuration file. Multiple execution states [if any] are separated from each other by the | character.

SEE ALSO ps(1), ebs_ps, apmd, apm_init() [API call]

8.3.6 apm_report

NAME

apm_report Generates a log report.

SYNOPSIS

apm_report [-b mmddyy***] [-e*** mmddyy***] [-p pri] [-d*** dir***] [-f*** file***] [-m***|***a] [***hostname(s)***]***

DESCRIPTION

apm_report Collects information from the *mlogd* log files stored on the individual host machines in the Distributed7 environment and creates a report. This utility organizes the records chronologically, searches the records for user-specified information, generates customized log reports, and displays the reports on the standard output. Without options, ***apm_report*** generates a report that contains all log messages existing in the master log files on the local host, up to the current point in time. Otherwise, the contents of the report depend on the options specified.

-b mmddyy Includes all log messages reported to *mlogd* on or after the specified date. The date is specified in the *mmddyy* format, with the month, day of the month, and year expressed in 2-digit numerals (as in the UNIX date(1) command).

-e mmddyy Includes all log messages reported to *mlogd* on or before the specified date. The date is specified in the *mmddyy* format, with the month, day of the month, and year expressed in 2-digit numerals (as in the UNIX `date(1)` command).

-p pri Includes log messages reported to *mlogd* with the specified priority levels only. Without this option, the default includes log messages at all priority levels. The ***pri*** argument may contain any combination of the following values:

- 1:Informational messages.
- 2:Messages at minor priority level.
- 3:Messages at major priority level.
- 4:Messages at critical priority level.

-f file Includes only the log messages that were generated by the executable whose source file is specified in the ***file*** argument. By default, all log messages are included, regardless of the name of the source file.

-d dir Locates master/alternative log files on specified host machines. By default, master/alternative log files are located under ***mlog*** and ***alog*** directories, respectively, in the ***$EBSHOME/access/RUN*** directory. If the ***dir*** is specified, the master/alternative log files are expected to be located under the ***dir/mlog*** and ***dir/alog*** directories, respectively.

-m Includes only the log messages from the master log files (default).

-a Includes only the log messages from the alternate [secondary] log files.

hostname Identifies the host machine(s) whose log files should be used for generating the report. If multiple hosts are specified, each hostname must be separated from the others by white space. If a hostname is not specified, the report will be generated for the local host only.

Important*: The* ***$EBSHOME*** *environment variable must be set before invoking this utility.*

FILES

$EBSHOME/access/RUN/mlog/MLog.*mmddyy*

$EBSHOME/access/RUN/alog/ALog.*mmddyy*

EXAMPLES

The following are example command lines for ***apm_report***.

- To display all *critical* log messages generated on the local host, up to the current time, and stored in the master log files:

  apm_report -p 4

- To display all *major* and *critical* log messages stored in the alternate log files that were generated since August 14, 1994 by the executable, named *sample.c*, which is on the host, *phantom*.

  apm_report -b 081494 -p 34 -f sample.c -a phantom

- To display all *minor*, *major*, and *critical* log messages in the master log files that were generated on the hosts, *sun* and *mars*, between the dates September 3, 1994 and December 7, 1994.

  apm_report -b 090394 -e 120794 -p 234 sun mars

Important: *Refrain from executing the* ***apm_report*** *utility on a live system (where several user-space application programs are running) as it may consume a large amount of CPU resources to search through and process the event log files accumulated on the system, which is likely to degrade the performance of user-space applications running on the system.*

SEE ALSO date(1), mlogd, ebs_report

8.3.7 apm_setstate

NAME

apm_setstate Manipulates *apmd* run state.

SYNOPSIS

apm_setstate [-c|s|x] [-h host***] newstate***

DESCRIPTION

apm_setstate Changes the current run state of the *apmd* daemon process on a host operating under Distributed7 environment. The change in state causes *apmd* to execute the *apmconfig* configuration file.

-c Changes the run state of the AccessCRP version of *apmd*. The ***$PRODID*** and ***$RUNID*** environment variables must be set to appropriate values.

-s Changes the run state of the AccessSERVICES version of *apmd*. The ***$DOMID*** environment variables must be set to an appropriate value.

-x Changes the run state of the basic Distributed7 version of *apmd*. The ***$EBSHOME*** environment variable must be set to an appropriate value.

-h host Changes the run state of *apmd* on a remote host identified by ***host***. The *apmd*s on both the remote and local host must be operational since the request is placed through the local *apmd*. If the option is not provided, the local host is the default.

newstate Changes the run state to the provided state. The entries in the configuration file (*apmconfig* or *apmconfig.old*) which have an execution state matching this state will be executed.

Depending on which entries of the configuration file are executed, a change in state may result in a change in the environment (e.g. new processes started and/or existing processes terminated). Since *apmd* supports multiple versions, if the user does not explicitly specify one in the command, then *apmd* determines the version by the following logic:

- If ***$DOMID*** is set, it assumes an AccessSERVICES environment.
- If ***$DOMID*** is not set but ***$PRODID*** and ***$RUNID*** are set, it assumes the AccessCRP (Call Routing Point) environment.
- If none of the above environment variables are set, but ***$EBSHOME*** is set, it assumes the basic Distributed7 environment.

Important*: The *apmd* daemon may reject a request to change its current run state if it is busy executing a scenario, possibly at a different run state. If this happens,* ***apm_setstate*** *will fail with an appropriate error code.*

SEE ALSO apmd, apm_getstate, apm_killall, apm_update

8.3.8 apm_start

NAME

apm_start Starts the *apmd* daemon.

SYNOPSIS

apm_start [-c|s|x] [-f** cfgfile**]

DESCRIPTION

apm_start Sets up the trace shared memory segment of the default size using *apm_trinit* and then starts the *apmd* daemon process on the local host. If the trace shared memory already exists, then only *apmd* will be started. Upon start-up, *apmd* will create and manage a set of processes as defined in the configuration file.

-c Starts the AccessCRP (Call Routing Point) version of *apmd*. The ***$PRODID*** and ***$RUNID*** environment variables must be set to appropriate values prior to program execution. This version supports multiple application domains on a single host. The settings of the environment variables are used to invoke the *apmd* instance.

-s Starts the AccessSERVICES version of *apmd*. The ***$DOMID*** environment variable must be set to an appropriate value prior to program execution. This version supports multiple application domains on a single host. The setting of the environment variable is used to invoke the *apmd* instance.

-x Starts the normal Distributed7 version of *apmd*. The ***$EBSHOME*** environment variable must be set to an appropriate value prior to program execution. This version does not support multiple application domains on a single host.The *apmd* daemon executes in a local-exclusive mode, i.e., only one instance of this version of *apmd* may be executing on a host.

-f cfgfile Use the configuration file specified by ***cfgfile*** instead of the default. By default, *apmd* uses the *apmconfig* or *apmconfig.old* configuration file under an appropriate release directory. (See *apmd*.)

The *apmd* supports different types of application environments, as described by the options (c, s, x). If the user does not explicitly specify one of the options, *apmd* will determine the appropriate environment based on environment variable settings and the following logic:

- If ***$DOMID*** is set, it assumes an AccessSERVICES environment.
- If ***$DOMID*** is not set but ***$PRODID*** and ***$RUNID*** are set, it assumes an AccessCRP environment.
- If none of the above environment variables are set but ***$EBSHOME*** is set, it assumes a basic Distributed7 environment.

SEE ALSO apmd, apm_stop

8.3.9 apm_stop

NAME

apm_stop Terminates the *apmd* daemon.

SYNOPSIS

apm_stop [-c|s|x] [-h** host**]

DESCRIPTION

apm_stop Terminates the *apmd* daemon and the processes that it had spawned through the configuration file. The daemon and its processes can be stopped on any host machine operating under the Distributed7 environment.

-c Assumes the AccessCRP version of *apmd*. The ***$PRODID*** and ***$RUNID*** environment variables must be set to appropriate values.

-s Assumes the AccessSERVICES version of *apmd*. The ***$DOMID*** environment variables must be set to an appropriate value.

-x Assumes the basic Distributed7 version of *apmd*. The ***$EBSHOME*** environment variable must be set to an appropriate value.

-h host Stops process and *apmd* on a remote host identified by ***host***. The *apmd*s on both the remote and local host must be operational since the request is placed through the local *apmd*. If the option is not provided, the local host is the default.

Since *apmd* supports multiple versions, if the user does not explicitly specify one in the command, then *apmd* determines the version by the following logic:

- If ***$DOMID*** is set, it assumes an AccessSERVICES environment.
- If ***$DOMID*** is not set but ***$PRODID*** and ***$RUNID*** are set, it assumes the AccessCRP (Call Routing Point) environment.
- If none of the above environment variables are set, but ***$EBSHOME*** is set, it assumes the basic Distributed7 environment.

The *apmd* terminates the processes that it has spawned differently for the different *apmd* versions.

AccessSERVICES and AccessCRP Versions

1. *apmd* warns all of its active processes by sending the *SIGTERM* signal to them.
2. *apmd* waits up to 60 seconds for the processes to exit.
3. If a process still has not exited, *apmd* will warn this process one more time by sending a second *SIGTERM* signal to it
4. *apmd* waits up to 3 seconds for the process to terminate.
5. If the process still has not exited, *apmd* terminates the process forcefully by sending a *SIGKILL* signal.

6. After all processes exit and/or are terminated, *apmd* terminates itself with an exit code of zero.

Distributed7 Version

The basic version of *apmd* terminates processes that are spawned by it and other processes currently using the Distributed7 platform.

1. *apmd* sends the *SIGTERM* signal to processes that are not currently registered with the Distributed7 environment. (This is the only notification these processes will receive.)
2. *apmd* issues a local system software shutdown request to its active processes.
3. *apmd* waits up to 60 seconds for its processes to exit.
4. If any of its processes still has not exited, *apmd* will warn each process by sending a *SIGTERM* signal to it
5. *apmd* waits up to 3 seconds for its remaining process(es) to terminate.
6. If a process still has not exited, *apmd* terminates the process forcefully by sending a *SIGKILL* signal.
7. After all processes exit and/or are terminated, *apmd* terminates itself with an exit code of zero.

SEE ALSO apmd, apm_start, apm_killall, apm_kill, apm_setstate

8.3.10 apm_trcapture

NAME

apm_trcapture Captures trace output.

SYNOPSIS

apm_trcapture [-c|s|x] [[-o] | [[-d dir|-f file] [-l limit]]] [{-m|n} mask0,...,mask63]
[{-p|r} progid0,...,progid511***]***

DESCRIPTION

apm_trcapture Displays the current values of the trace mask settings on the local host to the standard output. By default, trace mask settings are displayed for all program IDs, if at least one of the 64 trace masks is currently activated for any of the 512 program IDs.

-c Captures contents of the trace buffer for the AccessCRP version of *apmd*. The ***$PRODID*** and ***$RUNID*** environment variables must be set to appropriate values.

-s Captures contents of the trace buffer for the AccessSERVICES version of *apmd*. The ***$DOMID*** environment variables must be set to an appropriate value.

-x Captures contents of the trace buffer for the basic Distributed7 version of *apmd*. The ***$EBSHOME*** environment variable must be set to an appropriate value.

-o Displays trace information captured on ***stdout*** only. When this option is specified, captured trace messages will not be stored in a log file.

Note*: This option cannot be used in combination with any of the following options:*
[-d dir]
[-f file]
[-l limit]

-m mask0,...,mask63 Captures the trace message(s) specified in ***masks***. A maximum of 64 trace masks can be specified (0-63). Multiple trace masks must be separated from each other with the **,** character and no white space. A range of masks may be specified with the **-** character. All mask settings may be modified by specifying the string **all** as the ***masks*** argument.

-n mask0,...,mask63 Captures all trace messages EXCEPT the one(s) specified in ***masks***. Masks can be specified in the range from 0 to 63. Multiple trace masks must be separated from each other with the **,** character and no white space. A range of masks may be specified with the **-** character.

-p progid0,...,progid511 Captures the trace message(s) for the processes with the program IDs specified in ***progid***. A maximum of 512 program IDs can be specified (0-511). Multiple program IDs must be separated from each other using the **,** character with no white space. A range of program IDs may be specified using the **-** character (see example).

-r progid0,...,progid511 Captures the trace message(s) for all processes EXCEPT the ones whose program IDs are specified in ***progid***. A maximum of 512 program IDs can be specified (0-511). Multiple program IDs must be separated from each other using the **,** character with no white space. A range of program IDs may be specified using the **-** character.

-d dir Saves captured trace log files on specified host machines. By default, trace log files are located under ***tracelog*** directory in the ***$EBSHOME/access/RUN*** directory. If the ***dir*** is specified, the trace log files are stored under the ***dir/tracelog*** directory. If ***dir*** directory (or ***tracelog*** sub-directory) does not exist, ***apm_trcapture*** will make an attempt to create all necessary directories.

Note*: This option cannot be used in combination with the [-f file] or [-o] options.*

-f file Saves captured trace log information of filename specified by ***file*** argument. By default, all trace log files will be named with the "TLog" prefix and contain the process ID of the executing program as an extension. This option gives users the flexibility to name trace log files using their own naming conventions and store them in their preferred directories.

Note*: This option cannot be used in combination with the [-d dir] or [-o] options.*

-l limit Limits the number of trace statements to be stored in ***tracelog*** file at any given time to the value specified via ***limit*** argument. Once the specified limit is exceeded, the file is truncated to zero length and writing continues from the beginning of file. This option allows users to control the maximum size of ***tracelog*** files generated by ***apm_trcapture*** utility.

Note*: This option cannot be used in combination with the [-o] option.*

Since *apmd* supports multiple versions, if the user does not explicitly specify one in the command, then *apmd* determines the version by the following logic:

- If ***$DOMID*** is set, it assumes an AccessSERVICES environment.
- If ***$DOMID*** is not set but ***$PRODID*** and ***$RUNID*** are set, it assumes the AccessCRP (Call Routing Point) environment.
- If none of the above environment variables are set, but ***$EBSHOME*** is set, it assumes the basic Distributed7 environment.

EXAMPLES

apm_trcapture -m 1,3,5 -p 0,-,23

Captures trace messages corresponding to trace categories 1, 3, and 5 for program ID's 0 through 23.

apm_trcapture -m 10,20 -p all

Captures trace messages for trace categories 10 and 20 for all program ID's.

apm_trcapture -n 1,-,15 -r 0,-,200

Captures trace messages for all trace categories other than 1 through 15 and for all processes whose program ID's are above 200.

FILES

`$EBSHOME/access/RUN/tracelog/TLog.pid`

SEE ALSO apm_trinit, apm_trclear, apm_trgetmask

8.3.11 apm_trclear

NAME

apm_trclear Clears the contents of the trace shared memory.

SYNOPSIS

apm_trclear [-c|s|x]

DESCRIPTION

apm_trclear Clears the contents of the local host's IPC shared memory segment which is used for tracing the execution of application programs. This memory segment is also referred to as the *trace buffer.* (Tracing occurs through the ***libapm*** trace macros.)

-c Assumes the AccessCRP version of *apmd*. The ***$PRODID*** and ***$RUNID*** environment variables must be set to appropriate values.

-s Assumes the AccessSERVICES version of *apmd*. The ***$DOMID*** environment variables must be set to an appropriate value.

-x Assumes the basic Distributed7 version of *apmd*. The ***$EBSHOME*** environment variable must be set to an appropriate value.

Since *apmd* supports multiple versions, if the user does not explicitly specify one in the command, then *apmd* determines the version by the following logic:

- If ***$DOMID*** is set, it assumes an AccessSERVICES environment.
- If ***$DOMID*** is not set but ***$PRODID*** and ***$RUNID*** are set, it assumes the AccessCRP (Call Routing Point) environment.
- If none of the above environment variables are set, but ***$EBSHOME*** is set, it assumes the basic Distributed7 environment.

Important: *The **apm_trclear** utility should be used with care. It will permanently erase all trace messages logged for the specified environment.*

SEE ALSO apm_trinit, apm_trgetmask, apm_trsetmask, apm_trshow, apm_trcapture, apm_trace()

8.3.12 apm_trgetmask

NAME

apm_trgetmask Retrieves trace mask settings.

SYNOPSIS

apm_trgetmask [-c|s|x] [-e] [{-p|r} progid0,...,progid511***]***

DESCRIPTION

apm_trgetmask Displays the current values of the trace mask settings on the local host to the standard output. By default, trace mask settings are displayed for all program IDs, if at least one of the 64 trace masks is currently activated for any of the 512 program IDs.

-c Displays trace mask settings for the AccessCRP version of *apmd*. The ***$PRODID*** and ***$RUNID*** environment variables must be set to appropriate values.

-s Displays trace mask settings for the AccessSERVICES version of *apmd*. The ***$DOMID*** environment variables must be set to an appropriate value.

-x Displays trace mask settings for the basic Distributed7 version of *apmd*. The ***$EBSHOME*** environment variable must be set to an appropriate value.

-e Displays any ***pattern*** strings associated with individual trace masks for specified program IDs. The pattern strings comprise regular expressions and are set forth with the *apm_trsetmask* command line utility. If a pattern string exists and there is a match between the actual trace message contents and the regular expression, trace statements associated with the corresponding trace mask will be generated.

-p progid0,...,progid511 Displays the trace mask settings for the processes with the program IDs specified in ***progid***. A maximum of 512 program IDs can be specified (0-511). Multiple program IDs must be separated from each other using the **,** character with no white space. A range of program IDs may be specified using the **-** character (see example).

-r progid0,...,progid511 Displays the trace mask settings for all processes EXCEPT the ones whose program IDs are specified in ***progid***. A maximum of 512 program IDs can be specified (0-511). Multiple program IDs must be separated from each other using the **,** character with no white space. A range of program IDs may be specified using the **-** character.

Since *apmd* supports multiple versions, if the user does not explicitly specify one in the command, then *apmd* determines the version by the following logic:

- If ***$DOMID*** is set, it assumes an AccessSERVICES environment.

- If ***$DOMID*** is not set but ***$PRODID*** and ***$RUNID*** are set, it assumes the AccessCRP (Call Routing Point) environment.
- If none of the above environment variables are set, but ***$EBSHOME*** is set, it assumes the basic Distributed7 environment.

EXAMPLES

apm_trgetmask -p 1,3,5,10,-,20
Retrieves the current trace mask settings for program IDs 1, 3, 5, and 10 through 20.

SEE ALSO apm_trinit, apm_trclear, apm_trsetmask, apm_trshow, apm_trcapture, apm_trace(),

8.3.13 apm_trinit

NAME

apm_trinit Initializes IPC shared memory.

SYNOPSIS

apm_trinit [-f] [-c|s|x] [-m maxcnt***]***

DESCRIPTION

apm_trinit Creates and initializes the IPC shared memory segment used during the tracing of application programs on the local host.

-f Forces the initialization of the trace shared memory if it already exists. Normally, a user is not allowed to re-initialize the trace shared memory segment if it already exists.

-c Initializes trace shared memory for the AccessCRP version of *apmd*. The ***$PRODID*** and ***$RUNID*** environment variables must be set to appropriate values.

-s Initializes trace shared memory for the AccessSERVICES version of *apmd*. The ***$DOMID*** environment variables must be set to appropriate values.

-x Initializes trace shared memory for the basic Distributed7 version of *apmd*. The ***$EBSHOME*** environment variable must be set to an appropriate value.

-m maxcnt Specifies the size of the IPC shared memory segment to be initialized. The ***maxcnt*** argument specifies the size in number of messages. The size should never be made less than the default size of 1000 messages.

The trace shared memory segment is a circular buffer storing trace messages. The trace messages are generated by applications using the ***libapm*** trace macros. The shared memory segment must be initialized prior to the start-up of the *apmd* daemon and any other application program that will use the ***libapm***.
Since *apmd* supports multiple versions, if the user does not explicitly specify one in the command, then *apmd* determines the version by the following logic:

- If ***$DOMID*** is set, it assumes an AccessSERVICES environment.
- If ***$DOMID*** is not set but ***$PRODID*** and ***$RUNID*** are set, it assumes the AccessCRP (Call Routing Point) environment.
- If none of the above environment variables are set, but ***$EBSHOME*** is set, it assumes the basic Distributed7 environment.

Important*: The* ***apm_trinit*** *utility cannot be used while the apmd daemon on the local host is running. It should only be used before the apmd daemon is started.*

SEE ALSO apm_trclear, apm_trgetmask, apm_trsetmask, apm_trshow, apm_trcapture, apm_trace()

8.3.14 apm_trsetmask

NAME

apm_trsetmask Sets a trace mask.

SYNOPSIS

apm_trsetmask [-c|s|x] [-a] [{-m|n} mask0,...,mask63] [{-p|r} progid0,...,progid511] [pattern]

DESCRIPTION

apm_trsetmask Sets a trace mask or changes the current values of the trace mask settings available on the local host. Without any options, all 64 trace masks for all 512 program IDs will be set. The ***pattern*** command line argument specifies pattern strings of regular expressions for selective tracing under a particular trace mask setting and program ID.

-c Modifies mask setting for the AccessCRP version of *apmd*. The ***$PRODID*** and ***$RUNID*** environment variables must be set to appropriate values.

-s Modifies mask setting for the AccessSERVICES version of *apmd*. The ***$DOMID*** environment variables must be set to an appropriate value.

-x Modifies mask setting for the basic Distributed7 version of *apmd*. The ***$EBSHOME*** environment variable must be set to an appropriate value.

-a Appends the specified trace mask(s) to any existing ones.

-m mask0,...,mask63 Modifies the trace mask(s) specified in ***masks***. A maximum of 64 trace masks can be specified (0-63). Multiple trace masks must be separated from each other with the **,** character and no white space. A range of masks may be specified with the **-** character. All mask settings may be modified by specifying the string **all** as the ***masks*** argument.

-n mask0,...,mask63 Modifies all trace masks EXCEPT the one(s) specified in ***masks***. Masks can be specified in the range from 0 to 63. Multiple trace masks must be separated from each other with the **,** character and no white space. A range of masks may be specified with the **-** character.

-p progid0,...,progid511 Manipulates the trace mask settings for the processes with the program IDs specified in ***progid***. A maximum of 512 program IDs can be specified (0-511). Multiple program IDs must be separated from each other using the **,** character with no white space. A range of program IDs may be specified using the **-** character (see example).

-r progid0,...,progid511 Manipulates the trace mask settings for all processes EXCEPT the ones whose program IDs are specified in ***progid***. A maximum of 512 program IDs can be specified (0-511). Multiple program IDs must be separated from each other using the **,** character with no white space. A range of program IDs may be specified using the **-** character.

Since *apmd* supports multiple versions, if the user does not explicitly specify one in the command, then *apmd* determines the version by the following logic:

- If ***$DOMID*** is set, it assumes an AccessSERVICES environment.
- If ***$DOMID*** is not set but ***$PRODID*** and ***$RUNID*** are set, it assumes the AccessCRP (Call Routing Point) environment.
- If none of the above environment variables are set, but ***$EBSHOME*** is set, it assumes the basic Distributed7 environment.

Distributed7 system software supports a total of 64 separate trace masks for each program ID. A total of 512 program IDs are available on a given host. Therefore, 512 processes may be uniquely traced and identified. If more than 512 processes co-exist, some processes will be assigned the same program ID. The result of such a situation is that trace messages generated by both programs that share an ID would be displayed on the screen together.

EXAMPLES

apm_trsetmask -m 1,3,5 -p 0,-,23

Sets the masks corresponding to trace categories 1, 3, and 5 for program ID's 0 through 23.

apm_trsetmask -m 10,20 -p all

Sets trace categories 10 and 20 for all program IDs.
Note: The same result could also be achieved by not specifying the -p option.

apm_trsetmask -m 3 -p 5 '^lset=[A-Z]+ link=[0-7] '

Sets the mask corresponding to trace category 3 for program ID 5 selectively (if the trace message contents match the regular expression specified by the ***pattern*** argument).

SEE ALSO apm_trinit, apm_trclear, apm_trgetmask, apm_trshow, apm_trcapture, apm_trace()

8.3.15 apm_trshow

NAME

apm_trshow Displays the trace output.

SYNOPSIS

apm_trshow [-c|s|x] [{-m|n} mask0,...,mask63***] [{-p|r}*** prodig0,...,progid511***]***

DESCRIPTION

apm_trshow Displays the trace information that is in the IPC shared memory segment of the local host. (Tracing shows the execution of application programs.) Without any options, all trace messages in the shared memory segment will be displayed on the standard output. The options allow selective display of certain messages from memory.

-c Displays contents of the trace buffer for the environment that is controlled by the AccessCRP version of *apmd*. The ***$PRODID*** and ***$RUNID*** environment variables must be set to appropriate values.

-s Displays contents of the trace buffer for the environment that is controlled by the AccessSERVICES version of *apmd*. The ***$DOMID*** environment variables must be set to an appropriate value.

-x Displays contents of the trace buffer for the environment that is controlled by the basic Distributed7 version of *apmd*. The ***$EBSHOME*** environment variable must be set to an appropriate value.

-m mask0,...,mask63 Displays the trace mask(s) specified in ***masks***. A maximum of 64 trace masks can be specified (0-63). Multiple trace masks must be separated from each other with the **,** character and no white space. A range of masks may be specified with the **-** character. All mask settings may be modified by specifying the string **all** as the ***masks*** argument.

-n mask0,...,mask63 Displays all trace masks EXCEPT the one(s) specified in ***masks***. Masks can be specified in the range from 0 to 63. Multiple trace masks must be separated from each other with the **,** character and no white space. A range of masks may be specified with the **-** character.

-p progid0,...,progid511 Displays the trace mask settings for the processes with the program IDs specified in ***progid***. A maximum of 512 program IDs can be specified (0-511). Multiple program IDs must be separated from each other using the **,** character with no white space. A range of program IDs may be specified using the **-** character (see example).

-r progid0,...,progid511 Displays the trace mask settings for all processes EXCEPT the ones whose program IDs are specified in ***progid***. A maximum of 512 program IDs can be specified (0-511). Multiple program IDs must be separated from each other using the **,** character with no white space. A range of program IDs may be specified using the **-** character.

Since *apmd* supports multiple versions, if the user does not explicitly specify one in the command, then *apmd* determines the version by the following logic:

- If ***$DOMID*** is set, it assumes an AccessSERVICES environment.
- If ***$DOMID*** is not set but ***$PRODID*** and ***$RUNID*** are set, it assumes the AccessCRP (Call Routing Point) environment.
- If none of the above environment variables are set, but ***$EBSHOME*** is set, it assumes the basic Distributed7 environment.

EXAMPLES

apm_trshow -m 1,3,5 -p 0,-,23
Displays the trace messages for the trace categories 1, 3, and 5 for program ID's 0 through 23.

apm_trshow -m 10,20 -p all
Displays the trace messages for trace categories 10 and 20 for all program ID's.

apm_trshow -n 1,-,15 -r 0,-,200
Displays trace messages for all trace categories other than 1 through 15 and for all processes whose program IDs are above 200.

SEE ALSO apm_trinit, apm_trclear, apm_trgetmask, apm_trsetmask, apm_trcapture, apm_trace()

8.3.16 apm_update

NAME

apm_update Informs *apmd* of any changes in the configuration.

SYNOPSIS

apm_update [-c|s|x] [-h** host**]

DESCRIPTION

apm_update Notifies the *apmd* daemon process on a specified host that changes in the *apmd* configuration file have occurred. It causes the *apmd* to re-read and re-execute the instructions in the configuration file, based on its current state. It copies the updated file into its internal process table.

-c Assumes the AccessCRP version of *apmd* and re-reads the *apmconfig.old* configuration file. The ***$PRODID*** and ***$RUNID*** environment variables must be set to appropriate values.

-s Assumes the AccessSERVICES version of *apmd* and re-reads the *apmconfig.old* configuration file. The ***$DOMID*** environment variables must be set to an appropriate value.

-x Assumes the basic Distributed7 version of *apmd* and re-reads the *apmconfig* configuration file. The ***$EBSHOME*** environment variable must be set to an appropriate value.

-h host Causes an update of the *apmd* on a remote host identified by ***host***. The *apmd*s on both the remote and local host must be operational since the request is placed through the local *apmd*. If the option is not provided, the local host is the default.

Since *apmd* supports multiple versions, if the user does not explicitly specify one in the command, then *apmd* determines the version by the following logic:

- If ***$DOMID*** is set, it assumes an AccessSERVICES environment.
- If ***$DOMID*** is not set but ***$PRODID*** and ***$RUNID*** are set, it assumes the AccessCRP (Call Routing Point) environment.
- If none of the above environment variables are set, but ***$EBSHOME*** is set, it assumes the basic Distributed7 environment.

Important*: The *apmd* daemon may reject a request to re-execute and update the configuration file if it is busy executing a scenario. If this happens, **apm_update** will fail with an appropriate error code.*

SEE ALSO apmd, apm_setstate

8.4 DSM Utilities

8.4.1 dsm_apidemo

NAME

dsm_apidemo Demonstrates the capabilities of the Distributed Shared Memory (DSM) library functions.

SYNOPSIS

dsm_apidemo [-x]

DESCRIPTION

dsm_apidemo Starts a menu-driven program which demonstrates the basic set of capabilities provided as part of the Distributed7 Distributed Shared Memory (DSM) Applications Programming Interface (API) library (***libdsm***). The *dsmd* process on all involved hosts must be running.

-x Indicates that the program should not bind an address to the service endpoint associated with it. This option allows the DSM capabilities to be used by processes that are not registered to the Distributed7 environment. If this option is not specified, a named object entry for ***dsm_apidemo*** will be created in the process table of the local machine.

This program allows the user to:

- acquire/destroy a DSM segment identifier
- attach/detach a DSM segment to the data segment of the calling process
- acquire/release read-only or read-write locks on a DSM segment which must be done to perform consistent read/write operations through the segment
- read/write from/to a specified region of a DSM segment
- retrieve/manipulate various pieces of information about a DSM segment
- retrieve/manipulate DSM optional parameter settings

8.4.2 dsm_audit

NAME

dsm_audit Audits distributed shared memory dynamic data.

SYNOPSIS

dsm_audit [-a|l|m|q|s] [-h** host**]

DESCRIPTION

dsm_audit Places a manual request to audit the dynamic data and IPC communication resources associated with the *dsmd* process on a specified host. The *dsmd* processes on the local and specified hosts must be running.

Without any options, all dynamic data records and IPC communication resources maintained by the *dsmd* process on the local host will be audited. Otherwise, the audit is limited and defined by the command-line options.

-a Audits address records maintained by *dsmd* on the specified host. Address records are created when a process calls *dsm_attach()*. They are maintained until the process calls the *dsm_attach()* or *dsm_destroy()* functions or until the process associated with the address record terminates. This audit identifies and removes address records that belong to non-existing processes.

-l Audits lock records maintained by *dsmd* on the specified host. Lock records are created when a process calls ***dsm_lock()***. They are maintained until the process calls the *dsm_unlock()*, *dsm_detach()*, or *dsm_destroy()* functions or until the process associated with the lock record terminates. This audit identifies and removes lock records that belong to non-existing processes so other lock requests can be serviced.

-m Audits segment records maintained by the *dsmd* daemon on the specified host. Segment records are created as a result of *dsm_get()* function call by a process and maintained until a corresponding *dsm_destroy()* function call is issued. The main motivation in auditing segment records is to identify segment records associated with non-existing shared memory segments and delete them.

-q Audits the IPC message queue used by *dsmd* on the specified host, which is used to communicate with application processes on that host. This audit identifies and discards unattended messages (e.g. messages that belong to non-existing application processes) to prevent message accumulation.

-s Audits service records maintained by *dsmd* on the specified host. Service records are created when a process calls any ***libdsm*** function. They are

	maintained until the call is serviced by *dsmd* or the process associated with the service record terminates. This audit detects timeout conditions.
-h host	Places the audit request to the *dsmd* process on the specified host. The default is the local host.

Important*: The dsmd process has an automatic mechanism to periodically audit all dynamic data and IPC communication resources as follows:*

- lock records - every second,
- service records - every 10 seconds,
- main segment records - every 60 seconds,
- address records - every 60 seconds,
- IPC message queue - every 300 seconds,

Therefore, execution of this command is not normally required. This command simply provides a means to manually audit when an audit is desired between intervals.

8.4.3 dsm_bm

NAME

dsm_bm Benchmarks Distributed7 DSM framework

SYNOPSIS

dsm_bm -d -k key ***-s***

DESCRIPTION

dsm_bm Benchmarks the performance of the Distributed7 Distributed Shared Memory (DSM) framework in terms of the total number of read-write operations that can be performed within a specified time interval. The *dsmd* daemons on all involved hosts must be running.

When executed, ***dsm_bm*** will prompt the user for the specifics of the benchmark test to be performed (e.g., size and nature of the read-write locks to be acquired, number of worker threads). Subsequently, ***dsm_bm*** will create the user-specified number of threads and perform a total of 10,000 overlapping or non-overlapping read-write operations across a DSM segment. If multiple worker threads are in use, they will work together to perform the DSM read-write operations in parallel.

After the specified number of DSM read-write operations are completed, ***dsm_bm*** will calculate the average lock/unlock times involved in performing these operations and estimate the overall system performance in the terms of number of DSM read-write operations [of specified size] per second. The results will be displayed to the user on *stdout*.

-d Indicates that the benchmark tests should be conducted in the "debug" mode. In debug mode, statistics regarding each DSM lock/unlock operation will be displayed on *stdout*.

-k key Uses the IPC key base specified by the ***key*** argument when allocating the DSM segments necessary for conducting the benchmark tests. This option allows multiple instances of the ***dsm_bm*** utility to be executed in a concurrent yet non-intrusive manner. By default, the ***dsm_bm*** utility uses the key base 2000 and allocates three consecutive DSM segments with keys 2000, 2001, and 2002. While one of these DSM segments is used for the actual read-write operations, the other two are used for storing operational parameters and test results. All three DSM segments are destroyed upon completion of the benchmark tests.

-s Indicates that the *dsmd* daemon is operating in the stand-alone mode; therefore, DSM benchmark tests should be conducted in the stand-alone mode.

8.4.4 dsm_list

NAME

dsm_list Displays distributed shared memory information.

SYNOPSIS

dsm_list [-a|l|m|s] [-d] [-h** host**]

DESCRIPTION

dsm_list Displays information about the dynamic data records maintained by the *dsmd* process on a specified host. The *dsmd* processes on the local and specified hosts must be running.

Without any options, information is displayed about all existing DSM segments that the local host has data on. Otherwise, the type of information and the source host is defined by the command-line options.

-a Lists the address records maintained by *dsmd* on the specified host. Address records of a host include only those processes that are executing on that host and are attached to DSM segments. Address records are created when a process calls *dsm_attach()*. They are maintained until the process calls the *dsm_detach()* or *dsm_destroy()* functions or until the process associated with the address record terminates.

-l Lists lock records maintained by *dsmd* on the specified host. Lock records on the individual hosts must be identical since locks are considered global resources. Lock records are created when a process calls ***dsm_lock()***. They are maintained until the process calls the *dsm_unlock()*, *dsm_detach()*, or *dsm_destroy()* functions or until the process associated with the lock record terminates.

-m Lists DSM segment records maintained by *dsmd* on the specified host. Segment records on the individual hosts must be identical since they are considered global resources. Segment records are created when a process calls ***dsm_get()***. They are maintained until the process calls the *dsm_destroy()* function.

-s Lists service records maintained by *dsmd* on the specified host. Service records are created when a process calls any ***libdsm*** function. They are maintained until the call is serviced by *dsmd* or the process associated with the service record terminates. Service records on a host include the processes executing on that host which use ***libdsm*** and those remote processes which need the local *dsmd* process for coordination of multi-host DSM operations.

-d Lists detailed information.

-h host Retrieves information from the *dsmd* process on the specified host. The default is the local host.

OUTPUT VALUES

The output of this command contains several columns of information, depending on the options used in the command. The following table contains the column headings of the output, the meaning of the column, and possible values that may be displayed.

Table 8-8: dsm_list Output Column Description

Output Field Name	Description	Keyword Values
ID	DSM segment identifier.	-
KEY	Key associated with the DSM segment.	-
ADDR	Address at which the local copy of the DSM segment is attached to the data segment of the calling process.	-
MODE	Access mode of the DSM segment in octal form.	-
SIZE	Size of the DSM segment, in bytes.	-
HOST	When no option is specified in the command, this field contains information about the host through which the DSM segment has been created. When the -a option is specified, it contains information about the host for which the address records are retrieved and displayed. When the -l or -s option is specified, it contains information about the host through which the lock or service record was initially acquired.	-
PID	When no option is specified in the command, this field contains the UNIX process ID of the process on the specified host that created the DSM segment. Otherwise, it contains the process ID of the process on the specified host that is associated with a specific address, lock, or service record.	-
THR	Thread ID associated with a specific lock or service record.	-
UID	Effective user ID associated with the DSM segment.	-
GID	Effective group ID associated with the DSM segment.	-
PERM	Access permissions associated with a particular attachment of a DSM segment by the calling process.	read-only or read-write
LOCKID	Identifier associated with a particular instance of a lock record. It is assigned by the system when dsm_lock() is called. It is released when the corresponding dsm_unlock() function is called by the originating process.	-
LOCKTYPE	Specifies whether the corresponding lock has been acquired for read-only or read-write purposes.	read-only or read-write
FROM	Marks the beginning of the segment region protected by the specified lock. Expressed in terms of a byte-offset [inclusive] from the beginning of the DSM segment.	-
TO	Marks the end of the segment region protected by the specified lock. Expressed in terms of a byte-offset [exclusive] from the beginning of the DSM segment.	-

Table 8-8: dsm_list Output Column Description

Output Field Name	Description	Keyword Values
SVCNO	Identifier associated with a particular instance of a service record. It is assigned by the system when the first libdsm function call is made. It is released when the request has been serviced by the system.	-
SVCTYPE	Identifies the libdsm request that is being serviced by the system.	-
TIME	Local time the corresponding service request was received by the dsmd process.	-
STAT	When no option is specified in the command, it contains information about the current status of the local copy of the DSM segment on the specified host. It can be one of the following values:	synced - Indicates that the contents of the local copy of the DSM segment on the specified host are in the process of being synchronized with other copies across the network. When the synchronization process completes successfully, the status will be marked valid. If it fails, it will be marked invalid and an attempt will be made to re-sync the contents. valid - Indicates that the contents of the local copy of the DSM segment on the specified host are in sync with other copies across the network. invalid - Indicates that the contents of the local copy of the DSM segment on the specified host are not in sync with other copies across the network.
	When \-s option is specified, this field contains information about the current status of the application request. It can be one of the following values:	init - Initial state. Indicates that a service record has just been created by the dsmd process for the request specified. pend - Pending state. Indicates that the local dsmd process is handling the request but has not yet contacted its peers on the remote hosts. wait - Wait state. The operation requested requires cooperation between multiple hosts and the system is contacting dsmd processes on remote hosts. blkd - Blocked state. Indicates that the request cannot be serviced at this time because it requires resources that are currently in use by another process.

SEE ALSO dsm_stat, dsm_getstat()

8.4.5 dsm_rm

NAME

dsm_rm Removes a DSM segment.

SYNOPSIS

dsm_rm -i id **|** ***k*** key

DESCRIPTION

dsm_rm Destroys a user-specified distributed shared memory (DSM) segment and the data structures associated with it. The *dsmd* process of all involved hosts must be running. Upon successful completion, the following will also occur:

- the identifier associated with the DSM segment will be removed from the system
- all processes attached to the segment will automatically be detached
- all locks pertaining to the segment will automatically be released

-i id Destroys the DSM segment with the specified identifier. Identifiers can be obtained with the *dsm_get()* function call.

-k key Destroys the DSM segment with the given IPC key. The IPC key value is the same value that is supplied as an argument to a *dsm_get()* function call.

NOTE: Only superuser or the owner of the DSM segment may use this command.

SEE ALSO dsm_destroy()

8.4.6 dsm_stat

NAME

dsm_stat Retrieves information about a DSM segment.

SYNOPSIS

dsm_stat [-a] -i id | ***k*** key

DESCRIPTION

dsm_stat Displays information about the individual IPC shared memory segments that make up a distributed shared memory (DSM) segment. The contents of the information retrieved is similar to the output of the UNIX **`ipcs`** command with the **`-ma`** options. The *dsmd* process of all involved hosts must be running.

-a Prints additional information about the DSM segment. This option includes the owner's process ID, the last process that modified the segment, the last attach operation that was performed, the last modify operation that was performed, and the last detach operation that was performed.

-i id Retrieves information on the DSM segment with the specified identifier. Identifiers can be obtained with the *dsm_get()* function call.

-k key Retrieves information on the DSM segment with the given IPC key. The IPC key value is the same value that is supplied as an argument to a *dsm_get()* function call.

OUTPUT VALUES

The output of this command contains several columns of information, depending on the options used in the command. The following table contains the column headings of the output and the meaning of the column.

Table 8-9: dsm_stat Output Column Description

Output Field Name	Description
DSMID	DSM segment identifier.
KEY	Key associated with the DSM segment.
HOST	Name of the host machine on which a local copy of the DSM segment exists. The DSM exists in the form of an IPC shared memory segment.
SHMID	IPC shared memory identifier associated with the local copy of the DSM segment on the specified host.
MODE	Access mode of the specified host's local copy of the DSM segment, in octal form.
SIZE	Size of the specified host's local copy of the DSM segment, in bytes.
NATTCH	Total number of processes attached to the DSM segment on the specified host.
UID	Effective user ID associated with the specified host's local copy of the DSM segment.
GID	Effective group ID associated with the specified host's local copy of the DSM segment.
CPID	UNIX Process ID of the process that created the local copy of the DSM segment on the host specified.

Table 8-9: dsm_stat Output Column Description

Output Field Name	Description
LPID	UNIX process ID of the last process that performed an attach or detach operation on the local copy of the DSM segment on the host specified.
ATIME	Local time the last attach was performed on the local copy of the DSM segment on the host specified.
CTIME	Local time the last modify operation was performed on the local copy of the DSM segment on the host specified.
DTIME	Local time the last detach was performed on the local copy of the DSM segment on the host specified.

SEE ALSO dsm_getstat()

8.5 DKM Utilities

8.5.1 dkm_apidemo

NAME

dkm_apidemo Demonstrates the capabilities of the Distributed Kernel Memory (DKM) library functions.

SYNOPSIS

dkm_apidemo

DESCRIPTION

dkm_apidemo Starts a menu-driven program designed to demonstrate and exercise the basic set of capabilities provided as part of the Distributed7 Distributed Kernel Memory (DKM) Applications Programming Interface (API) library. The correct operations of this program require the *dkmd* daemon on all involved hosts to be up and running.

This program can:

- create or destroy DKM segments
- extend or shrink an existing DKM segment
- acquire or release read-only and/or read-write locks across a DKM segment or segment extension (to be able to perform consistent read/write operations through the segment or segment extension)
- initiate or cancel asynchronous DKM lock requests
- read/write from or to a specified region of a DKM segment or segment extension
- initiate a manual request to sync the contents of a specified DKM segment or segment extension
- retrieve information about a particular DKM segment or segment extension
- register for DKM event notification capability

SEE ALSO dkmd

8.5.2 dkm_bm

NAME

dkm_bm Benchmark Distributed7 DKM framework.

SYNOPSIS

dkm_bm [-k key ***]***

DESCRIPTION

dkm_bm This utility is intended to benchmark the performance of the Distributed7 Distributed Kernel memory (DKM) framework in terms of the total number of read-only or read-write operations that can be performed within a specified time interval.

When executed, ***dkm_bm*** will prompt the user in regard to the specifics of the benchmark test to be performed (e.g., size and nature of the DKM locks to be acquired). Subsequently, ***dkm_bm*** will perform a series of non-overlapping read-only or read-write operations across a DKM segment.

After the specified number of DKM operations are completed, ***dkm_bm*** will calculate the average lock/sync/unlock times involved in performing these operations and conjecture the overall system performance in the terms of number of DKM read-only or read-write operations (of specified size) per second. The results will be displayed to the user on ***stdout***.

*Note: The **dkm_bm** utility requires **dkmd** daemons on all involved hosts to be up and running.*

-k key The value specified in this argument will be used when allocating the DKM segment necessary for conducting the benchmark tests. This option allows multiple instances of the ***dkm_bm*** utility to be executed in a concurrent, yet non-intrusive manner. By default, the ***dkm_bm*** utility uses a key value of 2000 when allocating the DKM segment. This segment is automatically destroyed upon completion of the benchmark tests.

SEE ALSO dkmd

8.5.3 dkm_dump

NAME

dkm_dump Retrieve DKM segment contents.

SYNOPSIS

dkm_dump -i id ***[-x*** extid ***] [-n] [-o*** offset ***] [-s*** size ***]***

DESCRIPTION

dkm_dump	This utility allows users to retrieve and display contents of a user-specified Distributed Kernel Memory (DKM) segment or segment extension. Information retrieved is displayed on ***stdout***. The correct operations of this program require the *dkmd* daemon on the local host to be up and running.
-i id	Retrieve contents of the DKM segment whose identifier is specified by the ***id*** argument.
-x extid	Retrieve contents of the DKM segment extension whose segment identifier is specified by the ***id*** argument and extension identifier is specified by the ***extid*** argument.
-n	Retrieve contents of the DKM segment or segment extension specified without acquiring a read-only lock. By default, ***dkm_dump*** will try to acquire (in non-blocking mode) a read-only lock of appropriate size prior to retrieving the kernel-space data associated.
-o offset	Retrieve contents of the DKM segment or segment extension specified starting from the ***offset*** specified (first ***offset*** bytes of data contained within the segment or segment extension should be skipped). By default, ***offset*** is assumed to be zero.
-s size	Retrieve and display up to the specified number of bytes. By default, ***dkm_dump*** will retrieve or display the first 4096 bytes of the DKM segment or segment extension of interest starting from the ***offset*** specified, provided that the segment or extension range specified contains this much data.

SEE ALSO dkmd, dkm_list, dkm_stat

8.5.4 dkm_list

NAME

dkm_list Display DKM related information.

SYNOPSIS

dkm_list[-d] [-h | m | l | q | s | u | x]

DESCRIPTION

dkm_list This utility can be used to retrieve and display various pieces of information about the Distributed Kernel Memory (DKM) subsystem. Without any options, the ***dkm_list*** utility displays information about the DKM segments created so far, according to data maintained by the DKM multiplexer on the local host. Otherwise, the exact nature of information to be retrieved is controlled by the command-line options. The correct operations of this program require the *dkmd* daemon on the local host to be up and running.

-d Print additional (debugging) information.

-h Print information about the DKM multiplexer on the local host.

-m Print a list of DKM segment records maintained by the DKM multiplexer on the local host. Segment records are created as a result of the *dkm_get()* function call and are maintained until a corresponding *dkm_destroy()* call is issued. This is also the default option.

-l Print a list of DKM lock records maintained by the DKM multiplexer on the local host. Lock records are created as a result of the *dkm_lock()* or *dkm_schedule()* function call and are maintained until a corresponding *dkm_unlock()* call is issued.

-q Retrieve and display information about reserved DKM STREAMS queues. These queues are used to service DKM related requests that cannot be handled directly within the calling thread.

-s Print a list of service records maintained by the DKM multiplexer on the local host. Service records are created as a result of DKM API library calls issued by kernel threads and are maintained until the corresponding request is serviced by the DKM subsystem. Service records on a specified host include not only a list of DKM related requests placed by kernel threads executing on that host, but also a list of requests placed by threads executing on remote hosts for which cooperation of the DKM multiplexer on that host is required (for DKM requests that involve coordination between individual hosts).

-u Print a list of local DKM users that are interested in being notified about M_DKM events. User records are created as a result of the *dkm_notify()*

function call and are maintained until they are explicitly cancelled by the calling thread.

-x Print a list of DKM segment extension records maintained by the DKM multiplexer on the local host. Segment extension records are created as a result of the *dkm_extend()* function call and are maintained until a corresponding *dkm_shrink()* or *dkm_destroy()* call is issued.

The column headings and the meaning of individual columns in a ***dkm_list*** listing are given below. Not all fields are common to all output formats.

- ID - DKM segment identifier.
- KEY - Key associated with the DKM segment.
- USR - A sequential number assigned by the local DKM multiplexer to the DKM users that are interested in being notified about M_DKM events.
- QUEUE - STREAMS read-side queue address.
- OTHERQUEUE - STREAMS write-side queue address.
- ADDR - Context dependent.
  When no option or ***-m*** option is specified, this field contains the starting address of the corresponding DKM segment on the local host.
  When ***-x*** option is specified, it contains the starting address of the corresponding DKM segment extension on the local host.
  When ***-l*** option is specified, it contains the beginning address of the locked region on specified host.
- SIZE - Context dependent.
  When no option or ***-m*** option is specified, this field contains the size (in bytes) of the corresponding DKM segment.
  When ***-x*** option is specified, it contains the size of the corresponding DKM segment extension.
  When ***-l*** option is specified, it contains the size of the locked region.
- THREAD - Kernel thread identifier associated with the calling thread.
- LTID - Kernel thread identifier associated with the last calling thread.
- HOST - Host associated with the calling thread.
- HOSTID - Logical host ID associated with the local host. Assumes a unique value in the [DKM_HOSTID_MIN, DKM_HOSTID_MAX] range.
- OPRMODE - Specifies the current mode of operation of the DKM multiplexer on the local host, i.e., stand-alone or distributed.
- OPRSTAT - Specifies the current operational state of the DKM multiplexer on the local host (in-service, out-of-service, not-consistent, in-transition, being-synced, syncing-peer or sync-in-prog).

The distinction between different sync states is as follows: If DKM on the local host is in the process of synchronizing its data, the DKM operational state on that machine will be shown as `being-synced` whereas the DKM operational state of all other machines, with the exception of the one with the global DKM instance, will be shown as `sync-in-prog,` meaning that some remote host is in the process of being synced. During the course of a sync operation, the operational state of the global DKM instance will always be shown as `syncing-peer.`

- PEERCNT - Total number of remote DKM multiplexers according to the local host.
- GLBINST - The host on which the global (coordinating) instance of the DKM multiplexer is located.
- LOCKID - Identifier associated with a particular instance of a lock record. Assigned by the system at the time of a *dkm_lock()* request and maintained until a corresponding *dkm_unlock()* request is placed.
- LOCKTYPE - Specifies whether the corresponding lock has been acquired for read-only or read-write purposes.
- EXTNO - DKM segment extension identifier.
- SVCNO - Identifier associated with a particular instance of a service record. Assigned by the system at the time of a DKM API library request and maintained until the corresponding request is serviced by the system.
- REQTYPE - Identifies the specifics of the DKM API library request that is in the process of being serviced by the system.
- DATE - Local date the corresponding service request has been received by the DKM multiplexer.
- TIME - Local time the corresponding service request has been received by the DKM multiplexer.
- STAT - Context dependent.
  When no option or ***-m*** option is specified, it contains information about the current status of the DKM segment on the local host and assumes a value from the following list:
  synced - Indicates that the contents of the local copy of the DKM segment are in the process of being synchronized with other copies across the network. Provided that the synchronization process completes successfully, the status will be marked as valid; otherwise, it will be marked as invalid and an attempt will be made to re-sync its contents.
  valid - Indicates that the contents of the local copy of the DKM segment specified are valid (in sync with other copies across the network).
  invalid - Indicates that the contents of the local copy of

the DKM segment specified are invalid (not in sync with other copies across the network).

When ***-x*** option is specified, it contains information about the current status of the DKM segment extension on the local host and assumes a value from the following list:

synced - Indicates that the contents of the local copy of the DKM segment extension are in the process of being synchronized with other copies across the network. Provided that the synchronization process completes successfully, the status will be marked as valid; otherwise, it will be marked as invalid and an attempt will be made to re-sync its contents.

valid - Indicates that the contents of the local copy of the DKM segment extension specified are valid (in sync with other copies across the network).

invalid - Indicates that the contents of the local copy of the DKM segment extension specified are invalid (not in sync with other copies across the network).

When ***-s*** option is specified, this field contains information about the current status of the corresponding request. It assumes a value from the following list:

init - Initial state. Indicates that a service record has just been created by the DKM multiplexer.

pend - Pending state. Indicates that the local DKM multiplexer is in the process of handling the request and has not yet contacted its peers on the remote hosts.

wait - Wait state. The operation requested requires cooperation between multiple hosts and the system is in the process of contacting DKM multiplexers on remote hosts.

blkd - Blocked state. Indicates that the request specified cannot be serviced at this time because it requires resources that are currently in use by another kernel thread.

- CNT - Displays the expected/actual number of replies received from DKM multiplexers on remote hosts in regard to a broadcast operation performed by the local DKM multiplexer earlier (while processing a DKM related request that requires coordination between individual hosts).
- CLONE - DKM multiplexer clone device number.
- USAGE - Displays the number of times the associated STREAMS queues have been in use.
- PRIM - Displays the kernel-space address of the pending DKM primitive, if any.
- MUTEX - Displays the current state of the mutual-exclusion lock associated with the STREAMS queue.

SEE ALSO dkmd, dkm_stat, dkm_dump, dkm_getlist(), dkm_gethostid()

8.5.5 dkm_rm

NAME

dkm_rm Destroy DKM segment and/or segment extension.

SYNOPSIS

dkm_rm -i id ***[-x*** extid ***] [-l]***

DESCRIPTION

dkm_rm This utility can be used to place a request to destroy a user-specified Distributed Kernel Memory (DKM) segment or segment extension. Upon successful completion, ***dkm_rm*** will also cause the identifier associated with the DKM segment or segment extension to be removed from the system. The correct operations of this program require the *dkmd* daemon on all involved hosts to be up and running.

The command-line arguments supplied to ***dkm_rm*** are used to uniquely identify the DKM segment or segment extension.

-i id Destroy DKM segment whose identifier is specified by the ***id*** argument.

-x extid Destroy DKM segment extension whose segment identifier is specified by the ***id*** argument and extension identifier is specified by the ***extid*** argument.

-l Destroy local copy of the DKM segment or segment extension specified. By default, replicated copies of the segment or segment extension on all hosts will be destroyed.

SEE ALSO dkmd, dkm_destroy(), dkm_shrink()

8.5.6 dkm_sar

NAME

dkm_sar DKM system activity reporter

SYNOPSIS

dkm_sar[-r] [-d delay ***] [-n*** count ***]***

DESCRIPTION

dkm_sar This utility can be used to activate the optional statistics collection capability that is available as part of the DKM framework. When activated, this capability will result in collection of measurement peg counts for each and every DKM related request initiated via kernel threads executing on the local host. A count of these peg counts will be displayed on ***stdout*** by the ***dkm_sar*** utility on a periodic basis for a specified number of times or until the ***dkm_sar*** utility is terminated.

The correct operations of this program require the *dkmd* daemon on all involved hosts to be up and running.

-r Reset all DKM related measurement counts to zero for a fresh start.

-d delay Wait for the specified number of seconds before retrieving and displaying DKM related measurement counts. By default, ***dkm_sar*** waits for 30 seconds before retrieving or displaying the current values of the DKM related measurement peg counts on ***stdout***.

-n count Collect the specified number of samples where each sample is separated from the other by a specified number of seconds. By default, ***dkm_sar*** collects one sample only.

The column headings and the meaning of individual columns in a ***dkm_sar*** listing are given below.

- REQTYPE - DKM request type.
- ATTEMPT - Total number of times the specified DKM function call has been invoked.
- SUCCESS - Total number of times the specified DKM function call has completed its execution successful.
- FAILURE - Total number of times the specified DKM function call has failed to execute successfully.
- INCACHE - Total number of times the specified DKM function call has been serviced successfully by the system right away, without suspending the execution of the calling kernel thread at all, using information contained on the local host (no need to consult with remote hosts).

- UNAVAIL - Total number of times the specified DKM function call has failed to execute successfully, or could not be serviced right away and its execution has been suspended, due to unavailability of resources associated. This column is applicable to *dkm_lock()* and *dkm_schedule()* functions only.
- CONSULT - Total number of times the processing of the specified DKM function call has necessitated inter-host communication with one or more DKM multiplexers located on remote host(s).

SEE ALSO dkmd, dkm_list

8.5.7 dkm_stat

NAME

dkm_stat Retrieve DKM block status information.

SYNOPSIS

dkm_stat -i id ***[-x*** extid ***]***

DESCRIPTION

dkm_stat This utility allows users to retrieve and display information about the individual data blocks comprising a Distributed Kernel Memory (DKM) segment or segment extension. Among the information displayed are the current status of the block (whether it is locked or not), number of kernel threads reading through the block, number of kernel threads waiting on the block, and the last modification time of the block.

The correct operations of this program require the *dkmd* daemon on all involved hosts to be up and running.

-i id Retrieve information about the DKM segment whose identifier is given by the ***id*** argument.

-x extid Retrieve information about the DKM segment extension whose segment identifier is given by the ***id*** argument and extension identifier is given by the ***extid*** argument.

The column headings and the meaning of individual columns in a ***dkm_stat*** listing are given below.

- BLKCNT - A sequential number assigned by the system to identify the individual blocks contained within a DKM segment or segment extension.
- STATUS - Current status of the block (unlocked or locked for read-only or read-write operations).
- FLAGS - Status flags associated with the block - refer to the **<*dkm.h*>** header file for correct interpretation of the information displayed in this field.
- READER - Number of kernel threads reading through the block specified (number of kernel threads that have currently locked the block specified for read-only or read-write operations).
- WAITER - Number of kernel threads that are waiting to lock the block (to perform a read-only or read-write operation through the block).
- MOD_DATE - Local date the contents of the corresponding data block has been modified.

- MOD_TIME - Local time the contents of the corresponding data block has been modified.
- LASTWR - Host through which the last update operation on the block has been performed.

SEE ALSO dkmd, dkm_list, dkm_dump

8.5.8 dratest

NAME

dratest (Distributed Record Access Test) application is used to exercise DRA related functionality from user space, as well as to display DRA related information.

SYNOPSIS

dratest

DESCRIPTION

dratest This application is mainly a TCL shell, extended with a set of DRA related commands which communicate with the DRA module, and a TCL script file (***dra.tcl***) which defines some TCL procedures and variables to be used along with ***dratest***.

dratest accepts all TCL commands which can be used with ***tclsh***, and all the shortcuts (i.e giving only a unique prefix of the command) available with ***tclsh*** are also available with ***dratest***. Command parameters are either simple strings or TCL Lists.

For example {0xabc1 0x150 0x6 0x8 0x30 0x00030000} (or [list 0xabc1 0x150 0x6 0x8 0x30 0x00030000]) defines a TCL List which can be used as DRA segment definition. Most of the commands update the TCL array "dra" to store lock, address and other DRA related information. This array has two indices; first one shows the DRA index of the segment, second index is a constant string showing the type of information stored, i.e., the entry ***dra(1,lock)*** is used to store lock information for the segment with index 1, whereas ***dra(0,inseq)*** is used to store the in-sequence reference value for segment with index 0. Apart from basic TCL commands, the following commands can be specified to the command interpreter (all numeric parameters are assumed to be hex numbers, the leading 0x is optional):

dra_construct " seg_name " " seg_def " " prim_def " " [sec_def] "

Construct a DRA segment. See ***dra_construct()*** for detailed parameter descriptions.

- *seg_name* is a string used for debug purposes. It is used as the name of the segment to be created. DRA does not check if this string uniquely identifies the segment.
- *seg_def* is a TCL list with entries; segment key, segment private data size, size of record distributed portion, size of record private portion, number of records in a sub-segment and segment creation flags, respectively.
  See *dra.tcl* for sample segment definitions (*seg_def* array).

- *prim_def* is the definition for segment primary index, it also is a TCL list. First entry in the list shows the type of the index, if this value is 0 primary index is sorted, if it is 1 primary index is hashed. Second and third entries in the list give the offset of the key (in record distributed portion) and size of the key. Interpretation of the remaining elements depends on the index type. For sorted indexes third entry gives the extension ratio of the index and last entry is the minimum prefix size to match a wildcard search. For hashed indexes third and fourth entries give the size of index primary and secondary area respectively. Last entry for hashed indexes is the reference value. See *dra.tcl* for sample index definitions (*sort_def* and *hash_def* variables).
- *sec_def* is the optional secondary index definition. After successful completion, ***dra_construct*** displays an index to be used as the segment reference.

dra_destroy " dra_idx "

Destroy a DRA segment. See ***dra_destroy()***.

- *dra_idx* is the index of the segment to be destroyed. Can be retrieved via a ***dra_construct*** command, or set via a ***dra_seginfo*** call.

dra_new_rec " dra_idx " " prim_key " " [sec_key] "

Create a new DRA record. See ***dra_new_rec()***.

- *dra_idx* is the index of the DRA segment.
- *prim_key* is the primary key value for the new record, and
- *sec_key* is the secondary key value (if there exists one).

After successful completion, the record pointers (distributed and private) are displayed, and record is locked (RW). Record lock information is stored in TCL variable "dra(xx,lock)". xx is the dra_idx value passed to the command. Similarly dra(xx,adr_dist), dra(xx,adr_priv), dra(xx,rec_dist) and dra(xx,rec_priv) variables hold the address of private and distributed parts, private and distributed record values, respectively.

dra_find_rec " dra_idx " " key_type " " key " " flags "

Find a DRA record. See ***dra_find_record()***.

- *dra_idx* is the index of the DRA segment.
- *key_type* is the type of the key (0:primary, 1:secondary) to perform the search.
- *key* is the key value to be searched. It is a list of hex formatted octet values.
- *flags* specifies the set of DRA/DKM flags to be used.

After successful completion, the record pointers (distributed and private) are displayed, and record is locked depending on the lock mode specified by the *flags* argument. Record lock information is stored in TCL variable "dra(xx,lock)". xx is the *dra_idx* value passed to the command. Similarly dra(xx,adr_dist), dra(xx,adr_priv), dra(xx,rec_dist) and dra(xx,rec_priv) variables hold the address of private and distributed parts, private and distributed record values, respectively.

dra_find_inseq " dra_idx " " key_type " " key " " flags " " inseq_ref "

Get in sequence the set of DRA records which match the given partial key. See ***dra_find_inseq()***.

- *dra_idx* is the index of the DRA segment.
- *key_type* is the type of the key (0:primary, 1:secondary) to perform the search.
- *key* is the partial key value to be searched. It is a list of hex formatted octet values.
- *flags* specifies the set of DRA/DKM flags to be used.
- *inseq_ref* is the reference value needed for the in-sequence search. When initiating an in-sequence search this parameter must be specified as 0. After successful completion the new reference value is stored in "dra(xx,inseq)". This value must be passed to the next ***dra_find_inseq*** call as the *inseq_ref* parameter.

The record pointers (distributed and private) are displayed, and record is locked depending on the lock mode specified by the ***flags*** argument. Record lock information is stored in TCL variable "dra(xx,lock)". xx is the *dra_idx* value passed to the command. Similarly dra(xx,adr_dist), dra(xx,adr_priv), dra(xx,rec_dist) and dra(xx,rec_priv) variables hold the address of private and distributed parts, private and distributed record values, respectively.

dra_rls_lock " dra_idx " " lock " " flags "

Release a previously locked DRA record. See ***dra_rls_lock()***.

- *dra_idx* is the index of the DRA segment.
- *lock* is the lock information for the record. A lock value retrieved via dra_find_rec, dra_find_inseq or dra_new_rec should be used (dra(xx,lock) variable).
- *flags* specifies the set of DRA/DKM flags to be used during record release operation.

dra_del_rec " dra_idx " " key_type " " key " " flags "

Delete a DRA record. See ***dra_del_record()***.

- *dra_idx* is the index of the DRA segment.
- *key_type* is the type of the key (0:primary, 1:secondary) to perform the search.
- *key* is the key value for the record to be deleted. It is a list of hex formatted octet values.

- *flags* specifies the set of DRA flags to be used for deletion.

dra_del_locked " dra_idx " " lock " " flags "

Delete a previously locked DRA record. See ***dra_del_locked()***.

- *dra_idx* is the index of the DRA segment.
- *lock* is the lock information for the record. A lock value retrieved through ***dra_find_rec***, ***dra_find_inseq*** or ***dra_new_rec*** should be used (dra(xx,lock) variable).
- *flags* specifies the set of DRA flags to be used during record delete operation.

dra_validate " lock "

Validate a previously accessed DRA record (without locking). See ***dra_validate()***.

- *lock* is the lock information for the record. A lock value retreived through dra_find_inseq via ***dra_find_rec***, ***dra_find_inseq*** or ***dra_new_rec*** should be used (dra(xx,lock) variable).

dra_relock_async " dra_idx " " lock "

Async relock (RW) request for a previously acquired DRA record.See ***dra_rls_lock()***.

- *dra_idx* is the index of the DRA segment.
- *lock* is the lock information for the record. A lock value retrieved through ***dra_find_rec***,or ***dra_find_inseq*** should be used (dra(xx,lock) variable).

dra_seglist

List of existing DRA segment instances. Segment instance pointers are displayed.

dra_seginfo " seg_ptr " " [dra_idx] "

Detailed information about a DRA segment.

- *seg_ptr* is the instance pointer for the DRA segment, can be retrieved from ***dra_seglist***.
- *dra_idx* is assigned to the DRA instance, and it can be used in DRA calls which need a segment index (***dra_new_rec***, ***dra_find_rec***, etc.).

dra_all_segs

Detailed information about all DRA segments. Also each segment is automatically associated with an index which is stored in TCL variable ***segs(seg_name)***. ***seg_name*** is the name string given to ***dra_construct.***

get_mem " addr " " size "

Retrieve the copy of a kernel buffer.

- *addr* is the beginning address of the kernel buffer.
- *size* is the size of the kernel buffer.

set_mem " addr " " val_list "

Set the contents of a kernel buffer.

- *addr* is the beginning address of the kernel buffer.
- *val_list* is the list of new values (each entry in the list represents one octet).

dra_setopts " options " " level "

Set DRA debugging options.

- *options* gives the bitmap of DRA trace options (see opts() variable settings in *dra.tcl*).
- *level* indicates the DRA trace level (see lev() variable settings in *dra.tcl*)

hexpr " args "

Evaluate arguments and format the result as a hex value.

- *args* Mathematical expression to be evaluated

FILES

$EBSHOME/access/bin/dra.tcl

EXAMPLES

```
# segment definition in variable seg_def(1), see dra.tcl
% puts stdout $seg_def(1)
0xabc1 0x150 0x6 0x8 0x30 0x00030000
# sorted index definition in variable sort_def, see dra.tcl.
% puts stdout $sort_def
0x0 0x0 0x2 0x32 0
# construct a dra segment
% dra_construct tmp_seg $seg_def(1) $sort_def
dra_idx 0x0, dkm_id 0x1
# add a record to segment with index 0
# record key : 0xaa 0xbb
% dra_new_rec 0 {aa bb}
DKM Lock : 0x0, Sub.Seg.No : 0x0, Sub.Seg.Rec : 0x2f
Priv.Lock. : TRUE, Safe.Lock. : FALSE, DKM.Rec. : 0xf5f02dac
Dist. Part (addr, size) : (0xf5f02db0, 0x6)
Priv. Part (addr, size) : (0xf5c92980, 0x8)
# display record distributed portion
% get_mem 0xf5f02db0 0x6
0xaa 0xbb 0x00 0x00 0x00 0x00
```

```
# display record private portion
% get_mem 0xf5c92980 0x8
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
# modify record distributed portion
% set_mem 0xf5f02db2 {11 22 33 44}
# re-display record distributed portion
% get_mem 0xf5f02db0 0x6
0xaa 0xbb 0x11 0x22 0x33 0x44
# release the record, dra(0,lock) variable set
# by dra_new_rec call
% dra_rls_lock 0 $dra(0,lock) $flags(none)
# issue a search to find the added record
% dra_find_rec 0 0 {aa bb} $flags(rdwr)
DKM Lock : 0x0, Sub.Seg.No : 0x0, Sub.Seg.Rec : 0x2f
Priv.Lock. : TRUE, Safe.Lock. : FALSE, DKM.Rec. : f5f02dac
Dist. Part (addr, size) : (0xf5f02db0, 0x6)
Priv. Part (addr, size) : (0xf5c92980, 0x8)
# re-display record distributed portion
% get_mem 0xf5f02db0 0x6
0xaa 0xbb 0x11 0x22 0x33 0x44
# release the record, dra(0,lock) variable set
# by dra_find_rec call
% dra_rls_lock 0 $dra(0,lock) $flags(none)
# delete the newly added record
% dra_del_rec 0 0 {aa bb} $flags(none)
# issue a search to find the deleted record
% dra_find_rec 0 0 {aa bb} $flags(rdwr)
IDX Key not found
# display a list of existing segment pointers
% dra_seglist
0xf5fa6430
# display detailed info. about the segment
% dra_seginfo 0xf5fa6430
Key : 0xabc1, Ptr : 0xf5fa6430, Id : 0x1, Name : tmp_seg
# assign the segment pointer to index 1
% dra_seginfo 0xf5fa6430 1
Key : 0xabc1, Ptr : 0xf5fa6430, Id : 0x1, Name : tmp_seg
# add a record via index 1
% dra_new_rec 1 {aa dd}
DKM Lock : 0x0, Sub.Seg.No : 0x0, Sub.Seg.Rec : 0x2f
Priv.Lock. : TRUE, Safe.Lock. : FALSE, DKM.Rec. : f5f02dac
Dist. Part (addr, size) : (0xf5f02db0, 0x8)
Priv. Part (addr, size) : (0xf5c92980, 0x8)
```



```
# release the record via index 1
% dra_rls_lock 1 $dra(1,lock) $flags(none)
# display detailed info. about all segments
% dra_all Key : 0xabc1, Ptr : 0xf5fa6430, Id : 0x1, Name : tmp_seg
# find the  via the automatically created index variable
# segs(tmp_seg), without locking
% dra_find_rec $segs(tmp_seg) 0 {aa dd} $flags(nolock)
DKM Lock : 0x0, Sub.Seg.No : 0x0, Sub.Seg.Rec : 0x2f
Priv.Lock. : FALSE, Safe.Lock. : FALSE, DKM.Rec. : f5f02dac
Dist. Part (addr, size) : (0xf5f02db0, 0x8)
Priv. Part (addr, size) : (0xf5c92980, 0x8)
```

SEE ALSO dra_new_record (), dra_del_record (), dra_del_locked (), dra_find_record (), dra_find_inseq (), dra_validate (), dra_relock_sync (), dra_relock_async (), dra_rls_lock (), dra_lock_seg_priv (), dra_rls_seg_priv (), dra_get_dkm_id (), dra_destroy (), dkm_get (), dkm_getlist ()

8.6 TCAP Utilities

8.6.1 rtc_dump

NAME

rtc_dump Retrieve RTCMOD information

SYNOPSIS

rtc_dump [-p|m|t] [-s sp] [-n ssn]

DESCRIPTION

rtc_dump This utility retrieves and displays various pieces of information about the remote Distributed7 Transaction Capabilities Application Part (TCAP) layer. Without arguments, ***rtc_dump*** displays information about all RTCMOD instances executing on the local host. Otherwise, the information retrieved is controlled by the arguments, as follows:

-p Retrieve and display contents of kernel-resident private device data maintained by the specified RTCMOD modules on the local host.

-m Retrieve and display kernel-resident statistics maintained by the specified RTCMOD modules on the local host.

-t Retrieve and display contents of kernel-resident routing tables maintained by the specified RTCMOD modules on the local host.

-s sp Operation specified is performed only by the RTCMOD modules that are associated with the signalling point number specified, and not by all RTCMOD modules on the local host—the default option.

-n ssn Operation specified is performed only by the RTCMOD modules that are associated with the SCCP subsystem number specified, and not by all RTCMOD modules on the local host—the default option.

DISPLAY FORMATS

The column headings and the meaning of individual columns in a ***rtc_dump*** listing are given below.

Not all fields are common to all output formats.

- HOST - Host associated
- MOD - The STREAMS module, i.e., RTCMOD module, and clone device number associated.
- PEERMOD - The STREAMS module and clone device number that are considered to be the peer of those listed in the MOD column.
- MUX - The STREAMS multiplexor, i.e., TCAP multiplexor, associated.

- RMTUSER - The remote user category, i.e., primary vs. secondary.
- KEY - The [assigned] IPC key associated with the STREAMS multiplexor.
- ORGKEY - The [original] IPC key associated with the STREAMS multiplexor.
- SP - The signalling point number associated.
- SSN - The SCCP subsystem number associated.

NOTES

The use of this utility should be restricted to front-end, i.e., local, machines in a front/back-end configuration because the RTCMOD module runs on front-end machines only. Refer to the ***tcm_rmtopen()*** *man page for an explanation of the front/back-end configuration.*

SEE ALSO rtc_stat, rtc_agent, tcm_rmtopen

8.6.2 rtc_stat

NAME

rtc_stat	Enable/disable RTCMOD measurements collection

SYNOPSIS

rtc_stat [-e|d] [-r] [-s sp] [-n ssn]

DESCRIPTION

rtc_stat	This utility activates or deactivates the optional measurements collection capability that is available as part of the Distributed7 remote Transaction Capabilities Application Part (TCAP) layer. These statistics are maintained by the RTCMOD module in the form measurement peg counts, and are mostly STREAMS-related statistics.
-e	Enable measurements collection by the specified RTCMOD modules on the local host.
-d	Disable measurements collection by the specified RTCMOD modules on the local host.
-r	Reset all measurement peg counts maintained by the specified RTCMOD modules on the local host.
-s sp	Operation specified is performed only by the RTCMOD modules associated with the signalling point number specified, and not by all RTCMOD modules on the local host—the default option.
-n ssn	Operation specified is performed only by the RTCMOD modules associated with the SCCP subsystem number specified, and not by all RTCMOD modules on the local host—the default option.

NOTES

The use of this utility is restricted to front-end (local) machines in a front/back-end configuration because the RTCMOD module runs on front-end machines only. Refer to the ***tcm_rmtopen()*** *man page for an explanation of the front/back-end configuration.*

SEE ALSO rtc_dump, rtc_agent, tcm_rmtopen

8.6.3 tcm_apidemo

NAME

tcm_apidemo Demonstrates the capabilities of the TCAP library functions.

SYNOPSIS

tcm_apidemo

DESCRIPTION

tcm_apidemo Starts a menu-driven program which demonstrates the basic set of capabilities provided as part of the Distributed7 TCAP Applications Programming Interface (API) library (***libtcap***). The correct operations of this program require the *tcmd* daemon on the local host to be up and running.

Using this program, one can:

- register as a TC application,
- send/receive transactions,
- choose from a variety of transaction recovery policies,
- collect transaction related statistics.

SEE ALSO tcmd, tcm_list, tcm_stat

8.6.4 tcm_list

NAME

tcm_list Display TCAP subsystem information.

SYNOPSIS

tcm_list[-d] [-p | s | t | u]

DESCRIPTION

tcm_list Used to retrieve and display various pieces of information about the Distributed7 Transaction Capabilities Application Part (TCAP) subsystem. Without any options, this utility displays information about all TC applications executing on the local host. Otherwise, the exact nature of information to be retrieved is controlled by the command-line options.

-d Prints additional [debugging] information.

-p Retrieves and displays contents of kernel-resident private device data maintained by the TCAP multiplexers on the local host.

-s Retrieves and displays kernel-resident transaction statistics maintained by the TCAP multiplexers on the local host.

-t Retrieves and displays contents of kernel-resident transaction tables maintained by the TCAP multiplexers on the local host.

-u Retrieves and displays information about the TC users executing on any host within the network.

The column headings and meanings in a ***tcm_list*** listing are given below. Not all fields are common to all output formats.

- HOST: Host associated
- MUX: The STREAMS multiplexer associated
- OWNER: Transaction owner
- TRID: Transaction ID
- PEERTRID: Peer transaction ID
- STATE: Transaction state
- DIRECTION: Direction (incoming vs. outgoing)
- TRPROTO: Underlying transport service provider (SCCPvs.TCP/IP)
- VERSION: TCAP protocol version (ANSI vs. CCITT)
- OBJECT: TC user related information
- DATE: local date the corresponding transaction has been initiated
- TIME: local time the corresponding transaction has been initiated

NOTES

Correct operation of the ***tcm_list*** program requires the *tcmd* daemon on the local host to be up and running. Furthermore, at least one TC application must be running on the host where the ***tcm_list*** program is invoked.

SEE ALSO tcmd, tcm_stat, tcm_tune

8.6.5 tcm_stat

NAME

tcm_stat Enables and disables TCAP statistics collection

SYNOPSIS

tcm_stat[-e* | *d][-r][-p physdev ***]***

DESCRIPTION

tcm_stat Is used to activate or deactivate the optional statistics data collection capability that is available as part of the Distributed7 Transaction Capabilities Application Part (TCAP) subsystem. These statistics are maintained by the TCAP layer in the form of measurement peg counts (e.g., number of TCAP messages sent/received so far, number of a specified type of TCAP message sent/received so far). Without any options, this utility retrieves and displays the current values of the measurement peg counts involved. The correct operations of this program require the *tcmd* daemon on the local host to be up and running.

-e Enable transaction statistics data collection by the TCAP multiplexers on the local host.

-d Disable transaction statistics data collection by the TCAP multiplexers the local host.

-r Reset all measurement peg counts maintained by the TCAP multiplexers on the local host.

-p physdev Operation specified should be performed only on the TCAP multiplexer whose physical device number has been specified and not on all TCAP multiplexers on the local host (which is the default option).

SEE ALSO tcmd, tcm_list, tcm_tune

8.6.6 tcm_tune

NAME

tcm_tune Tune TCAP optional parameters

SYNOPSIS

tcm_tune[-m opermode ***] [-s*** syncopt ***][-p*** physdev ***]***

DESCRIPTION

tcm_tune Is used to tune various pieces of operational parameters associated with the Distributed7 Transaction Capabilities Application Part (TCAP) subsystem. The main motivation behind tuning the TCAP operational parameters is to increase the performance of TC applications running under Distributed7 by compromising on the reliability aspects of the system. Without any options, the ***tcm_tune*** utility will display the current settings of the operational parameters associated with the TC applications executing on the local host only. The correct operations of this program require the *tcmd* daemon on the local host to be up and running.

-m oprmode Change operation mode as specified. The permissible values of ***oprmode*** are as follows: *stand-alone* or *distributed*. The default is set to *distributed* for all TC applications executing under the Distributed7 environment. When the *stand-alone* mode of operation is specified, the Distributed7 system software does not keep track of TC applications running on remote hosts and assumes that the TC application specified is running on the local host only. No attempts are being made by the system to keep the TCAP transaction layer data in sync on multiple hosts (which improves the overall performance of the TCAP subsystem significantly). When the *stand-alone* mode of operation is in use, it is not possible to recover from failures associated with TC applications running on remote hosts.

-s syncopt Change data synchronization option as specified. The permissible values of ***syncopt*** are as follows: *do-not-sync*, *sync-later* or *sync-first*. By default, ***syncopt*** is set to *do-not-sync* for all stand-alone TC applications and to *sync-later* for distributed TC applications running with a transaction recovery policy other than the "purge" policy. If and when the "purge" policy is in effect, the ***syncopt*** with be set to *do-not-sync* by default.

Under the *sync-first* option, when the TCAP transaction layer on a particular host manipulates the contents of the kernel-resident transaction table (as a result of a change in the state of a transaction), this data change will be propagated to all other hosts on which an instance of the TC application involved is executing. This is essential for implementing a variety of TCAP transaction recovery mechanisms. The default *sync-*

later option instructs the system to perform this data synchronization operation off-line (without blocking the execution of the TC application on the host on which the data change has just occurred). Note that while this brings the possibility of not being able to recover every single transaction under certain types of failures (e.g., host crashes), it improves the performance of a TC application running under a distributed environment considerably (the application need not wait for the actual data synchronization to be completed before continuing with processing of the next transaction).

-p physdev — Operation specified should be performed only on the TCAP multiplexer whose physical device number has been specified, and not on all TCAP multiplexers on the local host (this is the default option).

SEE ALSO tcmd, tcm_list, tcm_stat

8.7 ISUP Utilities

8.7.1 i_trace

NAME

i_trace Activates (or deactivates) the ISUP message trace.

SYNOPSIS

i_trace [-sp] [-pcno] [-grpnum] [-cctnum]

DESCRIPTION

i_trace This utility activates (or deactivates) the ISUP message tracing capabilities based on the user-specified CICs on the Distributed7 platform. The user specifies sp, pcno, grpnum, cctnum, and command type; then the i_trace utility sends a message, ISUP_TRACE_MSG, to the isupd process that performs the message tracing.

The i_trace utility has four types of operations:

- deactivate trace
- activate trace (display trace on console)
- activate trace (log trace to file)
- dump all states of one user-specified CIC

-sp Specifies the isupd process that performs the message tracing.

-pcno Specifies a particular CIC, when used with grpnum and cctnum.

-grpnum Specifies a particular CIC, when used with pcno and cctnum.

-cctnum Specifies a particular CIC, when used with pcno and grpnum.

DISPLAY FORMATS

The log file is located at ***$EBSHOME/access/RUN/alog***. The columns in the log file or console display are as follows:

timesstampmessage-categorymessage-type

from-modulecurrent-module

current-module-entry(exit)-state

8.8 Virtual Board Utilities

8.8.1 vb_addhost

NAME

vb_addhost Used to add a host to an established virtual board environment.

SYNOPSIS

vb_addhost hostname

DESCRIPTION

vb_addhost When a host needs to be added to the established virtual board environment, ***vb_addhost*** script is used. ***vb_addhost*** script assumes that there is a ***host_list*** file, and a new host is being added to the previously established host connections.
Since host connection is done between two hosts, ***vb_addhost*** invokes the *vb_bridge* executable for each pair of hostname and hosts in the ***host_list*** file. As an example, lets say the ***host_list*** file contains the following host list:

A B C

The ***'vb_addhost D'*** command causes below commands to be executed:

vb_bridge A D
vb_bridge B D
vb_bridge C D

As a result connection is available between each pair of host.

The *vb_startup* file is also updated, to include the executed commands.

RETURN VALUES

Error is returned on below cases:

- ***hostname*** can not be pinged.
- ***hostname*** is already connected.

SEE ALSO vb_connhosts, vb_config, vb_bridge

8.8.2 vb_bridge

NAME

vb_bridge Establish a bridge for message transmission between two hosts.

SYNOPSIS

vb_bridge host1 host2

DESCRIPTION

vb_bridge This daemon is used to establish a message transmission bridge between two hosts. This bridge must be available before the virtual board driver (*vbrd*) performs any remote operation between host1 and host2.

The virtual board does not require any physical link connection. When a link connects two ports on the same host, *vbrd* handles message transmission of this link, automatically with its internal port tables. But *vbrd* also establishes links between ports of different hosts. In this case actual message transmission between two hosts is done in user space through a pipe or bridge.

The virtual board enables remote operations transparently to the user through the ***vb_bridge*** daemon. When a physical link connection is considered, the ***vb_bridge*** program enables connecting ports on different host machines. Starting the ***vb_bridge*** daemon is the very first step of connecting ports between remote hosts. If this daemon is not running, port connection between remote hosts is not allowed.

The ***vb_bridge*** daemon, executes itself first on host1, and then on host2 and creates a pipe between the two processes. Both local and remote processes open ***/dev/vbrdu*** and send I_VBRD_CONN_HOST ioctl. In this way when *vbrd* puts a message on first process's queue, a message is automatically received from the second process on host2, and vice versa. This is guaranteed as long as the ***vb_bridge*** daemon is alive.

Since there is only one pipe between host1 and host2, all types of messages heading for the remote host use that bridge. Messages traveling on the bridge include MSUs for all ports, and *vbrd* protocol messages.

The host1 and host2 parameters must be different. If a connection between multiple hosts must be established, the ***vb_bridge*** daemon must be started for each pair of hosts.

The ***vb_bridge*** daemon stays alive until it is killed with a SIGTERM by the driver. The *vb_config* program, when invoked with ***-r*** option, kills all the ***vb_bridge*** daemons in the *vrbd* environment.

RETURN VALUES

Error is returned in the cases below:

- host1 and host2 are the same.

- There is already a connection between host1 and host2.
- Fails to open ***/dev/vbrdu***

SEE ALSO vb_connhosts, vb_config

8.8.3 vb_config

NAME

vb_config The user interface for the virtual board driver.

SYNOPSIS

vb_config [-i] [-m host1:port1 host2:port2] [-b host1:port1] [-l con|dis|all] [-h con|dis|all]

DESCRIPTION

vb_config

This program is the user interface for the virtual board driver (*vbrd*). It parses arguments and sends a request with the correct parameters to the *vbrd*.

vb_config can be initiated from within a *vbrd* shell script (*vb_connports* or *vb_discport*), or from the command line. If, however, ***vb_config*** is started from the command line, the *vb_startup* file is not updated and, as a result, does not reflect a correct snapshot of the system.

-m

Defines a link connection between two ports. Instead of cabling between the ports on physical boards, port connection is done through virtual SS7 connections. Information for each port consists of a hostname and a port number ranging from 0 to 31. There is no restriction for host names, but they must be valid system names in the same network. The scope of the port connection setup operation is not limited to the local host: A port on a local host can be connected to a port on a remote host, or visa versa.

Even if a port is local, its host information must still be supplied. If one of the ports is on a remote host, the vb_bridge daemon process must be started for the remote host. When ports on different hosts are connected, operation can succeed on one host, but the remote host may fail to connect its port. For this reason, an acknowledgment mechanism is implemented for the port connection procedure. The local host sends a port connection request to the remote host and waits for a response. The response can be positive or negative acknowledgment. If the remote host succeeds in connecting its port, it sends the originator of the port connection request a positive acknowledgment; the originator connects its port too, and the command returns. If the remote host fails to connect its port, it sends a negative acknowledgment to the originator, and the originator returns an error. Due to ***vbrd*** limitations, only one port connection can be performed at a time. Therefore, until the port connection operation is completed, i.e., a positive or negative acknowledgment is received or the acknowledgment timer expires, and the ***vb_config*** program returns, other port connection requests are rejected.

	At the end of a successful port connection operation, no error message is returned and the ***vb_config -l con*** command on the related host returns the connected ports list. If the operation fails, an error message indicating the error displays on the console. Possible error cases are: • One of the local ports specified is already connected • Another port connection operation is in progress • There is no connection to the remote host specified • Operation on the remote end failed
-b	Breaks a link connection. Only one port of the link is given as a parameter, however the other port of the link is also broken. Port information is unified with a **`hostname`** and a port number value. **`host1`** can be local or remote. This operation simulates removing cables between two ports. Therefore, the port tables are updated so that the port connection is not included, and NewNet Communication Technologies, LLC Distributed7 is informed about the new state. If a remote operation is required in the break port connection operation, and the host is not connected, ***vb_config*** returns an error message.
-h	Lists all host connection information of local *vbrd* driver. Has no parameters.
-l	Lists port information of local *vbrd* driver. Parameter can be ***con***, ***dis***, or ***all***. • con - list all ports, that are connected. • dis - list all ports, that are disconnected. • all - list all ports, regardless of state.
-r	Resets all port and host connections currently established throughout the *vbrd* environment. Has no parameters.
-s	Resets all statistics counts. Sent and received MSU counts are reset to 0 after this operation.
-t delay_time	***vbrd*** delays the delivery of alignment messages for the amount of time set. Default value of delay_time is 0 msecs.

RETURN VALUES

1 On failure.

SEE ALSO vb_connhosts, vb_startup, vb_bridge, vb_addhost, vb_connports, vb_discport, vb_reset

8.8.4 vb_connhosts

NAME

vb_connhosts -ksh script that establishes connections between each pair of hosts.

SYNOPSIS

vb_connhosts host1 host2 [host3 ...]

DESCRIPTION

vb_connhosts This is a -ksh script that updates *vb_startup* file and issues the *vb_bridge* command(s). There must be at least two hosts in the parameter list. Host names in the parameter list are pinged, to eliminate unreachable hosts. Host name duplication is also checked. When a valid host list that can be pinged is retrieved, the list is printed to host_list file. If a host can not be pinged, nothing is done for that host, as if not given as parameter.

The host connection process between multiple hosts requires starting the *vb_bridge* program between each combination of hosts. As an example, if the given command is:

vb_connhosts A B C

The ***vb_connhosts*** script, initiates the *vb_bridge* program three times with the parameters listed below:

vb_bridge A B
vb_bridge A C
vb_bridge B C

The executed *vb_bridge* commands are appended to end of the *vb_startup* file, so that the *vb_startup* file will be up-to-date.

RETURN VALUES

A warning is printed in the following cases:

- a host in parameter list is unreachable (ping failed).
- a host name is repeated in parameter list.
- a *vb_bridge* command fails (on console).

ERROR CODES

Error is returned in below cases:

- less then two parameters given

SEE ALSO vb_config, vb_bridge, vb_startup, vbrd

8.8.5 vb_connports

NAME

vb_connports Defines a link between two ports.

SYNOPSIS

vb_connports host1:port1 host2:port2

DESCRIPTION

vb_connports This is used for defining a link between two ports. It is a ksh script that updates *vb_startup* file and issues a ***vb_config -m host1:port1 host2:port2*** command.

The ***vb_connports*** script is used for defining a link between two ports. Each port information consists of a hostname and a port number (range 0-31). There is no restriction for host names, except for being a valid system name in same network. Both host1 and host2 can be local, or both can be remote or a mixture. Even if the port is local, its host information must still be supplied.

There must be at least two ports in the parameter list. Host names in the parameter list are pinged. If at least one host is unreachable, an error is returned, and the *vb_startup* file is not updated. The successfully executed ***vb_connports port1:host1 port2:host2*** command is appended to end of *vb_startup* file, so that *vb_startup* file will be up-to-date.

If one of the ports is on remote host, make sure the *vb_bridge* daemon has been started for the remote host.

The ***vb_connports*** script will not return until all remote operations are complete.

RETURN VALUES

Error is returned in below cases:

- first port is already in connected state.
- second port is already in connected state.
- a remote operation is needed, but *vb_bridge* daemon is not running.
- remote end is too late to acknowledge.

SEE ALSO vb_connhosts, vb_config, vb_bridge

8.8.6 vb_discport

NAME

vb_discport Breaks a link connection.

SYNOPSIS

vb_discport host1:port1

DESCRIPTION

vb_discport This is a ksh script that updates *vb_startup* file and issues a ***vb_config -b host1:port1*** command. It is used for breaking a link connection. Only one port of the link to be broken is given as parameter. Port information is defined with a hostname and a portnumber combination. host1 can be any host (local or remote). The other port of link is also broken (whether on local, remote or a third host).

A host name in the parameter list is pinged. If the host is unreachable, an error is returned, and the *vb_startup* file is not updated. At the end of successful operation, the ***vb_discport port1:host1*** command is appended to end of the *vb_startup* file so that the *vb_startup* file will be up-to-date.

If at least one port of the link is on a remote host, make sure the *vb_bridge* daemon has been started for the remote host, that is, you have a bridge to that host for message transmission.

RETURN VALUES

Error is returned if remote operation is required but no host connection is available (no *vb_bridge* daemon for remote host).

SEE ALSO vb_connhosts, vb_config, vb_bridge, vb_startup

8.8.7 vb_lhosts

NAME

vb_lhosts Lists host connections information for local host.

SYNOPSIS

vb_lhosts

DESCRIPTION

vb_lhosts This function is used when the user wants to list host connections information for the local host.

The output is a list of host names for which a bridge connection between that host and local host exists. Consequently, any host in the output can be used for remote operations. Before performing a remote operation, a user can use the ***vb_lhosts*** command, to see whether a message transfer path (bridge) is available to that host.

The ***vb_lhosts*** script, performs a ***vb_config -h*** command. The *vb_config* program sends I_VBRD_LIST_HOST ioctl to the driver. The driver fills in a table less than 1K block, and sends it back to the utility. It is the utility that prints the output.

RETURN VALUES

No error case.

SEE ALSO vb_connhosts, vb_config, vb_bridge

8.8.8 vb_lports

NAME

vb_lports Lists host port information on local host.

SYNOPSIS

vb_lports [dis|con|all]

DESCRIPTION

vb_lports This function is used when user wants to see port information on local host. A filter can be defined for the retrieved output.

con This option only displays the connected ports. Port information is displayed in form of:

```
Local_Port: Host: Remote_Port: sp: lset: link: state:
    rstate: lpo: rpo:
```

dis This option prints the list of idle port numbers. This information is used to see which ports can be used.

all This option prints information for all of the ports.

RETURN VALUES

No error case.

SEE ALSO vb_connhosts, vb_config, vb_bridge

8.8.9 vb_reset

NAME

vb_reset Resets port and host connections on all hosts in the virtual board environment.

SYNOPSIS

vb_reset

DESCRIPTION

vb_reset This script is used when the user needs to reset all port and host connections on all hosts in the virtual board environment.
The ***vb_reset*** script performs ***vb_config -r*** command and clears the *vb_startup* file. Resetting of remote hosts is handled by the ***vbrd*** driver.

RETURN VALUES

No error case.

SEE ALSO vb_connhosts, vb_config, vb_bridge

8.8.10 vb_startup

NAME

vb_startup Virtual board environment configuration file.

SYNOPSIS

n/a

DESCRIPTION

vb_startup This is a file that reflects current state of the virtual board environment. The ***vb_startup*** file is executed when a host machine crashes, or for other reasons the exact state of virtual link and host connection must be established.
All *vbrd* shell scripts update ***vb_startup*** file according to what they have changed. This guarantees that, if the user always uses *vbrd* shell scripts, ***vb_startup*** file will be up-to-date, and is the snapshot of the environment.

RETURN VALUES

No error case.

SEE ALSO vb_connhosts, vb_config, vb_bridge, vb_addhost, vb_connports, vb_discport, vb_reset

8.8.11 snmptest

NAME

snmptest Communicates with a network entity using SNMP Requests

SYNOPSIS

snmptest -v 1 [-p dst port] hostname community

snmptest -v 2 [-s src port] [-p dst port] hostname noAuth

snmptest -v 2 [-s src port] [-p dst port] hostname srcParty dstParty context

DESCRIPTION

snmptest This is a flexible SNMP application that can monitor and manage information on a network entity.

host Specifies either a host name or an internet address specified in "dot notation"

sourceParty/ destinationParty Specify the party names for the transaction with the remote system, as they are defined in ***/etc/party.conf***.

After invoking the program, a command line interpreter proceeds to accept commands. It will prompt with:

Variable:

At this point you can enter one or more variable names, one per line. A blank line is a command to send a request for each of the variables (in a single packet) to the remote entity. For example:

snmptest -v 1 -p 7778 localhost public

Variable: $G

Request type is Get Request

Variable: 1.1.0

Will return some information about the request and reply packets, as well as the information:

Received Get Response from 127.0.0.1

requestid 0x110C3DC6 errstat 0x0 errindex 0x0

.iso.org.dod.internet.mgmt.mib2.system.sysDescr.0 = "AccessSNMP, SNMP agent"

Upon startup, the program defaults to sending a GET Request packet. This can be changed to a GET NEXT Request or a SET Request by typing the commands "$N" or "$S" respectively. Typing "$G" will go back to the GET Request mode. The command "$D" will toggle the dumping of each sent and received packet.

When in the "SET Request" mode, more information is requested by the prompt for each variable. The prompt:

Please enter variable type [i|s|x|d|n|o|t|a]:

requests the type of the variable to be entered. Type "i" for an integer, "s" for an octet string in ascii, "x" for an octet string as hex bytes seperated by white space, "d" for an octet string as decimal bytes separated by white space, "a" for an ip address in dotted IP notation, and "o" for an object identifier. At this point a value will be prompted for:

Please enter new value:

If this is an integer value, just type the integer (in decimal). If it is a string, type in white space separated decimal numbers, one per byte of the string. Again type a blank line at the prompt for the variable name to send the packet. At the variable name line, typing "$Q" will quit the program. Adding a "-d" to the argument list will cause the application to dump input and output packets.

8.8.12 snmptrapd

NAME

snmptrapd Receives and prints SNMP traps.

SYNOPSIS

snmptrapd [-v 1] [-p port#] [-d]

DESCRIPTION

snmptrapd An SNMP application that receives SNMP traps generated by an SNMP agent. It is especially used for SNMPv1. For SNMPv2, you can use snmptest (1s), which can receive and/or generate SNMPv2 traps.

port This number must be over 1024 if the user running snmptrapd does not have superuser privileges. Default port number that is bound is 162.

d Adding this argument causes the application to dump trap packets.

SEE ALSO RFC 1155, RFC 1156, and RFC 1157 in *SNMP Security Internet Drafts*.

***Important**: A URL for the Internet-Draft is:*
ftp://ftp.ietf.org/internet-drafts/draft-ietf-snmpv3-usm-v2-01.txt

8.8.13 snmpwalk

NAME

snmpwalk Communicates with a network entity using SNMP GET Next Requests

SYNOPSIS

snmpwalk -v version -h hostname -c community -P [AC] [-p dest_port] [-s src_port]

DESCRIPTION

snmpwalk An SNMP application that uses GET NEXT Requests to query for a tree of information about a network entity.

host Specifies either a host name or an internet address specified in dot notation

-c The community string is used for authentication purposes in SNMP-V1. This is set during configuration of the AccessSNMP (SNMP agent) in the file named `access/RUNx/config/SNMP/community.conf`.

sourceParty/ destinationParty Specify the party names for the transaction with the remote system, as they are defined in ***/etc/party.conf***.

For example:

snmpwalk -v 1 -h localhost -P C -p 7778 -c public

will retrieve all the variables in the tree.

If the network entity has an error processing the request packet, an error packet will be returned and a message will be shown, helping to pinpoint why the request was malformed.

If the tree search causes attempts to search beyond the end of the MIB, a message will be displayed: *End of MIB.*

8.8.14 snmpget

NAME

snmpget Communicates with a network entity using SNMP GET Requests

SYNOPSIS

snmpget -v 1 hostname community objectID [objectID]*

snmpget -v 2 hostname noAuth objectID [objectID]*

snmpget -v 2 hostname srcParty dstParty context objectID [objectID]*

DESCRIPTION

snmpget An SNMP application that uses GET Request to query for a node of information about a network entity.

host Specifies either a host name or an internet address specified in "dot notation"

sourceParty/ destinationParty Specify the party names for the transaction with the remote system, as they are defined in ***/etc/party.conf***.

The *oid* must be given in the command line. For example:

snmpget -v 1 -p 7778 sp0.xyz.com public system.sysDescr.0

will retrieve the variable:

.iso.org.dod.internet.mgmt.mib2.system.sysDescr.0 = "AccessSNMP, SNMP agent"

If the network entity has an error processing the request packet, an error packet will be returned and a message will be shown, helping to pinpoint why the request was malformed. Adding a "-d" to the argument list will cause the application to dump input and output packets.

8.8.15 db2date

NAME

db2date Converts old database files to new database files.

SYNOPSIS

db2date <directoryname>

DESCRIPTION

db2date Utility to convert the binary database files of an old release to the current release.

directoryname Points to the access tree of the old release.

Important*: The user must have the proper permissions to read the <directoryname> and to write to $EBSHOME/access/RUN*/DBfiles directories.*

EXAMPLE

With a previous Distributed7 release located in **/usr/EBS/access_old**, and the new release located in **$EBSHOME**, convert the old database files to the new release with the following command:

```
# db2date /usr/EBS/access_old
```

All database files having actual records are converted to the new version of Distributed7, and the newly installed Distributed7 software can be run without reconfiguration.

ENVIRONMENT

$EBSHOME must be set before running this utility because it makes use of this variable to locate the database files and the Distributed7 executables.

NOTES

- The Distributed7 software must be stopped, i.e., *apm_stop*/*ebs_stop*, before running the ***db2date*** utility.
- For each signaling point—ranging from 0 to 7—one single line indicating successful conversion is displayed, e.g., **trying to convert sp=0...**
- If the current release includes any database files in it, then those files are overwritten by the ***db2date*** utility.
- In case of conversion failure, the following error is displayed for each database record:
  record insertion for MO:<MO#> failed with errno:<error#>
  In such cases, note the error and consult the Technical Assistance Center.

SEE ALSO db2text

8.8.16 db2text

NAME

db2text Converts all release NewNet Communication Technologies, LLC Distributed7 ALARM, MML, NETWORK, SPM, MTP, SCCP, and ISUP configuration database files to text files containing the MML commands that created the configuration.

Important*: The EBSHOME environment variable must be set before running this command, as it makes use of this variable to locate the database files and NewNet Communication Technologies, LLC Distributed7 executables.*

SYNOPSIS

db2text <directoryname>

DESCRIPTION

db2text Utility to convert the binary database files in the ***$EBSHOME/access/RUN<sp#>/DBfiles*** directories into text files containing the appropriate MML commands to recreate the configuration. The utility determines the current NewNet Communication Technologies, LLC Distributed7 version by checking the executables in ***$EBSHOME/access/bin***. New parameters in the MML commands of the new release—set to the defaults—can be changed by editing the text file. The text files may also contain comment lines providing information or warnings in the form of:

#<comment_line>;

The text files are stored in the directory specified by the *<directoryname>* parameter . A text file contains the commands for a particular layer/module, i.e., ALARM, MML, NET, SPM, MTP, SCCP, and ISUP, on a signalling point (0, 1, .. 7). The file names have the format of *mml_<layer>_<sp#>*.txt. For example, if MTP and SCCP were configured for signalling points 0 and 1, then the command converts the database files in directories ***$EBSHOME/access/RUN0/DBfiles*** for signalling point 0 and ***$EBSHOME/access/RUN1/DBfiles*** for signalling point 1 to files named *mml_mtp_0.txt*, *mml_sccp_0.txt*, *mml_mtp_1.txt*, and *mml_sccp_1.txt*. The text files can be edited with a text editor prior to restoring the configuration. To restore the configuration, specify the text file name with the mml command when starting MML.

<directoryname> Full path of the directory where the MML text files should be stored. The user must have write-access to the directory. The directory must not be in the ***$EBSHOME/access*** path if the previous release is removed for an upgrade.

OUTPUT FORMAT

The format of *mml_<module>_<sp#>_.txt* files are in the form of mml commands. These files may also contain some comments in the form of mml comments in order to give some warnings, or information about MO and/or parameter changes, as shown below:

#<comment_line>;

EXAMPLE

In order to convert Distributed7 DB files to text ones in the form of mml commands to the directory ***/tmp***:

db2text /tmp

We assume that there is an spc=1-2-3 and its subsystem 123 defined in the SCCP network at sp 0. So, the content of mml_sccp_0.txt may be as follows:

#sccp configuration for sp=0;

#-----------------------------------

ADD-SNSP:SPC=1-2-3;

ADD-SUBSYS:SPC=1-2-3,SSN=123;

DIRECTORIES

$EBSHOME/access/RUN<sp#>/DBfiles - manage object database files located

$EBSHOME/access/bin - Distributed7 executables are used to identify the version of Distributed7

NOTES

- Distributed7 must be stopped by calling ***ebs_stop*** before running converter.
- If there were any file named like *mml_<module>_<sp#>.txt* in the directory *<text_dir>* given as a parameter to the converter, those files will be overwritten.
- If Distributed7 is upgraded with a new version, conversion must be done before starting installation of the new Distributed7. The converter tool needs the DB files as well as Distributed7 executables located at ***$EBSHOME/access/bin*** to identify the Distributed7 version to be upgraded.
- For some old releases, converter may prompt the following: *Does Distributed7 have TCP/IP gateway deployment [no/yes]?*

Chapter 9: Man-Machine Language Commands

This chapter describes Distributed7™ Man-Machine Language (MML), the terminal handler, the rules and conventions for using MML, and the MML commands. *Chapter 6: Operations* provides information on using MML commands and a sample configuration.

MTP

Table 9-1 describes the managed objects that are used to configure the MTP user part. Details of the function and parameters of each MTP MML command are explained in Section 9.4 on page 9-37.

SCCP

Table 9-2 describes the managed objects that are used to configure the SCCP user part. Details of the function and parameters of each SCCP MML command are explained in Section 9.5 on page 9-93.

ISUP

Table 9-3 describes the managed objects that are used to configure the ISUP user part. Details of the function and parameters of each ISUP MML command are explained in Section 9.6 on page 9-110.

SYSTEM

Table 9-4 describes the managed objects that are used to configure the system. Details of the function and parameters of each system MML command are explained in Section 9.7 on page 9-128.

9.1 Terminal Handler

Distributed7 includes a Terminal Handler for interpreting MML syntax for all the layers' commands. MML conforms with the command syntax, as described in Bellcore's MML specification document.

9.2 MML Conventions

This chapter uses consistent MML conventions for labels and syntax. Each of these topics is described in the subsections that follow.

***Important**: Users must strictly adhere to the rules of command line syntax to avoid errors (see section 9.2.2).*

9.2.1 MML Network Element Labels

The MML command labels for network elements are defined as follows:

- Link set, Link, Route set, and Signaling Point name:
  12-character labels that can include any number of alphanumeric characters, numbers, and hyphens.
- Hostname:
  15-character labels that can include any number of alphanumeric characters, numbers, and hyphens.
- Point Codes (DPC, OPC and SPC):
  11-character signaling point code labels formatted as three numbers separated by hyphens (xxx-yyy-zzz), where the numbers represent the following IDs based on the standard:

ANSI	xxx is the Network ID yyy is the Cluster ID zzz is the Member ID
ITU	xxx is the Zone ID yyy is the Area ID zzz is the SP ID

ANSI and 24-bit ITU:

Fields	Network/Zone	Cluster/Area	Member/SP
Format	8 bits	8 bits	8 bits
Value Range	0 - 255	0 - 255	0 - 255

16-bit ITU:

Fields	Zone	Area	SP
Format	5 bits	4 bits	7 bits
Value Range	0 - 31	0 - 15	0 - 127

14-bit ITU:

Fields	Zone	Area	SP
Format	3 bits	8 bits	3 bits
Value Range	0 - 7	0 - 255	0 - 7

***Note**: Leading zeros (0) are not necessary.*

9.2.2 Rules for Command Line Syntax

The following rules apply to the format of command line entries:

1. Decimal values must be typed directly and should only be digits.

2. Hexadecimal values must be preceded with ***H'***.

3. Octal values must be preceded with ***O'***.

4. Parameter names can only have alphanumeric characters.

5. Commands have the form: **<Operation>-<Managed Object>**. However, commands such as HELP do not have a managed-object component.

9.2.3 String-Constant Data Entry Method

The data entry of a string constant variable is slightly different than other data entry. Since some of the constant strings are long, a utility is introduced which allows shortened versions of the constants. Users can simply type the first characters that uniquely identify that constant string instead of typing the whole string. For example, it is possible to type **DT** for *DTE*, **N** for *NATIONAL*, or **A** for *ALINK*. Note that the abbreviated part cannot be ambiguous.

MODIFY-SP:NI=INT; (VALID)

MODIFY-SP:NI=I; (VALID)

MODIFY-MTP:SLTC=O; (INVALID)

MODIFY-MTP:SLTC=OF; (VALID)

MODIFY-MTP:SLTC=ON; (VALID)

9.2.4 Case Sensitivity

MML is ***NOT*** case sensitive with respect to command and parameter names. MML converts this input into upper case. However, the values entered for a parameter are case-sensitive. The following figure illustrates the rules of case-sensitivity in MML commands.

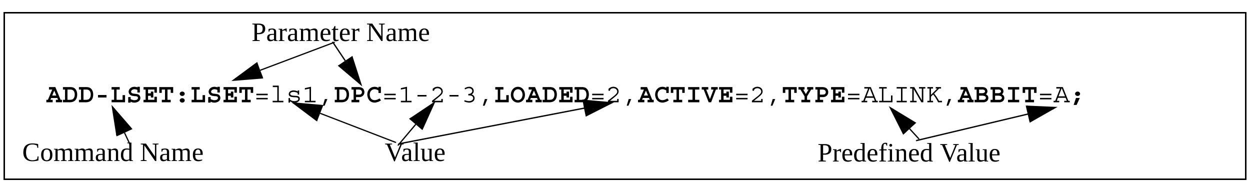


Figure 9-1: Case Sensitivity in MML Commands

- The **command name** and **parameter names** can be typed in either upper or lower case.
- The **values** ARE case sensitive.
  For example, **ADD-LSET:LSET**=ls1... and **ADD-LSET:LSET**=LS1... create two separate and unique link sets called, ls1 and LS1.

- **Predefined values** listed for a parameter, such as ALINK, must be typed in upper case.

9.2.5 Output Messages

MML prints "`<SUCCESS>`" when a command runs successfully. It prints "`<ERROR>::`*`error message`*." if a command fails. Error messages are issued by MML and by MO servers. MML performs syntactic and range checks on MML commands. MML displays an appropriate error message if a command is syntactically wrong or parameters have an out-of-range value. The following are some of the common error messages and their meanings:

1. `<ERROR>::Could not locate information for command` ***`commandname`*** `in database.`
   The MO Server does not work or MML cannot reach the database files related to this MO. Check the ***$EBSHOME*** environment variable.
2. `<ERROR>::Parameter` ***`[param]`*** `not defined.`
   The wrong parameter was entered for this MML command.
3. `<ERROR>::Missing value of` ***`[param]`*** `parameter.`
   A value should have been entered after = for this parameter.
4. `<ERROR>::Alias` ***`alias`*** `not found.`
   The parameter value, ***alias***, is considered an alias but it cannot be found in alias tables. It should be defined as a new alias.
5. `<ERROR>::Value syntax error.`
   The value of the parameter is syntactically wrong.
6. `<ERROR>::Invalid entry.`
   The command is syntactically wrong.
7. `<ERROR>::Unknown command.`
   This command is not defined. Check the command that was entered again.
8. `<ERROR>::Parameter` ***`[param]`*** `should be in range` *`n-m`*`.`
   The parameter should be in the given range.
9. `<ERROR>::Parameter` ***`[param]`*** `should be less than or equal to` *`m`* `in length.`
   Check the length of this string parameter.
10. `<ERROR>::Point code is not valid.`
    The point code you entered is not suitable for the protocol you are using, e.g., 9-1-9 is invalid for ITU.
11. `<ERROR>:: Managed Object Server not available.`
    The daemon process responsible for the command is not running, i.e., upmd or snmd. The required daemon must be started.
12. `<ERROR>::lset1 should be A and lset2 should be B PLANE`
    The link sets must be in different planes. *(For Japanese networks only.)*

13. `<ERROR>::MTP protocol is not set yet.`
The protocol of the signaling point is set by ADD-MTP command so the point code operations can only be performed after the protocol is set.

9.3 MML Tables

Table 9-1: MTP Configuration Managed Objects

Name	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
Link (Link Managed Object - see 9.4.1)	LINK	-	alphanumeric characters	String of 1 - 12 alphanumeric characters	ADD MODIFY DELETE DISPLAY
	LSET	-	alphanumeric characters	String of 1 - 12 alphanumeric characters	
	SLC	-	unsigned integer	0 - 127	
	PRIORITY	0 is the highest priority, and the highest available priority is the default.	unsigned integer	0 - 127	
	L2ECM	BASIC/PCR	-	-	
	PCRN1	-	unsigned integer	0 - 127	
	PCRN2	-	unsigned integer	0 - 9999	
	HOSTNAME	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	
	HOSTSTATUS	UNAVAILABLE/ AVAILABLE/ CONFLICT	-	-	
	BOARDNM	sbs334 pci334 pci3xpq pci3xapq cpc37xpq pmc8260 artic8260 pmc4539 vbrd adaxm	alphanumeric characters	-	
	INST	-	unsigned integer	0 - 7	
	PORT	depends upon the BOARDNM: sbs334 0 - 3 pci334 0 - 3 pci3xpq 0 - 23 pci3xapq 0 - 23 cpc37xpq 0 - 23 pmc8260 0 - 63 artic8260 0 - 63 pmc4539 0 - 3 vbrd 0 - 31	unsigned integer	-	

[1] ***Note:*** *When applicable, default values are shown in italics. Values NOT case sensitive appear in both upper and lower case.*
[2] ***Note:*** *Decimal numbers are typed without a prefix, octal numbers are preceded by O', hexadecimal numbers are preceded by H'.*

Table 9-1: MTP Configuration Managed Objects (Continued)

Name	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
LSET (Link set Managed Object - see 9.4.2)	LSET	-	alphanumeric characters	String of 1 - 12 alphanumeric characters	ADD MODIFY DELETE DISPLAY
	DPC		alphanumeric characters	Format: See Section 9.2.1 for more information about the point code format	
	TYPE	ALINK BLINK CLINK DLINK ELINK FLINK	characters	-	
	LOADED	<= value in ACTIVE	unsigned integer	1 - 128	
	ACTIVE	-	unsigned integer	1 - 128	
	ABBIT (Only used with Japan protocol)	A B	-	A - B	
	EMERGENCY	OFF - off ON - on	characters	-	

[1] ***Note:*** *When applicable, default values are shown in italics. Values NOT case sensitive appear in both upper and lower case.*
[2] ***Note:*** *Decimal numbers are typed without a prefix, octal numbers are preceded by O', hexadecimal numbers are preceded by H'.*

Table 9-1: MTP Configuration Managed Objects (Continued)

Name	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
MTP (Message Transfer Part Managed Object - see 9.4.3)	SPNO	-	unsigned integer	0 - 7	ADD MODIFY DELETE DISPLAY
	PROTOCOL	ITU_93 ITU_97 ANSI_92 ANSI_96	alphanumeric characters	-	
	VARIANT	GENERIC NEW_ZEL AT&T GTE BELL ETSI97	alphanumeric characters	-	
	PCSIZE	14_BIT 16_BIT 24_BIT	alphanumeric characters	-	
	MCONG	OFF - off ON - on	characters	-	
	MPRIO	OFF - off ON - on	characters	-	
	SLTC	OFF - off ON - on	characters	-	
	RTRC	OFF - off ON - on	characters	-	
	RESTART	OFF - off ON - on	characters	-	
	RPO2LPO	OFF - off ON - on	characters	-	
	NICHECK	OFF - off ON - on	characters	-	
	DPCCHECK	OFF - off ON - on	characters	-	
	MTP_STATE	CREATED ISOLATED RESTARTING RESTARTED	characters	-	

[1] ***Note:*** *When applicable, default values are shown in italics. Values NOT case sensitive appear in both upper and lower case.*
[2] ***Note:*** *Decimal numbers are typed without a prefix, octal numbers are preceded by O', hexadecimal numbers are preceded by H'.*

Table 9-1: MTP Configuration Managed Objects (Continued)

Name	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
RTSET (Route Set Managed Object - see 9.4.4)	RTSET	-	alphanumeric characters	String of 1 - 12 alphanumeric characters	ADD MODIFY DELETE DISPLAY
	DPC	-	alphanumeric character	Format: See Section 9.2.1 for more information about the point code format	
	RTYPE	MEMBER CLUSTER NETWORK	characters	-	
	CAPABILITY	OFF - off ON - on	characters	-	
	STATE	ACC INACC REST	characters	-	
	CONG	OFF - off ON - on	characters	-	
ROUTE (Route Managed Object - see 9.4.5)	RTSET	-	alphanumeric characters	String of 1 - 12 alphanumeric characters	ADD MODIFY DELETE DISPLAY
	LSET	-	alphanumeric characters	String of 1 - 12 alphanumeric characters	
	PRIORITY	0 is the highest priority and the highest available priority is the default if this parameter is not entered.	unsigned integer	0 - 7 (Up to two loadsharing routes can have the same priority)	
	STATE	NI RS PR	characters	-	
	LSSTATE	UA AV	characters	-	
	CURRENT	OFF - off ON - on	characters	-	
	RTCONG	OFF - off ON - on	characters	-	
	LSCONG	OFF - off ON - on	characters	-	

[1] ***Note:*** *When applicable, default values are shown in italics. Values NOT case sensitive appear in both upper and lower case.*
[2] ***Note:*** *Decimal numbers are typed without a prefix, octal numbers are preceded by O', hexadecimal numbers are preceded by H'.*

Table 9-1: MTP Configuration Managed Objects (Continued)

Name	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
SS7BOARD (SS7 Board Managed Object - see 9.4.6)	HOSTNAME	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	ADD MODIFY DELETE DISPLAY
	BOARDNM	sbs334 pci334 pci3xpq pci3xapq cpc37xpq pmc8260 artic8260 pmc4539 vbrd	alphanumeric characters	-	
	INST	-	unsigned integer	-	
	CONF	OFF - off ON - on SUSPEND - suspend RESUME - resume	characters	-	
	PM	OFF - off ON - on	characters	-	
	MODULES	trmod	alphanumeric characters	-	
	STATE	DETACHED ATTACHED DOWNLOADED READY	characters	-	
	CLASS	I II III IV	characters	-	
	PORTS	-	unsigned integer	1 - 64, where the maximum depends upon the maximum number of ports on the board.	
	LINES	-	unsigned integer	-	
	CLOCKMODE	LINE INTERNAL EXTERNAL REMOTE	characters	-	
	CLOCKSPAN	1/2/3/4/5/6/7/8	character	-	

[1] ***Note:*** *When applicable, default values are shown in italics. Values NOT case sensitive appear in both upper and lower case.*
[2] ***Note:*** *Decimal numbers are typed without a prefix, octal numbers are preceded by O', hexadecimal numbers are preceded by H'.*

Table 9-1: MTP Configuration Managed Objects (Continued)

Name	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
L2FLOW (MTP Level-2 Flow Managed Object - see 9.4.7)	LINK	-	alphanumeric characters	String of 1 - 12 alphanumeric characters	ADD MODIFY DELETE DISPLAY
	FCLEVEL	Depends on the ON/ OFF value in the MCONG parameter of the MTP MO: ON1 - 3 OFF1	integer	1 - 3	
	CONGONVAL	Depends upon the FCLEVEL value. See Table 9-10 on page 9-55 for more information.	integer	0 - 127	
	CONGABVAL	Depends upon the FCLEVEL value. See Table 9-10 on page 9-55 for more information.	integer	0 - 127	
	DISCONVAL (requires that MTP's MPRIO value be ON)	Depends upon the FCLEVEL value. See Table 9-10 on page 9-55 for more information.	integer	0 - 127	
	DISCABVAL (requires that MTP's MPRIO value be ON)	Depends upon the FCLEVEL value. See Table 9-10 on page 9-55 for more information.	integer	0 - 127	
L2TIMER (MTP Level-2 Timer Managed Object - see 9.4.8)	LINK	-	alphanumeric characters	String of 1 - 12 alphanumeric characters	MODIFY DISPLAY
	TIMER	See Integer in Table 9-11 on page 9-59.	integer	0 - 8	
	VALUE	See Range in Table 9-11 on page 9-59.	seconds or milliseconds	-	
L3TIMER (MTP Level-3 Timer Managed Object - 9.4.9)	TIMER	See Integer in Table 9-12 on page 9-61.	integer	1 - 31	MODIFY DISPLAY
	VALUE	See Range in Table 9-12 on page 9-61.	minutes or seconds	-	

[1] ***Note:*** *When applicable, default values are shown in italics. Values NOT case sensitive appear in both upper and lower case.*
[2] ***Note:*** *Decimal numbers are typed without a prefix, octal numbers are preceded by O', hexadecimal numbers are preceded by H'.*

Table 9-1: MTP Configuration Managed Objects (Continued)

Name	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
LINE (Line Managed Object - see 9.4.10)	HOSTNAME	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	MODIFY DISPLAY
	BOARDNM	sbs334 pci334 pci3xpq pci3xapq cpc37xpq pmc8260 artic8260 pmc4539	alphanumeric characters	-	
	INST	-	unsigned integer	0 - 7	
	SPAN	1,2 on class II and III boards 1/2/3/4/5/6/7/8 on class IV boards	character	-	
	LINE_FRMMOD	E1:*E1CRC4* E1FEBE E1BASIC T1:*T1ESF* T1ZBTSI T1SFRM T1SF4	alphanumeric characters	-	
	LINE_COD	E1:*E1HDB3* AMI T1:*T1B8ZS* T1B7ZS AMI	alphanumeric characters	-	
	LINE_LEN	T1:*L133* L266 L399 L533 L655 *L110* L220 L330 L440 L550 L660 LB000 LB075 LB150 LB225	alphanumeric characters	-	
	LINE_IMP	E1:*I120* I75	alphanumeric characters	-	

[1] ***Note:*** *When applicable, default values are shown in italics. Values NOT case sensitive appear in both upper and lower case.*
[2] ***Note:*** *Decimal numbers are typed without a prefix, octal numbers are preceded by O', hexadecimal numbers are preceded by H'.*

Table 9-1: MTP Configuration Managed Objects (Continued)

Name	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
LINE (Line Managed Object - see 9.4.10)	LINE_LPBK	*NONE* LOCAL REMOTE	characters	-	MODIFY DISPLAY
	LINE_NTFY	OFF - off ON - on	characters	-	
	LINE_TYP	E1: HSL/E1LSL: 2048 kbits/sec T1: HSL/T1LSL: 1544 kbits/sec	alphanumeric characters	-	
	LINE_ACCESS	FRONT=front access REAR=rear access	characters	-	
LINEHIST (Line History Managed Object - see 9.4.20)	HOSTNAME	-	alphanumeric characters	-	MODIFY DISPLAY
	BOARDNM	pci3xpq pci3xapq cpc37xpq pmc8260 artic8260 pmc4539	alphanumeric characters	-	
	INST	-	integer	0 - 7	
	SPAN	1/2/3/4/5/6/7/8	integer	-	
	INTERVAL	-	integer	-	
	RESET	YES NO	characters	-	
	ES	-	integer	-	
	UAS	-	integer	-	

[1] ***Note:*** *When applicable, default values are shown in italics. Values NOT case sensitive appear in both upper and lower case.*
[2] ***Note:*** *Decimal numbers are typed without a prefix, octal numbers are preceded by O', hexadecimal numbers are preceded by H'.*

Table 9-1: MTP Configuration Managed Objects (Continued)

Name	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
LINESTAT (Line Statistics Managed Object - see 9.4.19)	HOSTNAME	-	alphanumeric characters	-	MODIFY DISPLAY
	BOARDNM	pci3xpq pci3xapq cpc37xpq pmc8260 artic8260 pmc4539	alphanumeric characters	-	
	INST	-	integer	0 - 7	
	SPAN	1/2/3/4/5/6/7/8	integer	-	
	ERREVENTS	-	integer	-	
	CURSTATUS	SIG-AV SIG-UNAV	alphanumeric characters	-	
	CURTIMER	-	integer	-	
	CUR-ES	-	integer	-	
	CUR-UAS	-	integer	-	
	24H-ES	-	integer	-	
	24H-UAS	-	integer	-	
	VLDINTTOTAL	-	integer	-	
LINKSTAT (Link Status Managed Object - see 9.4.11)	LINK	-	alphanumeric characters	String of 1 - 12 alphanumeric characters	MODIFY DISPLAY
	STATUS	SET_ACT CLR_ACT CLR_EMR SET_EMR CLR_ECO SET_ECO CLR_INH SET_INH CLR_LPO SET_LPO TEST_SLTM	characters	-	
LSETSTAT (Link Set Status Managed Object - see 9.4.12)	LSET	-	alphanumeric characters	String of 1 - 12 alphanumeric characters	MODIFY DISPLAY
	STATUS	SET_ACT CLR_ACT	characters	-	

[1] ***Note:*** *When applicable, default values are shown in italics. Values NOT case sensitive appear in both upper and lower case.*
[2] ***Note:*** *Decimal numbers are typed without a prefix, octal numbers are preceded by O', hexadecimal numbers are preceded by H'.*

Table 9-1: MTP Configuration Managed Objects (Continued)

Name	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
PORT (Port Managed Object - see 9.4.13)	HOSTNAME	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	MODIFY DISPLAY
	BOARDNM	sbs334 pci334 pci3xpq pci3xapq cpc37xpq pmc8260 artic8260 pmc4539 vbrd	alphanumeric characters	-	
	INST	-	integer	0 - 7	
	PORTNUM	-	integer	0 - 63 (depends upon the system's configuration)	
	CLASS	I II III IV	alphanumeric characters	-	
	TYPE	DTE DCE NOTUSED	characters	-	
	BAUD	600 1200 2400 4800 7200 9600 *16000 19200 *32000 38400 *48000 *56000 *64000 *1544000 *2048000 * E1/T1 boards support only these baud rates.	integer	-	
	LPBKMODE	NONE LOCAL REMOTE	characters	-	
	IDLEDETECT	OFF - off ON - on	characters	-	

[1] ***Note:*** *When applicable, default values are shown in italics. Values NOT case sensitive appear in both upper and lower case.*
[2] ***Note:*** *Decimal numbers are typed without a prefix, octal numbers are preceded by O', hexadecimal numbers are preceded by H'.*

Table 9-1: MTP Configuration Managed Objects (Continued)

Name	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
SLTIMER (SLTM Timer Managed Object - see 9.4.14)	TIMER	1 2	integer	1 - 2	MODIFY DISPLAY
	VALUE	See Table 9-16 on page 9-74 for valid values.	integer	-	
SP (Signaling Point Managed Object - see 9.4.15)	SPNO	-	integer	0 - 7	MODIFY DISPLAY
	NAME	-	alphanumeric characaters	String of 1 - 10 alphanumeric characters	
	SPC	-	integer	Format: See Section 9.2.1 for more information about the point code format.	
	NI	INTERNATIONAL SPARE NATIONAL RESERVED	characters	-	
	TYPE	STP SEP SEPWRT	characters	-	
ALIAS (Alias Point Code Managed Object - see 9.4.16)	APC	-	integer	Format: See Section 9.2.1 for more information about the point code format	ADD MODIFY DELETE DISPLAY
	OGPC	OFF - off ON - on	characters	-	
	INFLTR	OFF SPC APC	characters	-	
	FLTRACT	ALARM UPU	characters	-	

[1] ***Note:*** *When applicable, default values are shown in italics. Values NOT case sensitive appear in both upper and lower case.*
[2] ***Note:*** *Decimal numbers are typed without a prefix, octal numbers are preceded by O', hexadecimal numbers are preceded by H'.*

Table 9-1: MTP Configuration Managed Objects (Continued)

Name	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
TIMESLOT (Time Slot Managed Object - see 9.4.17)	HOSTNAME	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	MODIFY DISPLAY
	BOARDNM	sbs334 pci334 pci3xpq pci3xapq cpc37xpq pmc8260 artic8260 vbrd	alphanumeric characters	-	
	INST	-	integer	0-7	
	DESTTYPE	LINE HDLC CTBUS	characters	-	
	DESTSPAN	LINE (II/III 1-2) (IV 1-available spans on board) CTBUS (0 -31)	integer	-	
	DESTSLOT	LINE: Class II (E1)0 -31 Class III (T1)0 - 23 HDLC: 0 - maximum ports on the board CTBUS 0 -31	integer		

[1] ***Note:*** *When applicable, default values are shown in italics. Values NOT case sensitive appear in both upper and lower case.*
[2] ***Note:*** *Decimal numbers are typed without a prefix, octal numbers are preceded by O', hexadecimal numbers are preceded by H'.*

Table 9-1: MTP Configuration Managed Objects (Continued)

Name	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
TIMESLOT (Time Slot Managed Object - see 9.4.17)	CLASS	I II III IV	alphanumeric characters	-	MODIFY DISPLAY
	ORIGTYPE	LINE HDLC NOCONNECT	characters	-	
	ORIGSPAN	LINE (II/III 1-2) (IV 1-available spans on board) CTBUS (0 -31)	integer	-	
	ORIGSLOT	LINE: Class II (E1)0 -31 Class III (T1)0 - 23 HDLC: 0 - maximum ports on the board NOCONNECT: 0	integer	-	
L2CS (MTP Level-2 Status - see 9.4.18)	LINK	-	alphanumeric characters	String of 1 - 12 alphanumeric characters	MODIFY DISPLAY
CTBUS (CTbus Managed Object- see 9.4.21 on page 9 - 88)	HOSTNAME	-	alphanumeric characters	String of 1 - 15alphanumericch aracters	MODIFY DISPLAY
	BOARDNM	sbs334pci334pci3xpqpc i3xapqcpc37xpqpmc82 60artic8260	alphanumeric characters	-	
	INST	-	unsigned integer	0-7	
	REFCLK	C8AC8BNETREF1NE TREF2SCSA2SCSA4S CSA8MVIPHMVIP	alphanumeric characters	-	
	REFINV	OFFON	characters	-	
	FBMODE	C8AC8BNETREF1NE TREF2INTERNALLIN E	alphanumeric characters	-	
	FBSPAN	-	unsigned integer	1-8	
	FB	OFFON	characters	-	

[1] ***Note:*** *When applicable, default values are shown in italics. Values NOT case sensitive appear in both upper and lower case.*
[2] ***Note:*** *Decimal numbers are typed without a prefix, octal numbers are preceded by O', hexadecimal numbers are preceded by H'.*

Table 9-1: MTP Configuration Managed Objects (Continued)

Name	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
CTBUS (CTbus Managed Object- see 9.4.21 on page 9 - 88)	COMP	OFFON	characters	-	MODIFY DISPLAY
	C8A	OFFON	characters	-	
	C8B	OFFON	characters	-	
	NRMODE	NETREF1NETREF2INTERNALLINE	alphanumeric characters	-	
	NRSPAN	-	unsigned integer	1-8	
	NR8KHZ	OFFON	characters	-	
	NRINV	OFFON	characters	-	
	NRACT	OFFON	characters	-	
	NR1	OFFON	characters	-	
	NR2	OFFON	characters	-	
	GRP_A	OFF204840968192	alphanumeric characters	-	
	GRP_B	OFF204840968192	alphanumeric characters	-	
	GRP_C	OFF204840968192	alphanumeric characters	-	
	GRP_D	OFF204840968192	alphanumeric characters	-	
	GRP_E	OFF204840968192	alphanumeric characters	-	
	GRP_F	OFF204840968192	alphanumeric characters	-	
	GRP_G	OFF204840968192	alphanumeric characters	-	
	GRP_H	OFF204840968192	alphanumeric characters	-	

[1] ***Note:*** *When applicable, default values are shown in italics. Values NOT case sensitive appear in both upper and lower case.*
[2] ***Note:*** *Decimal numbers are typed without a prefix, octal numbers are preceded by O', hexadecimal numbers are preceded by H'.*

Table 9-2: SCCP Configuration Managed Objects

Option	Parameter Name	Value	Unit	Range		Command Operation
CPC (Concerned Point Code Managed Object - see Section 9.5.1)	SPC	-	-	for CCITT networks:	Zone/ Network/ SPID (3-8-3 format)	ADD DELETE DISPLAY
				for ANSI networks:	Network/ Cluster/ Member (8-8-8 format)	
				for Japanese networks:	(5-4-7 format)	
	SSN	-	numeric	2 to 255		
	CPC	-	-	for CCITT networks:	Zone/ Network/ SPID (3-8-3 format)	
				for ANSI networks:	Network/ Cluster/ Member (8-8-8 format)	
				for Japanese networks:	(5-4-7 format)	
				for entire list:	*	
GT (Global Title Managed Object - see Section 9.5.2)	GT	-	bits	8		ADD DELETE DISPLAY MODIFY
	GTIE	-	-	1 to 15		
	TRTYPE	-	-	0 to 255		
	NUMPLAN	1	-	-		
	NATOFADDR	(replaces TRTYPE field when GTIE=1 or 4)[4]	-	-		
	ADDRINFO	-	each digit=1 byte	character string		
	LOADSHARE	ON OFF	-	character string		

Note: *Italics denotes the default*

Table 9-2: SCCP Configuration Managed Objects (Continued)

<table>
<tr><th>Option</th><th>Parameter Name</th><th>Value</th><th>Unit</th><th colspan="2">Range</th><th>Command Operation</th></tr>
<tr><td rowspan="10">GTENTRY
(Global Title Entry Managed Object - see Section 9.5.3)</td><td>IO</td><td>INCOMING
OUTGOING</td><td>-</td><td colspan="2">-</td><td rowspan="10">ADD
DELETE
DISPLAY
MODIFY</td></tr>
<tr><td>GT</td><td>-</td><td>-</td><td colspan="2">1 to 131,072</td></tr>
<tr><td>ENTRYTYPE</td><td>PRIMARY
SECONDARY</td><td>-</td><td colspan="2">-</td></tr>
<tr><td>XLATE_ID</td><td>-</td><td>Alpha Numeric characters</td><td colspan="2">1 to 12 characters</td></tr>
<tr><td rowspan="3">SPC</td><td rowspan="3">-</td><td rowspan="3">-</td><td>for CCITT networks:</td><td>Zone/ Network/ SPID (3-8-3 format)</td></tr>
<tr><td>for ANSI networks:</td><td>Network/ Cluster/ Member (8-8-8 format)</td></tr>
<tr><td>for Japanese networks:</td><td>(5-4-7 format)</td></tr>
<tr><td>SSN</td><td>-</td><td>-</td><td colspan="2">2 to 255</td></tr>
<tr><td>NEWGT</td><td>-</td><td>numeric</td><td colspan="2">1 to 4</td></tr>
<tr><td>WILDCARD</td><td>YES
NO</td><td>-</td><td colspan="2">-</td></tr>
<tr><td colspan="7">Note: Italics denotes the default</td></tr>
</table>

Table 9-2: SCCP Configuration Managed Objects (Continued)

Option	Parameter Name	Value	Unit	Range		Command Operation
MATE (Mate Managed Object - see Section 9.5.4)	SPC	-	-	for CCITT networks:	Zone/ Network/ SPID (3-8-3 format)	ADD DELETE DISPLAY
				for ANSI networks:	Network/ Cluster/ Member (8-8-8 format)	
				for Japanese networks:	(5-4-7 format)	
	SSN	-	-	2 to 255		
	MSPC	-	-	for CCITT networks:	Zone/ Network/ SPID (3-8-3 format)	
				for ANSI networks:	Network/ Cluster/ Member (8-8-8 format)	
				for Japanese networks:	(5-4-7 format)	
	MSSN	-	-	2 to 255		
Note: *Italics denotes the default*						

Table 9-2: SCCP Configuration Managed Objects (Continued)

Option	Parameter Name	Value	Unit	Range	Command Operation
SCCP (SCCP Managed Object - see Section 9.5.5)	SPNO	-	integer	0 to 7	DISPLAY MODIFY
	PROTOCOL	*DEFAULT* ANSI_92 ANSI_96 ITU_93 ITU_97	-	-	
	VARIANT	NONE ATT APLUS SNET	-	-	
	PCIND	YES NO	-	-	
	T_CONN_EST	-	decimal (in milliseconds)	-	
	T_IAS	-		-	
	T_IAR	-		-	
	T_REL	-		-	
	T_GUARD	-		-	
	T_RESET	-		-	
	T_SEGMENT	-		-	
SNSP (SCCP Signaling Point Managed Object - see Section 9.5.6)	SPC	-	-	for CCITT networks: Zone/Network/SPID (3-8-3 format)	ADD DELETE DISPLAY
				for ANSI networks: Network/Cluster/Member (8-8-8 format)	
				for Japanese networks: (5-4-7 format)	

Note: *Italics denotes the default*

Table 9-2: SCCP Configuration Managed Objects (Continued)

Option	Parameter Name	Value	Unit	Range		Command Operation
SUBSYS (Subsystem (Managed Object - see Section 9.5.7)	SPC	-	-	for CCITT networks:	Zone/ Network/ SPID (3-8-3 format)	ADD DELETE DISPLAY
				for ANSI networks:	Network/ Cluster/ Member (8-8-8 format)	
				for Japanese networks:	(5-4-7 format)	
	SSN	-	numeric	2 to 255		
LOCALSUBSYS (Local Subsystem Managed Object - see Section 9.5.8)	-	-	-	-		DISPLAY
CONNECTION (Connection Managed Object - see Section 9.5.9)	ID	-	-	0 to 16383 or ***** for all		DISPLAY
Note: *Italics denotes the default*						

Table 9-3: ISUP Configuraton Managed Objects

Option	Parameter Name	Value	Unit	Range	Command Operation
ISUPCCT (ISUP Circuits Managed Object - see Section 9.6.1)	PCNO	-	-	-	ADD DELETE DISPLAY MODIFY
	GRPID	-	integer	0 to 3039	
	CCTNUM	-	integer	0 to max # defined for this node	
	RANGE	values specified in CCTNUM[1]	-	-	
	OPERSTATE	BLO GRS HCGB HCGU MCGB MCGU RSC UBL STOP	-	-	
ISUPCGRP (ISUP Circuit Group Managed Object - see Section 9.6.2)	PCNO	-	-	-	ADD DELETE DISPLAY MODIFY
	GRPID	-	integer	0 to 3039	
	CCTNUM	-	integer	0 to max cct # defined for this node	
	TRNKGRPID	-	integer	0 to 8191	
	SCGA	ON OFF	-	-	

Note:*Italics denote the default*

[1] *Explicit value is required for group operation states, i.e., GRS, HCGB, HCGU, MCGB, and MCGU.*

[2] *The DPC parameter in the ADD command assigns a dpc to a point code number; the DPC parameter in the MODIFY command assigns a new dpc to that point code number.*

Table 9-3: ISUP Configuraton Managed Objects (Continued)

<table>
<tr><th>Option</th><th>Parameter Name</th><th>Value</th><th>Unit</th><th colspan="2">Range</th><th>Command Operation</th></tr>
<tr><td rowspan="10">ISUPNODE (ISUP Signaling Node Managed Object - see Section 9.6.3)</td><td>PCNO</td><td>-</td><td>Integer</td><td colspan="2">0 to 2047</td><td rowspan="10">ADD
DELETE
DISPLAY
MODIFY</td></tr>
<tr><td rowspan="2">DPC[2]</td><td rowspan="2">-</td><td rowspan="2">-</td><td>for CCITT:</td><td>Zone-Network-SPID</td></tr>
<tr><td>for ANSI:</td><td>Network-Cluster-Member</td></tr>
<tr><td>ANMOFF</td><td>ON
OFF</td><td>-</td><td colspan="2">-</td></tr>
<tr><td>ACMOFF</td><td>ON
OFF</td><td>-</td><td colspan="2">-</td></tr>
<tr><td>CRGOFF</td><td>ON
OFF</td><td>-</td><td colspan="2">-</td></tr>
<tr><td>CICCONTROL</td><td>ODD
EVEN
ALL
NONE
DEFAULT</td><td>-</td><td colspan="2">-</td></tr>
<tr><td>LOCATION</td><td>For ITU:
• LOCUSER
• PUBNETLOCUSER
• PRVNETREMUSER
• PRVNETLOCUSER
• TRANSNET PUBNETREMUSER
• LOCINTER
• INTERNATNET BEYINTWORKPNT
For Spain:
• LOCUSER
• PUBNETLOCUSER
• PRVNETREMUSER
• PRVNETLOCUSER
• TRANSNET PUBNETREMUSER
• LOCINTER
• INTERNATNET BEYINTWORKPNT
• PCKHNDNAT
For ANSI:
• LOCUSER
• LOCLOCNET
• PRVNETLOCUSER
• TRANSNET</td><td>Character</td><td colspan="2">-</td></tr>
<tr><td>MAXCCT</td><td></td><td>Integer</td><td colspan="2">1 to 32</td></tr>
<tr><td>FIRSTCIC</td><td></td><td>Integer</td><td colspan="2">0 to 65535</td></tr>
</table>

Note: *Italics denote the default*

[1] *Explicit value is required for group operation states, i.e., GRS, HCGB, HCGU, MCGB, and MCGU.*

[2] *The DPC parameter in the ADD command assigns a dpc to a point code number; the DPC parameter in the MODIFY command assigns a new dpc to that point code number.*

Table 9-3: ISUP Configuraton Managed Objects (Continued)

Option	Parameter Name	Value	Unit	Range	Command Operation
ISUP (ISUP Configuration Managed Object - see Section 9.6.4)	CFGNAME	CF<sp#>	-	-	DISPLAY MODIFY
	VARIANT	-	-	for ANSI: GENERIC ANSI92 ANSI96 BELL DSC MCI	
				for ITU: GENERIC AUSTRALIA BELGIUM CHILE CHI24 CZECH ETSI97 FINLAND FRANCE GERMANY HONG KONG ITALY ITU92 ITU97 MEXICO NEW_ZEALAND NORWAY PHILIPPINES Q767 RUSSIA SINGAPORE SPAIN SWEDEN SWEDENVI SWITZERLA ND THAILAND TURKEY UAE UNIPAC	
	MNTCIND	ON OFF GRPINDON	-	-	
	CONGES	ON OFF	-	-	
	RECMODE	RESCALL RELCALL			
	AUTORESP	*ON* OFF	character	-	
	EXCHODC	ON *OFF*	character	-	
	UPMIND	*ON* OFF (valid for ITU only)	character	-	

Note: *Italics denote the default*

[1] *Explicit value is required for group operation states, i.e., GRS, HCGB, HCGU, MCGB, and MCGU.*

[2] *The DPC parameter in the ADD command assigns a dpc to a point code number; the DPC parameter in the MODIFY command assigns a new dpc to that point code number.*

Table 9-3: ISUP Configuraton Managed Objects (Continued)

Option	Parameter Name	Value	Unit	Range	Command Operation
ISUPTMR (ISUP Timer Managed Object - see Section 9.6.5)	TIMERID	-	ms	1 to n	DISPLAY MODIFY
	VALUE	-	ms	10 msec to 25 hours	

Note:*Italics denote the default*
[1] *Explicit value is required for group operation states, i.e., GRS, HCGB, HCGU, MCGB, and MCGU.*
[2] *The DPC parameter in the ADD command assigns a dpc to a point code number; the DPC parameter in the MODIFY command assigns a new dpc to that point code number.*

Table 9-4: System Configuration Managed Objects

Option	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
HOST (Host Managed Object - see 9.7.1 on page 9 - 128)	HOSTNAME	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	ADD MODIFY DELETE DISPLAY
	RMTHOST	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	
	ALIAS	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	
	RMTHOSTTYP	AMGR OTHER	characters	-	
	CONF	ON - on OFF - off	characters	-	

[1] *Note: When applicable, default values are shown in italics. Values NOT case sensitive appear in both upper and lower case.*
[2] *Note: Decimal numbers are typed without a prefix, octal numbers are preceded by O', hexadecimal numbers are preceded by H'.*

Table 9-4: System Configuration Managed Objects (Continued)

Option	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
STRDALM (Stored Alarm Managed Object - see 9.7.2 on page 9 - 130)	HOSTNAME	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	DELETE DISPLAY
	GROUP	DKM ETMOD ISUP ISUPMOD MTPL1 MTPL2 APM NIMOD OMAP SCCP SPM TCAP TCMOD UPM PMON PMMOD	characters	-	
	MODULE	-	unsigned integer	Middle two digits of the alarm(s)	
	TYPE	-	unsigned integer	Last two digits of the alarm(s)	
	LAST_OCC	-	unsigned integer	Format: hh:mm:ss@ MM/DD/YY	
	FIRST_OCC	-	unsigned integer	Format: hh:mm:ss@ MM/DD/YY	
	NUM_OF_OCCUR	-	integer	-	
	ALM_TEXT	-	characters	-	

[1] *Note: When applicable, default values are shown in italics. Values NOT case sensitive appear in both upper and lower case.*
[2] *Note: Decimal numbers are typed without a prefix, octal numbers are preceded by O', hexadecimal numbers are preceded by H'.*

Table 9-4: System Configuration Managed Objects (Continued)

Option	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
ALARM (Alarm Managed Object - see 9.7.3 on page 9 - 133)	HOSTNAME	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	MODIFY DISPLAY
	DISPLAY	ON - on OFF - off	characters	-	
	CONS_THRS	INFO MINOR MAJOR CRITICAL FATAL	characters	-	
	USER_THRS	INFO MINOR MAJOR CRITICAL FATAL	characters	-	
	REPEAT	3	unsigned integer	0 - 100	
	GLOBAL	ON - on OFF - off	characters	-	
	UPDATE	ON - on OFF - off	characters	-	
ALMEVENT (Alarm Event Managed Object - see 9.7.4 on page 9 - 136)	HOSTNAME	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	DISPLAY
	REQ_HOSTNAME	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	
	GROUP	-	unsigned integer	0 - 255	
	MODULE	-	unsigned integer	0 - 255	
	TYPE	-	unsigned integer	Last two digits of the alarm(s)	
	THRESHOLD	INFO MINOR MAJOR CRITICAL FATAL	characters	-	

[1] *Note: When applicable, default values are shown in italics. Values NOT case sensitive appear in both upper and lower case.*
[2] *Note: Decimal numbers are typed without a prefix, octal numbers are preceded by O', hexadecimal numbers are preceded by H'.*

Table 9-4: System Configuration Managed Objects (Continued)

Option	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
ALMGRP (Alarm Group Managed Object - see 9.7.5 on page 9 - 138)	GROUP	DKM ETMOD ISUP ISUPMOD MTPL1 MTPL2 APM NIMOD OMAP SCCP SPM TCAP TCMOD UPM PMON PMMOD	characters	-	MODIFY DISPLAY
	CONS_THRS	INFO MINOR MAJOR CRITICAL FATAL	characters	-	
	USER_THRS	INFO MINOR MAJOR CRITICAL FATAL	characters	-	
MMLCONF (MML Configuration Managed Object- see 9.7.6 on page 9 - 140)	CONFNAME	-	alphanumeric characters	-	MODIFY DISPLAY
	LOG	ON - on OFF - off	characters	-	
	TIMEOUT	15000	milliseconds	0 - 240000	

[1] *Note: When applicable, default values are shown in italics. Values NOT case sensitive appear in both upper and lower case.*
[2] *Note: Decimal numbers are typed without a prefix, octal numbers are preceded by O', hexadecimal numbers are preceded by H'.*

Table 9-4: System Configuration Managed Objects (Continued)

Option	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
NTWK (Network Managed Object- see 9.7.7 on page 9 - 142)	HOSTNAME	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	MODIFY DISPLAY
	MODE	STNDLN DSTRBTD	characters	-	
	CLOCKSYNC	ON - on OFF - off	characters	-	
	FREQUENCY	0 - stand alone 1000 - distributed	milliseconds	60 - 10000 (for distributed mode)	
	DUALHOST	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	
	NETMASK1	`7f000000`Class A `3fff0000`Class B `1fffff00`Class C	-	32-bit mask in hex format used to extract primary network ID	
	NETMASK2	`7f000000`Class A `3fff0000`Class B `1fffff00`Class C	-	32-bit mask in hex format used to extract secondary network ID	

[1] *Note: When applicable, default values are shown in italics. Values NOT case sensitive appear in both upper and lower case.*
[2] *Note: Decimal numbers are typed without a prefix, octal numbers are preceded by O', hexadecimal numbers are preceded by H'.*

Table 9-4: System Configuration Managed Objects (Continued)

Option	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
TCPCON (TCP/IP Connections Managed Object - see 9.7.8 on page 9 - 144)	HOSTNAME	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	MODIFY DISPLAY
	RMTHOST	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	
	MODE	AUTO MASTER SLAVE	characters	-	
	SERVICE	NETDBASE	-	-	
	PROTO	TCP	-	-	
	MODULES	*NIMOD - nimod* TCMOD - tcmod	-	-	
	HBEAT	ON - on OFF - off	characters	-	
	FREQU	1000 if HBEAT is ON 0 if HBEAT is OFF	milliseconds	0 - 1000	
	MAXTRIES	-1 (unlimited tries)	integers	-1 - 5000	
	ACT_EST	IGNORE - ignore INFORM - inform	characters	-	
	ACT_RMV	IGNORE - ignore INFORM - inform	characters	-	
	HB_LOSS	NOACTION - noaction SYNCDATA - syncdata	characters	-	

[1] *Note: When applicable, default values are shown in italics. Values NOT case sensitive appear in both upper and lower case.*
[2] *Note: Decimal numbers are typed without a prefix, octal numbers are preceded by O', hexadecimal numbers are preceded by H'.*

Table 9-4: System Configuration Managed Objects (Continued)

Option	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
SET-LOG (Log Managed Object - see 9.7.11 on page 9 - 148)	TO	NMDOBJ SS7OBJ	characters	-	-
	NAME	-	characters	String of 1 to 14 characters maximum	
	SPID	-	integer	0 - 7	
	UPID	0 - MTP 3 - SCCP 5 - ISUP	integer	-	
	SSN	-	integer	2 - 255	
	INST	-	unsigned integer	-	
	LOG	ON - on OFF - off	characters	-	

[1] *Note: When applicable, default values are shown in italics. Values NOT case sensitive appear in both upper and lower case.*
[2] *Note: Decimal numbers are typed without a prefix, octal numbers are preceded by O', hexadecimal numbers are preceded by H'.*

Table 9-5: Passive Monitor Managed Objects

Managed Object	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
PMLINK(Passive Monitor Link - see 9.8.1 on page 9 - 149)	HOSTNAME	-	alphanumeric characters	String of 1-12 alphanumeric characters	ADD DELETE DISPLAY MODIFY
	BOARDNM	pci3xpq pci3xapq pmc8260 artic8260	alphanumeric characters	String of 1-12 alphanumeric characters	
	INST	-	unsigned integer	0 - 7	
	PORT	-	unsigned integer	Depends on the BOARDNUM: pci3xpq 0 - 32 pci3xapq 0 - 32 pmc8260 0 - 63 artic8260 0 - 63	
	ADMINSTAT	ACTIVATE DEACTIVATE	alphanumeric characters	-	
	OPERSTAT	SHUTOFF INACTIVE IDLE/OOS ALIGNING INSERVICE PROC-OUT	alphanumeric characters	-	
	LINKF	-	unsigned integer	-	
	RXFRAMES	-	unsigned integer	-	
	RXOCTETS	-	unsigned integer	-	
	RSU_E	-	unsigned integer	-	
	D_RXL	-	unsigned integer	-	
	D_BO	-	unsigned integer	-	

[1] *Note: When applicable, default values are shown in italics. Values NOT case sensitive appear in both upper and lower case.*
[2] Note: Decimal numbers are typed without a prefix, octal numbers are preceded by O', hexadecimal numbers are preceded by H'.

9.4 MTP MML Commands

9.4.1 Link (LINK)

NAME

LINK — Adds, modifies, deletes, or displays information about an SS7 signaling link.

COMMANDS

ADD — Adds a signaling link to an existing link set. The SS7BOARD managed object must be set to ON prior to this command. The following are created when a new LINK is added:

- an instance of LINKSTAT, which includes the status of the link.
- instances of the L2TIMER managed object (eight timers).
- instances of the L2FLOW managed object (4 flow control level).

ADD-LINK:LINK=link***,LSET***=lset***,SLC***=slc***,PRIORITY***=priority***,***
HOSTNAME=hostname***,BOARDNM***=boardnm***,INST***=inst***,***
PORT=port***[L2ECM***=l2ecm,***][PCRN1***=pcrN1,***][PCRN2***=pcrN2,***]***
[SEQUENCING=sequencing,***];***

Note*: The PROTOCOL parameter of the MTP managed object determines the instance values of L2TIMER and the number of L2FLOW instances.*

MODIFY — Modifies the parameters of an existing link.

MODIFY-LINK:LINK=link ***[,PRIORITY***=priority***]***
[,L2ECM=l2ecm***][,PCRN1***=pcrN1***][,PCRN2***=pcrN2***];***

DELETE — Deletes a link from its link set.

DELETE-LINK:LINK=link***;***

Note*: The link must be deactivated with MODIFY-LINKSTAT* ***BEFORE*** *it can be deleted. Also, the corresponding LINKSTAT instance is deleted with this command.*

DISPLAY — Displays configuration information on one link, all links in a link set, or all existing links.

DISPLAY-LINK:[LINK=link***][,LSET***=lset***];***

PARAMETERS

link — Link identification. It is a string of 1 to 12 alphanumeric characters maximum, or an * to display every link.

lset — Link set identification. It is a string of 1 to 12 alphanumeric characters maximum.

slc — Signaling link code. It is an unsigned integer from 0 to 127.

priority The priority of the signaling link. It is an unsigned integer from 0 to 127, where 0 is the highest priority. The highest available priority is assigned if no value is entered.

l2ecm MTP Level 2 Error Correction Method for the link. This parameter is optional and can be entered as one of the following values:

- **BASIC** basic method (default)
- **PCR** preventive cyclic retransmission method

Note*: The preventive cyclic retransmission method applies for intercontinental signaling links where one-way propagation delay is greater than or equal to 15ms and for all signaling links established via satellite. (ITU-T Q.703, Section 1.4: Error Correction)*

pcrN1 Maximum number of MSUs available for retransmission. ***pcrN1*** is applicable to preventive cyclic retransmission method. This parameter is optional and ignored if basic method is selected. It is an unsigned integer from 0 to 127. (Default value is 127).

pcrN2 Maximum number of MSU octets available for retransmission. ***pcrN2*** is applicable to preventive cyclic retransmission method. This parameter is optional and ignored if basic method is selected. It is an unsigned integer from 0 to 9999. (Default value is 2000).

hostname The name of the host to which the link is physically connected. It is a string of 1 to 15 alphanumeric characters maximum.

boardnm Board type, entered as one of the following values:

- **sbs334** common name for 4-port Sbus boards (sbs334/sbs37x)
- **pci334** common name for 4-port PCI bus boards (pci334/pci37x)
- **pci3xpq** common name for 24-port PCI bus boards (pci37xpq)
- **pci3xapq** common name for 24-port PCI bus boards (pci37xapq)
- **cpc3xpq** common name for 24-port CompactPCI bus boards (cpc370pq/cpc372pq)
- **pmc8260** common name for 64-port CompactPCI bus boards (pmc8260)
- **artic8260** common name for 64-port CompactPCI bus boards (artic1000 and artic2000)
- **pmc4539** common name for-128 port CompactPCI/PCI bus boards (PMC4539F)
- **vbrd** 32-port virtual board driver
- **adaxm** common name for 124 port PCIe bus boards (HDCII-LPe)

Note: *ADAX boards can support both low speed and high speed links on the same card. For the low speed links the line type of the span must be set to either E1 or T1. For the high speed links the line type of the span must be set to either E1HSL or T1HSL. In order to configure an HSL link, time slot 1 of the span with line type E1HSL or T1HSL must be* *switched to an HDLC port. The high speed link will be detected automatically when this HDLC port is configured as a link at the MTP layer.*

Note: *Although PCI3xPQ, PCI3xAPQ and CPC37xPQ boards allow configuration of up to 24 links, use of more than 16 for PCI3xPQ boards is not recommended for systems requiring full bandwidth on all configured links.*

inst Physical instance number of the board. It is an unsigned integer from 0 to 10.

port Port number of the link, entered as a numerical value. Valid range depends on board type:

- **sbs334** 0 to 3
- **pci334** 0 to 3
- **pci3xpq** 0 to 23
- **pci3xapq** 0 to 23
- **cpc3xpq** 0 to 23
- **pmc8260** 0 to 63
- **vbrd** 0 to 31
- **artic8260** 0 to 63
- **pmc4539** 0 to 3 for HSL, 0 to 127 for LSL
- **adaxm** 0 to 123

sequencing Method for keeping track of FSN/BSN values for the signalling link, set type. Valid values are:

- REGULAR—7-bit counting (only valid value for non-PMC4539 boards)
- EXTENDED—12-bit counting (*default* for PMC4539 boards)

ERRORS

<ERROR>::No room for new entry

<ERROR>::LINK MO instance already exists

<ERROR>::Missing LINK parameter

<ERROR>::Missing SLC parameter

<ERROR>::Missing PRIORITY parameter

<ERROR>::Missing HOSTNAME parameter

<ERROR>::HOSTNAME is not defined in the network

<ERROR>::Missing LSET parameter

<ERROR>::LSET MO instance does not exist

<ERROR>::Missing BOARDNM parameter

<ERROR>::Missing INST parameter

<ERROR>::Missing PORT parameter

<ERROR>::Pre-used HOSTNAME+BOARDNM+INST+PORT combination
<ERROR>::Pre-used SLC value
<ERROR>::Pre-used PRIORITY value
<ERROR>::cnfg library error
<ERROR>::Device not configured
<ERROR>::Link information can not be retrieved
<ERROR>::Invalid stream no is retrieved from spmd
<ERROR>::mtpl2 MO operation failed
<ERROR>::LINK is activated.
<ERROR>::Nothing to list.

EXAMPLES

ADD-LINK:LINK=Link_11***,LSET***=Lset_1***,SLC***=0***,PRIORITY***=0***,***
HOSTNAME=chicago***,BOARDNM=***sbs334***,INST=***0***,PORT***=0***;***

ADD-LINK:LINK=Link_14***,LSET***=Lset_1***,SLC***=0***,PRIORITY***=0***,L2ECM=***PCR***,***
HOSTNAME=chicago***,BOARDNM=***pmc4539***,INST=***0***,PORT***=0***;***

MODIFY-LINK:LINK=ls102030-01***,PRIORITY***=3***;***

MODIFY-LINK:LINK=ls102030-01***,PRIORITY***=3,***PCRN1=***120***, PCRN2=***2001***;***

DELETE-LINK:LINK=ls908070-08***;***

DISPLAY-LINK:LINK=****;***

SAMPLE OUTPUT

```
MML_TH> DISPLAY-LINK:LINK=*;
------------------------------------------------------------------------
LINKLSETSLCPRIORITYL2ECMPCRN1PCRN2HOSTNAME
------------------------------------------------------------------------
l1ls1   00    BASIC--  chicago
l2ls1   11    BASIC--  chicago
l3ls1   22    BASIC--  chicago
l4ls1   33    PCR1272000chicago


---------------------------------------------------------
HOSTSTATUSBOARDNMINSTPORTSEQUENCING
---------------------------------------------------------
AVAILABLEsbs33400REGULAR
AVAILABLEsbs33401REGULAR
AVAILABLEsbs33402REGULAR
AVAILABLEpmc453903EXTENDED
<SUCCESS>:: 4 records found
```


9.4.2 Link Set (LSET)

NAME

LSET Adds, modifies, deletes, or displays information pertaining to a link set.

COMMANDS

ADD Adds a link set. Upon adding a new LSET instance, an LSETSTAT managed object instance, which includes the status information of the link set, will also be created.

ADD-LSET:LSET=lset***,DPC***=dpc***,TYPE***=type***,LOADED***=loaded***,***
ACTIVE=active***[,ABBIT***=abbit***] [,EMERGENCY***=emergency***];***

MODIFY Modifies an existing link set.

MODIFY-LSET:LSET=lset***[,LOADED***=loaded***][,ACTIVE***=active***]***
[,TYPE=type***] [,EMERGENCY***=emergency***];***

Important*: The* ***LOADED*** *and* ***ACTIVE*** *parameters are optional but at least one of them must be entered in the MODIFY-LSET command. The LSET* ***must*** *be deactivated* ***BEFORE*** *modifying LOADED or ACTIVE parameters.*

DELETE Deletes a link set from the network.

DELETE-LSET:LSET=lset***;***

Note*: The link set* ***must*** *be deactivated with MODIFY-LSETSTAT, and the route(s) and route set for the link set* ***must*** *be deleted* ***BEFORE*** *it can be deleted.*

All LINK and LINKSTAT instances associated with this link set (if any) are deleted with this command. The LSETSTAT instance is also deleted with this command.

DISPLAY Displays the status of one link, all links in a link set, or all existing links.

DISPLAY-LINKSTAT:[LSET=lset***,][LINK***=link***];***

PARAMETERS

lset Link set identification. It is string of 1 to 12 alphanumeric characters maximum, or an ***** to display every link set.

dpc The adjacent destination point code to which this link set connects. It is entered in a X-X-X format, where the sum of the Xs must equal the bits in the PCSIZE parameter of the MTP MO:

Sample DPC	PCSIZE Parameter	Sum
3-8-3	14_BIT	3+8+3=14
5-4-7	16_BIT	5+4+7=16
8-8-8	24_BIT	8+8+8=24

type Link set type. It is a character string that accepts the following values:

- **ALINK** Access link

- **BLINK** Bridge link
- **CLINK** Cross link
- **DLINK** Diagonal link
- **ELINK** Link between a Signaling End Point (SEP)and a member of a remote Signaling Transfer Point (STP) pair
- **FLINK** Link between two Signaling End Points (SEPs)

emergency Emergency alignment of the first link in the link set. It is a character string that accepts the following values:

- **ON** Enables emergency link alignment
- **OFF** Disablles emergency link alignment (*default*)

loaded Number of links in the link set that will carry traffic, and it ***MUST*** be less than or equal to the number in ***active***. It is usually equal. It is an unsigned integer from 1 to 128.

active Number of links in the link set that are to be aligned and ready for service at all times. It is an unsigned integer from 1 to 128.

abbit AB plane value, which is used only with the Japan protocol. It is a character string that accepts the following values:

- **A** Plane A link set
- **B** Plane B link set

ERRORS

<ERROR>::Missing LSET parameter

<ERROR>::No room for new entry

<ERROR>::LSET MO instance already exists

<ERROR>::Missing DPC parameter

<ERROR>::Own point code is the same

<ERROR>::An LSET instance exists with the same Point Code

<ERROR>::Parameter LOADED is greater than parameter ACTIVE

<ERROR>::Missing TYPE parameter

<ERROR>::Missing LOADED parameter

<ERROR>::Missing ACTIVE parameter

<ERROR>::Missing ABBIT parameter

<ERROR>::LSET MO instance does not exist.

<ERROR>::LSET is being used by a LINK instance.

<ERROR>::LSET is being used by a ROUTE instance.

<ERROR>::LSET is activated.

<ERROR>::Nothing to list.

<ERROR>::ABBIT parameter is valid for japan protocol only.

EXAMPLES

ADD-LSET:LSET=Lset_1***,DPC***=2-3-4***,TYPE***=ALINK***,LOADED***=4***,ACTIVE***=4 ***[,ABBIT***=A***];***
MODIFY-LSET:LSET=ls102675***,LOADED***=4***;***
DELETE-LSET:LSET=ls000102***;***
DISPLAY-LSET:LSET=*;

SAMPLE OUTPUT

```
MML_TH> DISPLAY-LSET:LSET=*;
----------------------------------------------------------------
LSETDPC TYPE  LOADEDACTIVE ABBIT EMERGENCY
----------------------------------------------------------------
LS12-2-2ALINK 4  4  A     OFF
<SUCCESS>:: 1 records found.
```

Important*: The MTP protocol automatically activates another link when the total number of aligned links falls below the value of the ACTIVE parameter. This activation may occur because of a signaling link test (SLT) signal failure during an activation attempt, or when a link is deactivated by a management request, i.e., LINKSTAT STATUS set to CLR_ACT. The links are automatically activated in order of their priority values, with the one with the highest priority value activated first. The number of links configured in the link set must be greater than, or equal to, the number in the ACTIVE parameter of the LINKSET managed object.*

Important*: NA, for NOT APPLICABLE, is displayed in the ABBIT parameter if the protocol is not Japan.*

9.4.3 Message Transfer Part (MTP)

NAME

MTP — Adds, modifies, displays or deletes an MTP.

COMMANDS

ADD — Adds an MTP. The following are also created automatically upon adding a new MTP:

- L3TIMER with a value determined by the PROTOCOL parameter
- SLTIMER with a value determined by the PROTOCOL parameter
- SP with these parameters:
  - NAME: AMGR
  - SPC: 0-0-1
  - NI: INTERNATIONAL
  - TYPE: SEP

ADD-MTP:PROTOCOL=protocol***,PCSIZE***=pcsize***[SPNO***=spno***]
[,VARIANT***=variant***][,MCONG***=mcong***][,MPRIO***=mprio***]
[,SLTC***=sltc***][,RTRC***=rtrc***][,RPO2LPO***=rpo2lpo***]
[,NICHECK***=nicheck***] [,DPCCHECK***=dpccheck***];***

MODIFY — Modifies MTP layer 3 parameters

MODIFY-MTP:[SPNO=spno***][,VARIANT***=variant***][,MCONG***=mcong***]
[,MPRIO***=mprio***][,SLTC***=sltc***][,RTRC***=rtrc***]
[,RPO2LPO***=rpo2lpo***][,RESTART***=restart***] [,NICHECK***=nicheck***]
[,DPCCHECK***=dpccheck***];***

DELETE — Deletes an MTP instance.

DELETE-MTP:[SPNO=spno***];***

***Note**: Be sure that all related managed objects, such as LSET, LINK, RTSET, etc., are deleted when deleting an MTP.*

DISPLAY — Displays variant and congestion information for the Message Transfer Part (MTP) layer of the signaling point.

DISPLAY-MTP:[SPNO=spno***];***

PARAMETERS

***Note**: Unless otherwise specified, the default value is used if an optional parameter is not entered.*

spno — Signaling Point number entered as a numerical value from 0 to 7 or an * to display every number. If this optional parameter is not entered, the the number of the SP that this command is started with is used.

protocol	Protocol of the MTP layer entered with one of the following values: • **ITU_93** ITU 1993 specifications • **ITU_97** ITU 1997 specifications • **ANSI_92** ANSI 1992 specifications • **ANSI_96** ANSI 1996 specifications
variant	Variant of the MTP layer entered with one of the following values: • **GENERIC** No variant set *(default)* • **BELL** BELL variant (valid for ANSI) • **NEW_ZEL** New Zealand variant (reserved for now) • **AT&T** AT&T variant (valid only for ANSI92) • **GTE** GTE variant (reserved for now) • **ETSI97** ETSI 1997 variant (valid for ITU97)
pcsize	Point code size of the MTP layer. It is a character string that accepts the following values: • **14_BIT** 14 bit point code • **16_BIT** 16 bit point code • **24_BIT** 24 bit point code
mcong	Multiple Congestion flag. It is a character string that accepts the following values: • **OFF** this function *is not* operational *(default)* • **ON** this function *is* operational
mprio	Multiple Priority flag. It is a character string that accepts the following values: • **OFF** this function *is not* operational *(default)* • **ON** this function *is* operational
sltc	Signaling Link Test Control flag. It is a character string that accepts the following values: • **OFF** this function *is not* operational *(default)* • **ON** this function *is* operational
rtrc	Transfer Restricted Control flag. It is a character string that accepts the following values: • **OFF** this function *is not* operational *(default)* • **ON** this function *is* operational
restart	MTP Restart flag, which initiates a manual MTP restart operation. It is a character string that accepts the following values: • **OFF** this function *is not* operational *(default)* • **ON** this function *is* operational

rpo2lpo Remote and Local Processor Outage flag ... It is a character string that accepts the following values:

- **OFF** this function *is not* operational *(default)*
- **ON** this function *is* operational

nicheck Optional Network Indicator check flag for incoming messages. It is a character string that accepts the following values:

- **ON** (*default*)
- **OFF**

dpccheck Optional Destination Point Code check flag for incoming user part messages. This check applies to SEPs only, and not to STPs. It is a character string that accepts the following values:

- **ON** (*default*)
- **OFF**

Important*: If **PROTOCOL** is set to ANSI_92 or ANSI_96 then the **PCSIZE** value must be 24_BIT, the **MCONG** value must beo ON, **MPRIO** value must ON, **RTRC** value must ON and **RPO2LPO** value must be OFF.*

ERRORS

<ERROR>::MTP MO instance already exists

<ERROR>::Missing PROTOCOL parameter

<ERROR>::Missing PCSIZE parameter

<ERROR>::RESTART parameter is not allowed in add operation

<ERROR>::Parameters are incompatible with ansi protocols

<ERROR>::MTP MO instance does not exits

<ERROR>::PROTOCOL parameter can not be modified

<ERROR>::PCSIZE parameter can not be modified

<ERROR>::At least one routeset is in use

<ERROR>::Nothing to list

<ERROR>::Invalid SPNO for this upmd MOS

<ERROR>::Mgmt, MTP_STATE RESTARTING

<ERROR>::Mgmt, MTP_STATE RESTARTED

<ERROR>::Invalid VARIANT for this protocol.

EXAMPLES

ADD-MTP:PROTOCOL=ITU***,PCSIZE***=14_BIT***,MCONG***=OFF***,MPRIO***=OFF***, SLTC***=OFF***;***
MODIFY-MTP:SLTC=ON***,MPRIO***=OFF***;***
DELETE-MTP:;
DISPLAY-MTP:;

SAMPLE OUTPUT

```
MML_TH> DISPLAY-MTP:;
---------------------------------------------------------------------------------------
SPNO PROTOCOL  VARIANT PCSIZE M CONG  MPRIO SLTC MTP_STATE  RTRC  RPO2LPO NICHECK DPCCHECK
---------------------------------------------------------------------------------------
0    ANSI_92    GENERIC  24_BIT   ON     ON     OFF  CREATED    ON     OFF      ON       ON
<SUCCESS>:: 1 record found.
```

Table 9-6: MTP Display Values

MTP_STATE
CREATED ISOLATED RESTARTING RESTARTED

9.4.4 Route Set (RTSET)

NAME

RTSET Adds, deletes, displays, or modifies a route set, or information about a route set

COMMANDS

ADD Adds a route set.

ADD-RTSET:RTSET=rtset,DPC=dpc[,RTYPE=rtype][,CAPABILITY=capability];

MODIFY Modifies a route's STATE if CAPABILITY is set to ON. This command is functional only for GATEWAY processes.

MODIFY-RTSET:RTSET=rtset***,STATE***=state***;***

DELETE Deletes a route set from the network.

DELETE-RTSET:RTSET=rtset***;***

***Important**: Be sure that none of the existing SS7 components, such as ISUPNODE, SNSP, etc., use the destination point code (dpc) defined by this route set **BEFORE** deleting a routeset.*

DISPLAY Displays information about one or more route sets.

DISPLAY-RTSET:[RTSET=rtset***];***

PARAMETERS

rtset Route set name. It is a string of 1 to 12 alphanumeric characters maximum, or an * to display all route sets.

dpc Destination point code for this route set. It is entered in a X-X-X format, where the sum of the Xs must equal the bits in the PCSIZE parameter of the MTP MO:

Sample DPC	PCSIZE Parameter	Sum
3-8-3	14_BIT	3+8+3=14
5-4-7	16_BIT	5+4+7=16
8-8-8	24_BIT	8+8+8=24

Important*: The 0-0-2 point code is reserved for internal use. Users* ***MUST NOT*** *use this point code to create a route set.*

rtype Routing type of the route set. This parameter must be set to MEMBER in ITU networks. Valid values are:

- **MEMBER**Routing based on the full point code. (default)
- **CLUSTER**Routing based on the network and cluster portions of the point code.
- **NETWORK**Routing based on the network portion of the point code.

CAPABILITY: Whether the rtset supports capability routing attribute.

- **ON** Capability routing attribute is set for gateway application.
- **OFF** Capability routing attribute is not set.

state State of a CAPABILITY route set. It is a character string that accepts the following values:

- **ACC** activates the capability route set
- **INACC** deactivates the capability route set *(default)*
- **RESTR** changes a CAPABILITY route set to a restricted state.

ERRORS

<ERROR>::Missing RTSET parameter

<ERROR>::No room for new entry

<ERROR>::RTSET MO instance already exists

<ERROR>::Missing DPC parameter

<ERROR>::Own point code is the same

<ERROR>::A routeset instance exists with the same Point Code

<ERROR>::Only member routing is valid for ITU protocols

<ERROR>::RTSET MO instance does not exist.

<ERROR>::Instance is marked by another MOS.

<ERROR>::Nothing to list.

EXAMPLES

ADD-RTSET:RTSET=denver***,DPC=***2-32-0***,RTYPE=***CLUSTER***;***
ADD-RTSET:RTSET=takoma***,DPC=***2-32-6***;***

DELETE-RTSET:RTSET=philpa13***;***
DISPLAY-RTSET:RTSET=*;

SAMPLE OUTPUT

```
MML_TH> DISPLAY-RTSET:RTSET=*;
-------------------------------------------------
RTSETDPCTYPECAPABILITYSTATECONG
-------------------------------------------------
rs12-3-4CLUSTEROFFINACCOFF
rs2240-23-2MEMBERONACCOFF
<SUCCESS>:: 2 records found.
```

Table 9-7: RTSET Display Values

CONG	STATE
Congestion Status: • ON • OFF	State of Routeset: • ACC Accessible • INACC Inaccessible • REST Restricted

9.4.5 Route

NAME

ROUTE Adds, deletes, or displays a route, or information about a route.

COMMANDS

ADD Adds a route to a route set.

ADD-ROUTE:RTSET=rtset***,LSET***=lset***,[PRIORITY***=priority***];***

DELETE Deletes a route (link set) from an existing route set.

DELETE-ROUTE:RTSET=rtset***,LSET***=lset***;***

DISPLAY Displays information about one or more routes in a route set.

DISPLAY-ROUTE:[RTSET=rtset***][,LSET***=lset***];***

PARAMETERS

rtset: Route set name. It is a string of 1 to 12 alphanumeric characters maximum, or an * to display all route sets.

lset: Link set identification. It is a string of 1 to 12 alphanumeric characters maximum, or an * to display all route sets.

priority	Priority of the link set. It is an unsigned integer from 0 to 7, where 0 is the highest priority. The first available priority value is assigned if no priority is entered. The maximum number of routes that can have the same priority in a route set is two. These are load-sharing routes.

ERRORS

<ERROR>::Missing RTSET parameter

<ERROR>::Missing LSET parameter

<ERROR>::ROUTE MO instance already exists

<ERROR>::RTSET MO instance does not exist

<ERROR>::No room for new ROUTE entry in this RTSET

<ERROR>::No more room for equal priority routes

<ERROR>::LSET MO instance does not exist

<ERROR>::ROUTE MO instance does not exist.

<ERROR>::Nothing to list.

EXAMPLES

ADD-ROUTE:RTSET=chicago***,LSET***=chicago_1***;***
ADD-ROUTE:RTSET=chicago***,LSET***=chicago_2***,PRIORITY***=2***;***
DELETE-ROUTE:RTSET=trumct00cl ***3,LSET***=Is111111***;***
DISPLAY-ROUTE:RTSET=rs1***,LSET***=****;***

SAMPLE OUTPUT

```
MML_TH> DISPLAY-ROUTE:RTSET=rs1;
-----------------------------------------------------------------------
RTSET   LSET  PRIORITY  STATE   LSSTATE  CURRENT  RTCONG  LSCONG
-----------------------------------------------------------------------
  rs1    ls1         0     NI        UA       OFF     OFF    OFF
<SUCCESS>:: 1 records found.
```

Table 9-8: ROUTE Display Values

STATE	LSSTATE	CURRENT	RTCONG	LSCONG
State of route: • PRprohibited • NINot initiated • RSrestricted	State of route (link set). • Aavailable • UAunavailable	Availability of link set as a route to destination: • OFF • ON	Congestion of route: • OFF • ON	Congestion of link set: • OFF • ON

9.4.6 SS7 Board (SS7BOARD)

NAME

SS7BOARD Adds, modifies, deletes, or displays an SS7 board and information about its configuration in the system.

COMMANDS

ADD Adds and configures a board in the system.

ADD-SS7BOARD:HOSTNAME=hostname***,BOARDNM***=boardnm***,***
INST=inst***[,PORTS***=ports***][,MODULES***=modules***]***
[,CLOCKMODE=clockmode***][,CLOCKSPAN***=clockspan***]***
[,CONF=conf***][,PM***=pm***];***

MODIFY Modifies the configuration settings of a board that is defined.

MODIFY-SS7BOARD:HOSTNAME=hostname***,***
BOARDNM=boardnm***,INST***=inst***[MODULES***=modules***]***
[,CLOCKMODE=clockmode***][,CLOCKSPAN***=clockspan***]***
[,CONF=conf***];***

DELETE Deletes an SS7 board from the system.

DELETE-SS7BOARD:HOSTNAME=hostname***,***
BOARDNM=boardnm***,INST***=inst***;***

DISPLAY Displays the configuration information of the SS7 boards in the system.

DISPLAY-SS7BOARD:[HOSTNAME=hostname***]***
[,BOARDNM=boardnm***][,INST***=inst***];***

PARAMETERS

hostname Name of the host. It is a string of 1 to 15 alphanumeric characters maximum.

boardnm Board type, entered as one of the following values:

- **sbs334** common name for 4-port Sbus boards (sbs334/sbs37x)
- **pci334** common name for 4-port PCI boards (pci334/pci37x)
- **pci3xpq** common name for 24-port PCI boards (pci370pq/372pq)
- **pci3xapq** common name for 24-port PCI boards (pci370apq/372apq)
- **cpc3xpq** common name for 24-port CompactPCI bus boards (cpc370pq/cpc372pq)
- **pmc8260** common name for 64-port CompactPCI bus boards (pmc8260)
- **artic8260** common name for 64-port CompactPCI bus boards (artic1000 and artic2000)

- **pmc4539** common name for 128 port CompactPCI/PCI bus boards (PMC4539F)
- **vbrd** 32-port virtual board driver (vbrd)
- **adaxm** common name for 124 port PCIe bus boards (HDCII-LPe)

Note: *Although PCI3xPQ, PCI3xAPQ and CPC3xPQ boards allow configuration of up to 24 links, use of more than 16 for PCI3xPQ boards is not recommended for systems requiring full bandwidth on all configured links.*

inst Identifies the SS7 board driver instance number. The *getcfg* command provides the configured SS7 board slot, driver, and instance information.

pm passive monitoring option. it is used to select the software layer that will be downloaded on to the board. Set type, possible values are:

- **ON**: download Passive Monitor software
- **OFF**: download MTPL2 protocol software *(default)*

conf Determines the configuration of the host. It is a character string that accepts the following values:

- **ON**: establish this connection
- **OFF**: wait for some other configurations *(default)*
- **SUSPEND**: suspend board instance for CompactPCI hot-swap operation
- **RESUME**: resume board instance after CompactPCI hot-swap operation

modules Specifies an ordered list of STREAMS modules to be pushed over this device connection. ***trmod*** is the default module name.

ports Maximum number of available ports on this board. The default value is determined by the physical board hardware. The value ranges from 1 to 32, for different types of boards, where the maximum is the maximum number of ports on the board.

clock Clock source of board—class II, class III or class IV type. It is a character string that accepts the following values:

- **LINE** (for class II, III and IV boards LINE-1 (default) maps to the older LINEB and LINE-2 maps to the older LINEA)
- **INTERNAL** (for class II, III and IV boards)
- **EXTERNAL** (for class II and III boards)
- **REMOTE (**for Class IV boards, designates that the board is a CT bus clock slave, in all other clockmodes (except EXTERNAL) the board is a CT bus clock master with its internal clock synchronized to either the LINE 1 to 8 or self-synchronized in case of INTERNAL)

clockspan Clock source span number if the clock parameter is LINE. For class II and III boards ***clockspan*** is in the range 1 to 2. For class IV boards its range is 1 to the number of available physical spans on the board. Ignored for clock sources other than LINE.

ERRORS

<ERROR>::Missing BOARDNM attribute

<ERROR>::Missing INST attribute

<ERROR>::Board already configured

<ERROR>::Board is in use

<ERROR>::No such a SS7BOARD MO instance

<ERROR>::Can not add board MO

<ERROR>::Can not get board attributes

<ERROR>::CLOCKMODE value must be one of {LINE, INTERNAL, EXTERNAL}

<ERROR>::PORTS value is greater than number of available ports on the board

<ERROR>::Board is OFFLINE

EXAMPLES

ADD-SS7BOARD:HOSTNAME=host-A***,BOARDNM=***sbs334***,INST=***0***,***
MODULES=trmod***,PORTS=***4***,CLOCKMODE=***LINE***,CLOCKSPAN=***1***,CONF=***ON***;***
MODIFY-SS7BOARD:HOSTNAME=host-A***,BOARDNM=***sbs334***,INST=***0***,***
MODULES=trmod***,CLOCKMODE=***LINE***,CLOCKSPAN=***1***,CONF=***ON***;***
DELETE-SS7BOARD:HOSTNAME=ultra5***,BOARDNM=***pci334***,INST=***0***;***
DISPLAY-SS7BOARD:;
DISPLAY-SS7BOARD:HOSTNAME=host-A***,BOARDNM=***sbs334***;***

SAMPLE OUTPUT

```
MML_TH>dis-ss7board:;
-----------------------------------------------------------------------------------------
                      -
HOSTNAME BOARDNM INST CONF PM MODULES STATE CLASS PORTS LINES CLOCKMODE CLOCKSPAN SPMLINKNO
-----------------------------------------------------------------------------------------
                      -
ziwo     pci334    0  ON   OFF trmod  READY  I     4    0     NOTUSED      -        0
ziwo     pci3xpq   0  ON   OFF trmod  READY  III   24   2     LINE         1        1
ziwo     pci3xapq  0  ON   ON  pmmod  READY  III   24   2     INTERNAL     -        2
ziwo     pmc8260   0  ON   OFF trmod  READY  IV    64   4     LINE         2        3
ziwo     artic8260 0  ON   OFF trmod  READY  IV    64   4     LINE         1        4
<SUCCESS>:: 5 records found
```


Table 9-9: SS7BOARD Display Values

CLASS	LINES	STATE	SPMLINKNO
This read-only parameters identifies the board's preassigned hardware class type. The board's hardware determines the value. **I** class 1 64kbits/s board **II** class 2 E1 board **III**class 3 T1board **IV**class 4 E1/T1 configurable board	This read-only parameter identifies the number of lines on this board. • For E1/T1 boards, this parameter is the number of E1/T1 spans on the board hardware. • For 64K boards, this parameter is 0.	This read-only parameter is the state of this board, which is one of the following: • DETACHED • ATTACHED • CDWNLOADED (mtp/l2 downloaded) • READY • RESET	This read-only parameter corresponds to the stream number underneath the SPM multiplexer and is of interest because it shows exactly where the corresponding device driver has been linked. It is an unsigned integer from 0 to 7.

9.4.7 Level-2 Flow

NAME

L2FLOW Modifies or displays MTP Level-2 flow control information.

COMMANDS

MODIFY Modifies level-2 flow control values for a specific link. All links can have different flow control ***values***, however, all links on a signaling point must have the same flow control ***levels***.

MODIFY-L2FLOW:LINK=link***,FCLEVEL***=fclevel
[,CONGONVAL=congonval***][,CONGABVAL***=congabval***]***
[,DISCONVAL=disconval***][,DISCABVAL***=discabval***];***

DISPLAY Displays flow control information.

DISPLAY-L2FLOW:[LINK=link***][,FCLEVEL***=fclevel***];***

Table 9-10: Default Flow Control Values

FCLEVEL	CONGONVAL	CONGABVAL	DISCONVAL	DISCABVAL
Multiple Congestion and Multiple Priority: (e.g. ANSI)				
1	20	0	38	18
2	56	36	74	54
3	92	72	110	90
Standard ITU				
1	76	42	n/a	n/a

PARAMETERS

link Link identification entered as a 12-character alphanumeric label.

fclevel Numerical value for the flow control level or an * to display all values. The valid range depends on the setting of the ***mcong*** parameter in the MTP managed object:

- ON **1** to **3**
- OFF **1**

congonval The number of messages in the queue that indicate the onset of congestion. It is an unsigned integer from 0 to 127. Default values for the ***fclevels*** are in Table 9-10.

congabval The number of messages in the queue at which congestion ends. It is an unsigned integer from 0 to 127. Default values for the ***fclevels*** are in Table 9-10.

disconval The number of messages in the queue at which to begin discarding messages. It is an unsigned integer from 0 to 127. Default values for the ***fclevels*** are in Table 9-10.

discabval The number of messages in the queue at which to stop discarding messages. It is an unsigned integer from 0 to 127. Default values for the ***fclevels*** are in Table 9-10.

***Important**: DISCONVAL and DISCABVAL are applicable only if the **MPRIO** parameter of MTP MO is set to ON.*

ERRORS

<ERROR>::threshold value is out of range

<ERROR>::threshold level is out of range

<ERROR>::Missing FCLEVEL parameter.

<ERROR>::Missing LINK parameter

<ERROR>::CONGONVAL is less than previous level value.

<ERROR>::CONGONVAL is greater than next level value.

<ERROR>::CONGONVAL is less than CONGABVAL.

<ERROR>::CONGONVAL is greater than DISCONVAL.

<ERROR>::CONGABVAL is less than previous level value.

<ERROR>::CONGABVAL is greater than next level value.

<ERROR>::CONGABVAL is greater than CONGONVAL.

<ERROR>::CONGABVAL is greater than DISCABVAL.

<ERROR>::DISCONVAL is less than previous level value.

<ERROR>::DISCONVAL is greater than next level value.

<ERROR>::DISCONVAL is less than DISCABVAL.

<ERROR>::DISCONVAL is less than CONGONVAL.

<ERROR>::DISCABVAL is less than previous level value.

<ERROR>::DISCABVAL is greater than next level value.

<ERROR>::DISCABVAL is greater than DISCONVAL.

<ERROR>::DISCABVAL is less than CONGABVAL.

<ERROR>::LINK MO instance does not exist

<ERROR>::Nothing to list

EXAMPLES

MODIFY-L2FLOW:LINK=Link_1_1***,FCLEVEL***=2***,CONGONVAL***=60***;***
DISPLAY-L2FLOW:LINK=l1***,FCLEVEL***=****;***

SAMPLE OUTPUT

(MCONG=OFF, MPRIO=OFF)

```
MML_TH> DISPLAY-L2FLOW:LINK=11;
----------------------------------------------------------------------
LINKFCLEVELCONGONVALCONGABVALDISCONVALDISCABVAL
----------------------------------------------------------------------
11 1     76    42
<SUCCESS>:: 1 records found.
```

(MCONG=ON, MPRIO=OFF)

```
MML_TH> DISPLAY-L2FLOW:LINK=11;
----------------------------------------------------------------------
LINKFCLEVELCONGONVALCONGABVALDISCONVALDISCABVAL
----------------------------------------------------------------------
11 1     20    0     38    18
11 2     56    36    74    54
11 3     92    72    110   90
<SUCCESS>:: 3 records found.
```

9.4.8 Level-2 Timer (L2TIMER)

NAME

L2TIMER Modifies or displays the MTP Level-2 timer values.

COMMANDS

MODIFY Modifies MTP Level 2 timer values for a specific link from their defaults. All links can have different level-2 timer values.

MODIFY-L2TIMER:LINK=link***,TIMER***=timer***,VALUE***=value***;***

DISPLAY Displays the MTP Level 2 timer values.

DISPLAY-L2TIMER:[LINK=link***][,TIMER***=timer***];***

PARAMETERS

link Link identification entered as a 12-character alphanumeric label.

timer It is an unsigned integer from 0 to 8, as listed in Table 9-11 on page 9-59, or an * to display all timers for a given LINK.

value Numerical value specifying the timer value in milliseconds. It must be in the range defined in Table 9-11.

ERRORS

<ERROR>::Missing LINK parameter

<ERROR>::Missing TIMER parameter.

<ERROR>::LINK MO instance does not exist.

<ERROR>::Missing VALUE parameter.

<ERROR>::parameter value out of range.

<ERROR>::Nothing to list

EXAMPLES

MODIFY-L2TIMER:LINK=l11***,TIMER***=1***,VALUE***=120***;***
DISPLAY-L2TIMER:TIMER=l1***;***

SAMPLE OUTPUT

```
MML_TH> DISPLAY-L2TIMER:LINK=l1;
----------------------------------------------
LINK TIMER    VALUE     MINVAL MAXVAL
----------------------------------------------
l1          0      160          0    500
l1          1    13000      13000  13000
l1          2    11800      11500  23000
l1          3    11500      11500  11500
```



```
l1         4    2300       2300   2300
l1         5      80         80    120
l1         6    5000       3000  12000
l1         7    2000        500   6000
l1         8     600        600    600
<SUCCESS>:: 9 records found
```

***Note**: Level 2 timers have a precision of 20 ms and are rounded to the nearest multiple-of-20-ms value. Therefore, the values are multiples of 20.*

Table 9-11: MTP-L2 Timer Definitions

L2 Timers	Alias string	Integer	Range[1]		
			ITU-1992[2]	ANSI-1992[3]	ANSI-1996[4]
Timer for link idle-detect	T0	0	0 - 500 ms	0 - 500 ms	0 - 500 ms
Timer for aligned/ready	T1	1	40 s - 600 s	13.0 s	12.9 s - 16 s
Timer for not aligned/ waiting for destination activation	T2	2	5 s - 150 s	11.5 s - 23.0 s	5 s - 30 s
Timer for aligned/ waiting for alignment completion	T3	3	1 s - 2 s	11.5 s	5 s - 14 s
Normal Proving period timer	T4N	4	7.5 s - 120 s	2.3 s	2.3 s (+/- 10%)
Emergency Proving period timer	T4E	8	400 ms - 8s	600 ms	600 ms (+/- 10%)
Timer for sending SIB	T5	5	80 ms - 120 ms	80 ms - 120 ms	80 ms - 120 ms
Timer for monitoring remote congestion	T6	6	3 s - 12 s	3 s - 12 s	1 s - 2 s
Timer for excessive delay of acknowledgment	T7	7	500 ms - 6 s	500 ms - 6s	500 ms - 6 s

[1] For ranges: s=seconds; ms=milliseconds
[2] Timer values are rate-dependent; refer to CCITT Blue Book, Vol. 6, Fascicle VI.7-Rec. Q.703, Para.12
[3] Timer values are rate-dependent; refer to ANSI T1.111-1992, Para.12.3
[4] Timer values are rate-dependent; refer to ANSI T1.111-1996, Para.12.3

9.4.9 Level-3 Timer (L3TIMER)

NAME

L3TIMER Modifies or displays MTP Level-3 timer values.

COMMANDS

MODIFY Modifies MTP Level 3 timer values from their defaults.

MODIFY-L3TIMER:TIMER=timer***,VALUE***=value***;***

DISPLAY Displays information about the MTP level 3 timer values.

DISPLAY-L3TIMER:[TIMER=timer***];***

PARAMETERS

timer Numerical value for the MTP Level 3 timer, ranging from 1 to 31, as listed in Table 9-12 on page 9-61, or an * to display all timers.

value Numerical value specifying the timer value in milliseconds. It must be in the range defined in Table 9-12.

ERRORS

<ERROR>::Missing TIMER parameter.

<ERROR>::L3TIMER MO instance does not exist.

<ERROR>::Missing VALUE parameter.

<ERROR>::parameter value out of range.

<ERROR>::Nothing to list

EXAMPLES

MODIFY-L3TIMER:TIMER=1***,VALUE***=120***;***
DISPLAY-L3TIMER:TIMER=8***;***

SAMPLE OUTPUT

```
MML_TH> DISPLAY-L3TIMER:TIMER=8;

------------------------------------

TIMER    VALUE       MINVAL      MAXVAL

------------------------------------

8        1200        800          1200

<SUCCESS>:: 1 records found.
```


Note: Level 3 timers have a precision of 10 ms and are rounded to the nearest multiple-of-10-ms value. Therefore the values are multiples of 10.

Table 9-12: MTP-L3 Timer Definitions

L3 Timers	Alias String	Integer	ITU[1] Range[2]	ANSI-92/96[3] Range[2]
Delay to avoid message mis-sequencing on changeover	T1	1	0.5 - 1.2 s	0.5 - 1.2 s
Wait for changeover ACK	T2	2	0.7 - 2 s	0.7 - 2 s
Time-controlled delay on changeover	T3	3	0.5 - 1.2 s	0.5 - 1.2 s
First Wait for changeback ACK	T4	4	0.5 - 1.2 s	0.5 - 1.2 s
Second Wait for changeback ACK	T5	5	0.5 - 1.2 s	0.5 - 1.2 s
Delay to avoid message mis-sequencing on re-routing	T6	6	0.5 - 1.2 s	0.5 - 1.2 s
Wait for signaling data link connection ACK	T7	7	1 - 2 s	1 - 2 s
Transfer-prohibited inhibited timer (transient solution)	T8	8	0.8 - 1.2 s	0.8 - 1.2 s
Wait to repeat signaling route set test message	T10	10	30 - 60 s	30 - 60 s
Transfer-restricted timer	T11	11	30 - 90 s	30 - 90 s
Wait for uninhibit ACK	T12	12	0.8 - 1.5 s	0.8 - 1.5 s
Wait for force uninhibit	T13	13	0.8 - 1.5 s	0.8 - 1.5 s
Wait for inhibit ACK	T14	14	2 - 3 s	2 - 3 s
Wait to start route set congestion test	T15	15	2 - 3 s	2 - 3 s
Wait for route set congestion status update	T16	16	1.4 - 2 s	1.4 - 2 s
Delay to avoid oscillation of initial alignment failure and link re-start	T17	17	0.8 - 1.5 s	0.8 - 1.5 s
Wait for available links at re-starting STP (ITU) Repeat TFR once by response method (ANSI-96)	T18	18	40 s	2 s - 20 s
Wait for all traffic re-start messages at re-starting STP after T18	T19	19	67 - 69 s	480 - 600 s
Wait to broadcast traffic re-start allowed messages at re-starting STP after T19	T20	20	59 - 61 s	90 - 120 s
Wait to re-start traffic through adjacent SP at re-starting SP having no STP function	T21	21	63 - 65 s	90 - 120 s
Local inhibit test timer (ITU) Waiting for signaling links become available at restarting SP (ANSI)	T22	22	3 min - 6 min	90 - 120 s

[1] Timer values are rate-dependent; refer to ITU white book 1992. Rec. Q.704, Para. 16.8
[2] For ranges: s=seconds; ms=milliseconds; min=minutes
[3] Timer values are rate-dependent; refer to ITU 1993/1997. Rec Q.704, Para. 16.8

Table 9-12: MTP-L3 Timer Definitions (Continued)

L3 Timers	Alias String	Integer	ITU[1] Range[2]	ANSI-92/96[3] Range[2]
Remote inhibit test timer Waiting to receive all TRA message (ANSI)	T23	23	3 min - 6 min	10 s
Stabilizing after removal of LPO (ITU) Waiting to broadcast all TRA messages (ANSI)	T24	24	500ms	10 s
Waiting for traffic restart allowed message (ANSI)	T25	25	0 (not used)	30 - 35 s
Waiting for repeat traffic restart waiting message (ANSI	T26	26	0 (not used)	12 - 15 s
Minimum duration of unavailability for full restart	T27	27	0 (not used)	2 - 5 s
Waiting for traffic restart waiting message (ANSI)	T28	28	0 (not used)	3 - 35 s
Timer started when TRA sent in response to unexpected TRA or TRW (ANSI)	T29	29	0 (not used)	60 - 65 s
Timer to limit sending of TFPs and TFRs in response to unexpected TRA or TRW (ANSI)	T30	30	0 (not used)	30 - 35 s
False link congestion detection timer	T31	31	0 (not used)	10 - 120 s

[1] *Timer values are rate-dependent; refer to ITU white book 1992. Rec. Q.704, Para. 16.8*
[2] *For ranges: s=seconds; ms=milliseconds; min=minutes*
[3] *Timer values are rate-dependent; refer to ITU 1993/1997. Rec Q.704, Para. 16.8*

9.4.10 Line (LINE)

NAME

LINE — Modifies or displays the configuration of an SS7 line on a board.

COMMANDS

MODIFY — Modifies a class instance of an unconfigured SS7 board. LINE cannot be modified unless the SS7 board configuration is set to OFF (MODIFY-SS7BOARD:CONF=OFF;).

MODIFY-LINE:HOSTNAME=hostname***,BOARDNM***=boardnm***,***
INST=inst***,SPAN***=span***[,LINE_FRMMOD***=line_frmmod***]***
[,LINE_COD=line_cod***][,LINE_LEN***=line_len***]***
[,LINE_IMP=line_imp***][,LINE_LPBK***=line_lpbk***]***
[,LINE_NTFY=line_ntfy***][,LINE_ACCS***=line_accs***];***

DISPLAY — Display class instance(s).

DISPLAY-LINE:[HOSTNAME=hostname***][,BOARDNM***=boardnm***]***
[,INST=inst***][,SPAN***=span***];***

PARAMETERS

hostname — Name of host. It is a string of 1 to 15 alphanumeric characters maximum.

boardnm — Board type, entered as one of the following values:

- **sbs334** common name for 4-port Sbus boards (sbs37x)
- **pci334** common name for 4-port PCI bus boards (pci37x)
- **pci3xpq** common name for 24-port PCI bus boards (pci37xpq)
- **pci3xapq** common name for 24-port PCI bus boards (pci37xapq)
- **cpc3xpq** common name for 24-port CompactPCI bus boards (cpc370pq/cpc372pq)
- **pmc8260** common name for 64-port CompactPCI bus boards (pmc8260)
- **artic8260** common name for 64-port CompactPCI bus boards (artic1000 and artic2000)
- **pmc4539** common name for 128-port CompactPCI bus board (pmc4539f)
- **vbrd** 32-port virtual board driver
- **adaxm** common name for 124 port PCIe bus boards (HDCII-LPe)

Note: *ADAX boards can support both low speed and high speed links on the same card. For the low speed links the line type of the span must be set to either E1 or T1. For the high speed links the line type of the span must be set to either E1HSL or T1HSL. In order to configure an HSL link, time slot 1 of the span with line type E1HSL or T1HSL must be* *switched to an HDLC port. The high speed link will be detected automatically when this HDLC port is configured as a link at the MTP layer.*

Note: *Although PCI3xPQ, PCI3xAPQ and CPC3xPQ boards allow configuration of up to 24 links, use of more than 16 for PCI3xPQ boards is not recommended for systems requiring full bandwidth on all configured links.*

inst Identifies the SS7 board driver instance number.

span Identifies the span number; **1** or **2** for class II (E1), III (T1) boards and **1,2,3,4,5,6,7,** or **8** for class IV (E1/T1) boards.

line_frmmod Line framemod is a class II(E1) and class III(T1) type board parameter, entered as one of the following values:

- E1: **E1CRC4** *(default)*
  E1FEBE
  E1BASIC
- T1: **T1ESF** *(default)*
  T1ZBTSI
  T1SLC96
  TISF4
  T1SFRM

line_cod Line code is a class II (E1) and class III (T1) type board parameter, entered as one of the following values:

- E1: **E1HDB3** *(default)*
  AMI
- T1: **T1B8ZS** *(default)*
  T1B7ZS
  AMI

line_len Line length is a class III or class IV type board parameter, entered as one of the following values:

- **L133** for Class III boards *(default)*
- **L266** for Class III boards
- **L399** for Class III boards
- **L533** for Class III boards
- **L655** for Class III boards
- **L110** for Class IV boards *(default)*
- **L220** for Class IV boards
- **L330** for Class IV boards
- **L440** for Class IV boards
- **L550** for Class IV boards
- **L660** for Class IV boards

- **LB000** for Class IV boards
- **LB075** for Class IV boards
- **LB150** for Class IV boards
- **LB225** for Class IV boards

line_imp Line impedance is a class II (E1) type board parameter entered as one of the following values:

- E1: **I120** *(default)*
  I75

line_lpbk Loopback function is set with one of the following values:

- **NONE** *(default)*
- **LOCAL**
- **REMOTE**

line_ntfy Turns on or off the notification of framer line alarms with one of the following values:

- **ON** *(default)*
- **OFF**

line_typ Identifies the Primary Rate Interface. This parameter is used to set the line interface for Class IV type boards. For other boards (Class II and Class III) this is a read only parameter. It can be entered as a character string that accepts the following values:

- **E1** interface at 2048 kbit/sec
- **T1** interface at 1544 kbit/sec

Note: *E1/E1HSL and T1/T1HSL line types cannot be combined together on the same ADAX card due to clocking restrictions. ADAX cards can support either E1/E1HSL or T1/T1HSL Line types on its spans. If the line type of one span is switched then the line types of all the remaining spans are also switched automatically. The HSL flags are taken into consideration while the line type are being switched automatically. The existing E1HSL links are switched to T1HSL line type if any E1 span is switched to T1 on the adax board and vice versa.*

line_accs Selects front or rear access for line interfaces on ARTIC1000/2000 boards. For boards other than ARTIC1000 and ARTIC2000 an error will be returned. The default value is FRONT for SBS334, PCI334, PCI3XPQ, PCI3XAPQ, CPC3XPQ, PMC8260-F variant, ARTIC2000 and ARTIC1000 without RTB. The LINE_ACCS default value is REAR for PMC8260-R variant and ARTIC1000 with RTB. Acceptable values are:

- **FRONT**
- **REAR**

ERRORS

<ERROR>::Missing BOARDNM attribute
<ERROR>::Missing INST attribute
<ERROR>::Missing SPAN attribute
<ERROR>::LINE_LEN is class III (T1) attribute
<ERROR>::LINE_IMP is class II (E1) attribute
<ERROR>::No such a LINE MO instance
<ERROR>::Can not add line MOs
<ERROR>::Can not delete line MOs
<ERROR>::ACCESS value can not be modified for this board type

EXAMPLES

MODIFY-LINE:HOSTNAME=ultra5***,BOARDNM***=pci334***,INST***=0***,SPAN***=1***,***
LINE_FRMMOD=E1CR4***,LINE_COD***=E1HDB3***,LINE_IMP***=I120***,***
LINE_LPBK=NONE***,LINE_NTFY***=OFF***;***
DISPLAY-LINE:;
DISPLAY-LINE:HOSTNAME=ultra5***,BOARDNM***=pci334***,INST***=0***,SPAN***=1***;***

SAMPLE OUTPUT

```
 MML_TH>dis-line:;
---------------------------------------------------------------------------------------------------
HOSTNAME BOARDNM INST SPANCLASS LINE_TYPLINE_FRMMODLINE_CODLINE_LENLINE_IMPLINE_LPBKLINE_NTFY LINE_ACCS
---------------------------------------------------------------------------------------------------
diablo   pmc8260    0    1  IV       T1       T1ESF   T1B8ZS     L110  I100   NONE    OFF     FRONT
diablo   pmc8260    0    2  IV       T1       T1ESF   T1B8ZS     L110  I100   NONE    OFF     FRONT
diablo   pmc8260    0    3  IV       T1       T1ESF   T1B8ZS     L110  I100   NONE    OFF     FRONT
diablo   pmc8260    0    4  IV       T1       T1ESF   T1B8ZS     L110  I100   NONE    OFF     FRONT
diablo   artic8260  0    1  IV       T1       T1ESF   T1B8ZS     L110  I100   NONE    OFF      REAR
diablo   artic8260  0    2  IV       T1       T1ESF   T1B8ZS     L110  I100   NONE    OFF      REAR
diablo   artic8260  0    3  IV       T1       T1ESF   T1B8ZS     L110  I100   NONE    OFF      REAR
diablo   artic8260  0    4  IV       T1       T1ESF   T1B8ZS     L110  I100   NONE    OFF      REAR

<SUCCESS>:: 4 records found
```

Table 9-13: LINE Display Values

CLASS
This read-only parameter identifies the board’s preassigned hardware class type. The board’s hardware determines the value. **I** class 1 64 kbits/s board **II** class 2 E1 board **III** class 3 T1board **IV** class 4 E1/T1 configurable board

9.4.11 Link Status (LINKSTAT)

NAME

LINKSTAT Modifies or displays the state of an existing link.

COMMANDS

MODIFY Modifies the state of an existing link.

MODIFY-LINKSTAT:LINK=link***,STATUS***=status***;***

DISPLAY Displays status information about one or all existing links.

DISPLAY-LINKSTAT:[LINK=link***];***

PARAMETERS

link Link identification. It is a string of 1 to 12 alphanumeric characters maximum, or an ***** to display all links.

status State of the link. It is a one character string that accepts the following values:

- **SET_ACT** activate link
- **CLR_ACT** deactivate link
- **CLR_EMR** clear emergency link alignment
- **SET_EMR** set emergency link alignment
- **CLR_ECO** clear emergency changeover
- **SET_ECO** set emergency changeover
- **CLR_INH** clear inhibit
- **SET_INH** set inhibit
- **CLR_LPO** clear local processor outage
- **SET_LPO** set local processor outage
- **TEST_SLTM** send a single SLTM message over the link

ERRORS

<ERROR>::Missing LINK parameter.

<ERROR>::LINKSTAT MO instance does not exist.

<ERROR>::Parameter value is out of range.

<ERROR>::Nothing to list.

EXAMPLES

MODIFY-LINKSTAT:LINK=l1***,STATUS***=SET_ACT***;***
DISPLAY-LINKSTAT:;

SAMPLE OUTPUT

```
MML_TH> DISPLAY-LINKSTAT:;
------------------------------------------------------------------------
LINKLSET SLC   LOADEDACT   AVL   EMR   ECO   LIN   RIN   LPO   RPO
------------------------------------------------------------------------
l1 ls1   0     ON    ON    ON    OFF   OFF   OFF   OFF   OFF   OFF
l2 ls1   1     OFF   OFF   OFF   OFF   OFF   ON    OFF   OFF   OFF
l3 ls1   2     OFF   OFF   OFF   OFF   OFF   OFF   ON    OFF   OFF
l4 ls1   3     OFF   OFF   OFF   ON    ON    OFF   OFF   OFF   OFF
<SUCCESS>:: 4 records found.
```

Table 9-14: LINKSTAT Display Values

LSET	SLC	LOADED	ACT	AVL
Link Set label.	Signaling Link Code, which is an unsigned integer 0 to 5	Link state: • **ON**=Link is loaded • **OFF**= link is not loaded	Activation state: • **ON**=activated • **OFF**= not activated	Availability state: • ON=available • OFF= not available

EMR	ECO	LIN	RIN	LPO
Emergency alignment state: • **ON**=emergency alignment set • **OFF**= not set	Emergency change over state: • **ON**=emergency change over set • **OFF**= not set.	Locally inhibited state: • **ON**=locally inhibited • **OFF**= not locally inhibited.	Remotely inhibited state. • **ON**=remotely inhibited • **OFF**= not remotely inhibited.	Local processor outage state. • **ON**=local processor outage • **OFF**= no local processor outage.

RPO
Remote processor outage state. • **ON**=remote processor outage • **OFF**= no remote processor outage.

9.4.12 LinkSet Status (LSETSTAT)

NAME

LSETSTAT Modifies or displays the state of an existing link set.

COMMANDS

MODIFY Modifies the state of an existing link set.

MODIFY-LSETSTAT:LSET=lset***,STATUS***=status***;***

DISPLAY Displays status information about one or more link sets.

DISPLAY-LSETSTAT:[LSET=lset***];***

PARAMETERS

lset Link set identification. It is a string of 1 to 12 characters maximum, or an ***** to display all link sets.

status State of the link set. It is a one character string that accepts the following values:

- **SET_ACT** activate link set
- **CLR_ACT** deactivate link set

ERRORS

<ERROR>::Missing LSET parameter.

<ERROR>::LSETSTAT MO instance does not exist.

<ERROR>::Parameter value is out of range

<ERROR>::Nothing to list.

EXAMPLES

MODIFY-LSETSTAT:LSET=ls1***,STATUS***=SET_ACT***;***
DISPLAY-LSETSTAT:LSET=ls1***;***

SAMPLE OUTPUT

```
MML_TH> DISPLAY-LSETSTAT:;
--------------------------------
LSET      DPC   ACT    AVL
--------------------------------
ls1       2-2-2   ON     OFF
<SUCCESS>:: 1 records found.
```

Table 9-15: LSETSTAT Display Values

DPC	ACT	AVL
Destination Point Code	Activation state: • ON=activated • OFF= not activated	Availability state: • ON=available • OFF= not available

9.4.13 Port (PORT)

NAME

PORT — Modifies or displays the configuration of an SS7 port on a board.

COMMANDS

MODIFY — Sets the configuration of the SS7 port on a board. PORT cannot be modified until the SS7 board configuration is set to OFF. (MODIFY-SS7BOARD:CONF=OFF;)

MODIFY-PORT:HOSTNAME=hostname***,BOARDNM***=boardnm***,INST***=inst***,PORTNUM***=portnum***[,TYPE***=type***][,BAUD***=baud***][,LPBKMODE***=lpbkmode***][,IDLEDETECT***=idledetect***];***

DISPLAY — Displays the configuration settings of ports.

DISPLAY-PORT:[HOSTNAME=hostname***,BOARDNM***=boardnm***][,INST***=inst***][,PORTNUM***=portnum***];***

PARAMETERS

hostname — Name of the host. It is a string of 1 to 15 alphanumeric characters maximum.

boardnm — Board type, entered as one of the following values:

- **sbs334** common name for 4-port Sbus boards (sbs334/sbs37x)
- **pci334** common name for 4-port PCI bus boards (pci334/pci37x)
- **pci3xpq** common name for 24-port PCI bus boards (pci37xpq)
- **pci3xapq** common name for 24-port PCI bus boards (pci37xapq)
- **cpc3xpq** common name for 24-port CompactPCI bus boards (cpc370pq/cpc372pq)
- **pmc8260** common name for 64-port CompactPCI bus boards (pmc8260)
- **artic8260** common name for 64-port CompactPCI bus boards (artic1000 and artic2000)
- **pmc4539** common name for 128 port CompactPCI/PCI bus boards (PMC4539F)

- **vbrd** 32-port virtual board driver (vbrd)
- **adaxm** common name for 124 port PCIe bus boards (HDCII-LPe)

Note: *ADAX boards can support both low speed and high speed links on the same card. For the low speed links the line type of the span must be set to either E1 or T1. For the high speed links the line type of the span must be set to either E1HSL or T1HSL. In order to configure an HSL link, time slot 1 of the span with line type E1HSL or T1HSL must be switched to an HDLC port. The high speed link will be detected automatically when this HDLC port is configured as a link at the MTP layer.*

Note: *Although PCI3xPQ, PCI3xAPQ and CPC3xPQ boards allow configuration of up to 24 links, use of more than 16 for PCI3xPQ boards is not recommended for systems requiring full bandwidth on all configured links.*

inst Identifies the SS7 board driver instance number. The *getcfg* command provides the configured SS7 board slot, driver, and instance information.

portnum Port number in the SS7 board, entered as a numerical value from 0 to 23 for class II and II boards 0 to 63 for class IV boards.

type Clocking type of the port, only entered for class I boards:

- **DTE** *(default)*
- **DCE**

baud Baud rate for the link on the port, entered as one of the following:

- **600**
- **1200**
- **2400**
- **4800**
- **7200**
- **9600**
- **16000***
- **19200**
- **32000***
- **38400**
- **48000***
- **56000***
- **64000*** (*default*)
- **1544000***
- **2048000***

Note: *Ports on E1/T1 boards support only the E1/T1 sub-channeling baud rates that have an *.*

lpmode Loopback mode is an all class parameter. It is a character string that accepts the following values:

- **NONE** *(default)*
- **LOCAL**
- **REMOTE**

idledetect Sets the line idle-detection facility of the port ON or OFF. When idledetect is ON, the board software monitors the port receive side for the occurrence of a line idle state and informs the port about a possible line idle state which leads the link on that port to go out of alignment. This mechanism allows board software to detect line disconnects or other party transmission problems which cannot be detected easily in some cases. It is a character string that accepts the following values::

- **OFF** turns off idle-detection
- **ON** turns on idle-detection *(default)*

This parameter enables/disables idle-detection through the board configuration commands. MTPL3 has also the MODIFY-L2TIMER command to enable/disable idle-detection by modifying the T0 value.

ERRORS

<ERROR>::Missing BOARDNM attribute

<ERROR>::Missing INST attribute

<ERROR>::Missing PORTNUM attribute

<ERROR>::Can not add port MOs

<ERROR>::Can not delete port MOs

<ERROR>::Port is already inuse

<ERROR>::Port is already free

<ERROR>::No such a PORT MO instance

EXAMPLES

MODIFY-PORT:HOSTNAME=host-A***,BOARDNM=***sbs334***,INST=***0***,***
PORTNUM=0***,BAUD=***64000***,LPBKMODE=***NONE***,IDLEDETECT=***ON***;***
DISPLAY-PORT:;
DISPLAY-PORT:HOSTNAME=ultra5***,BOARDNM=***pci334***,INST=***0***,***
PORTNUM=0***;***

SAMPLE OUTPUT

```
MML_TH>display-port:;
------------------------------------------------------------------------
HOSTNAME BOARDNM INST PORTNUM CLASS     TYPE   BAUD LPBKMODE IDLEDETECT
------------------------------------------------------------------------
ziwo    pci3xpq    0        0    III  NOTUSED  64000     NONE         ON
```



```
ziwo     pci3xpq     0          1    III   NOTUSED   64000      NONE          ON
ziwo     pci3xpq     0          2    III   NOTUSED   64000      NONE          ON
ziwo     pci3xpq     0          3    III   NOTUSED   64000      NONE          ON
<SUCCESS>:: 4 records found
```

9.4.14 MTP SLTM Timer (SLTIMER)

NAME

SLTIMER   Modifies or displays the MTP SLTM timer values

COMMANDS

MODIFY   Modifies MTP SLTM (Q.707) timer values.

MODIFY-SLTIMER:TIMER=timer***,VALUE***=value***;***

DISPLAY   Displays information about the MTP SLTM (Q.707) timer values.

DISPLAY-SLTIMER:[TIMER=timer***];***

PARAMETERS

timer   MTP SLTM (Q.707) timer id entered as an integer from 1 to 2, or an * to display all MTP SLTM timers.

value   Timer value entered as an integer. Valid ranges are in Table 9-16 on page 9-74.

ERRORS

<ERROR>::SLTIMER MO instance does not exist

<ERROR>::missing TIMER parameter

<ERROR>::missing VALUE parameter

<ERROR>::parameter value is out of range

<ERROR>::nothing to list

EXAMPLES

MODIFY-SLTIMER:TIMER=1***,VALUE***=120***;***
DISPLAY-SLTIMER:;

***Note**: Level 3 timers have a precision of 10 ms and are rounded to the nearest multiple-of-10-ms value. Therefore the values shall be multiples of 10.*

Table 9-16: SLTIMER Definitions

SLTM Timers	Alias String	Integer	ITU[1] Range[2]	ANSI-92/96[3] Range[2]
Supervision timer for signaling link test acknowledgment message	T1	1	4 - 12 s	4 - 12 s
Interval timer for sending signaling link test messages	T2	2	30 -90 s	30 - 90 s

1 Timer values are rate-dependent; refer to ITU white book 1992. Rec. Q.707, Para.5.5
2 For ranges: s=seconds
3 Timer values are rate-dependent; refer to ITU 1993/1997. Rec. Q.704, Para.5.5

SAMPLE OUTPUT

```
MML_TH>DISPLAY-SLTIMER:;
-------------------------------
TIMER   VALUE    MINVAL MAXVAL
-------------------------------
1       6000       4000   12000
2       90000      30000  90000
<SUCCESS>:: 2 records found
```

9.4.15 Signaling Point (SP)

NAME

SP Modifies or displays signaling point parameters.

COMMANDS

MODIFY Modifies signaling point parameters of the Distributed7 SS7 node.

MODIFY-SP:[SPNO**=spno**][,NAME**=name**][,SPC**=spc**][,NI**=ni**] [,TYPE**=type**];

DISPLAY Displays network and signaling information about the own signaling point.

DISPLAY-SP:[SPNO**=spno**];

PARAMETERS

spno Signaling point number entered as an integer from 0 to 7, or an ***** to display all signaling points. The SP number at which this command is entered is used if this optional parameter is not entered with the DISPLAY command,

name Name of signaling point. Entered as a 10-character alphanumeric label.

spc Own signaling point code. It is entered in a X-X-X format, where the sum of the Xs must equal the bits in the PCSIZE parameter of the MTP MO:

Sample DPC	PCSIZE Parameter	Sum
3-8-3	14_BIT	3+8+3=14
5-4-7	16_BIT	5+4+7=16
8-8-8	24_BIT	8+8+8=24

ni Network indicator. It is a character string that accepts the following values:

- **INTERNATIONAL** (international)
- **SPARE** (international with spare bits set)
- **NATIONAL** (national)
- **RESERVED** (national with reserved bits set).

type Type of signaling point. It is a character string that accepts the following values:

- **STP** (Signaling Transfer Point)
- **SEP** (Signaling End Point)
- **SEPWRT** (Signaling End Point with Routing Option).

Important*: All the parameters are optional, but at least one of them must be entered.*

ERRORS

<ERROR>::SP MO instance does not exist.

<EUUPEXISTS>::In order to modify SPC/NI parameters all user parts must terminate.

EXAMPLES

MODIFY-SP:NI=NATIONAL***,SPC***=10-20-10***,TYPE***=STP***;***
DISPLAY-SP:;

SAMPLE OUTPUT

```
MML_TH> DISPLAY-SP:;
------------------------------------------------
SPNO     NAME  SPC       NI       TYPE
------------------------------------------------
0        DAMGR 254-1-23 NATIONAL SEP
<SUCCESS>:: 1 records found.
```

9.4.16 Alias Point Code (ALIAS)

NAME

ALIAS Adds modifies, deletes, or displays alias point code information.

COMMANDS

ADD Adds second point code to the node.

ADD-ALIAS:APC=apc***[,OGPC***=ogpc***][,INFLTR***=infltr***]***
[,FLTRACT=fltract***];***

MODIFY Modifies any of the attributes of alias point code

MODIFY-ALIAS:[APC=apc***][,OGPC***=ogpc***][,INFLTR***=infltr***]***
[,FLTRACT=fltract***];***

DELETE Deletes an alias point code.

DELETE-ALIAS:[APC=apc***];***

DISPLAY Displays the configuration information of alias point code.

DISPLAY-ALIAS:[APC=apc***];***

Important*: ADD, MODIFY, and DELETE operations for Alias Point Code are allowed only when MTP state is Created or Isolated.*

PARAMETERS

apc Alias point code of the node. It is entered in a X-X-X format, where the sum of the Xs must equal the bits in the PCSIZE parameter of the MTP MO:

Sample DPC	PCSIZE Parameter	Sum
3-8-3	14_BIT	3+8+3=14
5-4-7	16_BIT	5+4+7=16
8-8-8	24_BIT	8+8+8=24

ogpc Outgoing point code. Based on this parameter, decision is taken as to which point code is to be populated in the originating point code of the routing label. Values are:

- **OFF** Originating Point Code field of the routing label is populated with the Signaling Point Code (SPC) of the node.
- **ON** Originating Point Code field of the routing label is populated with the Alias Point Code (APC) of the node. (*default*)

infltr Incoming message filter. This parameter indicates if any incoming userpart messages are to be filtered. Values are:

- **OFF** Incoming userpart messages to either SPC or APC are not filtered.
- **SPC** Incoming userpart messages to SPC are filtered. (*default*)

- **APC** Incoming userpart messages to APC arefiltered.

fltract Filter action. This parameter indicates the action to be taken for the filtered messages. Values are:

- **ALARM** Filtered messages aree discarded with an alarm.
- **UPU** Filtered messages are discarded with an alarm, and a "UserPartUnavailable" message is sent to the originator. *(default)*

ERRORS

<ERROR>::ALIAS MO does not exist

<ERROR>::Nothing to list

<ERROR>::ALIAS MO instance already exists

<ERROR>::MTP protocol not set

EXAMPLES

ADD-ALIAS:APC=2-3-4***,OGPC=***OFF***,INFLTR=***APC***,FLTRACT=***ALARM***;***
DISPLAY-ALIAS:;

SAMPLE OUTPUT

```
MML_TH> DISPLAY-ALIAS:;
------------------------------------------------------------------------
ALIAS    OGPC  INFLTR      FLTRACT
------------------------------------------------------------------------
2-3-4    ON    SPC         UPU
<SUCCESS>:: 1 records found
```

9.4.17 Time Slot (TIMESLOT)

NAME

TIMESLOT Modifies or displays the configuration of an SS7 time slot on a board.

COMMANDS

MODIFY Sets the configuration of the SS7 time slot on the board. A time slot cannot be modified until the SS7 board configuration is set to OFF (MODIFY-SS7BOARD:CONF=OFF;). The initial state for all time slots is ORIGTYPE=NOCONNECT and ORIGSLOT=0. This is because there is a time slot conflict detection mechanism that is activated with the MODIFY-TIMESLOT command. If any two time slots have the same ORIGTYPE and ORIGSLOT value pair (except ORIGTYPE=NOCONNECT) an error is returned informing the user about the conflict. This mechanism prevents accidental misuse of time slot mapping.

NOCONNECT timeslots are no longer kept in the TIMESLOT database. Therefore, only timeslots with an origtype other than NOCONNECT are displayed and if a timeslot origtype is modified to NOCONNECT its record is deleted from database.

For CTBUS spans the number of available slots changes with the data rate of a span that can be done with MODIFY-CTBUS command. Please refer to Section 9.4.21 on page 9-88 for more detail about this issue.

MODIFY-TIMESLOT:HOSTNAME=hostname***,***
BOARDNM=boardnm***,INST***=inst***,DESTTYPE***=desttype***,***
DESTSPAN=destspan***,DESTSLOT***=destslot***,***
ORIGTYPE=origtype***, ORIGSLOT***=origslot,
ORIGSPAN=origspan***;***

DISPLAY Displays the configuration settings for time slots.

DISPLAY-TIMESLOT:[HOSTNAME=hostname***]***
[,BOARDNM=boardnm***][,INST***=inst***][,DESTTYPE***=desttype***]***
[,DESTSPAN=destspan***][,DESTSLOT***=destslot***]***
[,ORIGSPAN=origspan***];***

PARAMETERS

hostname Name of the host. It is a string of 1 to 15 alphanumeric characters maximum.

boardnm Board type, entered as one of the following values:

- **sbs334** - common name for 4-port Sbus boards (sbs334/sbs37x)
- **pci334** - common name for 4-port PCI bus boards (pci334/pci37x)
- **pci3xpq** - common name for 24-port PCI bus boards (pci37xpq)
- **pci3xapq** - common name for 24-port PCI bus boards (pci37xapq)

- **cpc3xpq** - common name for 24-port CompactPCI bus boards (cpc370pq/cpc372pq)
- **pmc8260** - common name for 64-port CompactPCI bus boards (pmc8260)
- **artic8260** common name for 64-port CompactPCI bus boards (artic1000 and artic2000)
- **vbrd** -32-port virtual driver board
- **adaxm**common name for 124 port PCIe bus boards (HDCII-LPe)

Note: *ADAX boards can support both low speed and high speed links on the same card. For the low speed links the line type of the span must be set to either E1 or T1. For the high speed links the line type of the span must be set to either E1HSL or T1HSL. In order to configure an HSL link, time slot 1 of the span with line type E1HSL or T1HSL must be switched to an HDLC port. The high speed link will be detected automatically when this HDLC port is configured as a link at the MTP layer.*

Note: *Although PCI3xPQ, PCI3xAPQ and CPC3xPQ boards allow configuration of up to 24 links, use of more than 16 for PCI3xPQ boards is not recommended for systems requiring full bandwidth on all configured links.*

inst Identifies the SS7 board driver instance number. The **getcfg** command provides the configured SS7 board slot, driver, and instance information.

desttype Identifies the destination span, the span to which PCM time slot data is directed. Desttype must be one of the following values:

- **LINE** (spans 1 to 2 on the class II, III boards and spans 1 to 8 on class IV boards; LINE-1 maps to the older LINEB, and LINE-2 maps to the older LINEA)
- **HDLC** (HDLC controllers on the E1/T1 board)
- **CTBUS** (CT bus spans 0-31 on the class IV board)

destspan Identifies the destination span number if the ***desttype*** parameter is either **LINE** or **CTBUS**. Ignored for ***desttype*** values other than **LINE** and **CTBUS**. For class II and III boards the range is 1 to 2. For class IV boards the range is 1 to the number available physical spans on the board if ***desttype*** is **LINE**, or 0 to 31 if ***desttype*** is **CTBUS**.

destslot Identifies the destination slot, which is the slot of the destination span to which PCM time slot data is directed.

- For LINE[1-8] spans:
  - (E1) board values must be in the range of **0** to **31**
  - (T1) board values must be in the range of **0** to **23**
- For HDLC spans, the destslot is in the range of **0** to the maximum number of ports on board.
- For CTBUS spans, the destslot is in the range of **0** to **127**

origtype Identifies the source span, the span from which PCM time slot data is coming from. Origtype can be one of the following values:

- **LINE** (spans 1 to 2 on the class II, III boards and spans 1 to 8 on class IV boards; LINE-1 maps to the older LINEB, and LINE-2 maps to the older LINEA)
- **HDLC** (HDLC controllers on the E1/T1 board)
- **NOCONNECT** (no connection designator, not a real span)
- **CTBUS** (CT bus spans 0-31 on the class IV board)

origspan Identifies the destination span number if the ***origtype*** parameter is either **LINE** or **CTBUS**. Ignored for ***origtype*** values other than **LINE** and **CTBUS**. For class II and III boards the range is 1 to 2. For class IV boards the range is 1 to the number available physical spans on the board if ***origtype*** is **LINE**, or 0 to 31 if ***origtype*** is **CTBUS**.

origslot Identifies the source slot, the slot of the source span from which PCM time slot data is coming from.

- For LINE[1-8] spans:
  - (E1) board values must be in the range of **0** to **31**
  - (T1) board values must be in the range of **0** to **23**
- For HDLC spans, the destslot is in the range of **0** to the maximum number of ports on board.
- For CTBUS spans, the destslot is in the range of **0** to **127**
- NOCONNECT spans must be **0**

ERRORS

<ERROR>::Missing BOARDNM attribute

<ERROR>::Missing INST attribute

<ERROR>::Missing DESTTYPE attribute

<ERROR>::Missing DESTSLOT attribute

<ERROR>::Missing ORIGTYPE attribute

<ERROR>::Missing ORIGSLOT attribute

<ERROR>::Can not add timeslot MOs

<ERROR>::Can not delete timeslot MOs

<ERROR>::Self connection in TIMESLOT assignment not allowed

<ERROR>::ORIGSLOT value must be 0 for NOCONNECT

<ERROR>::No such a TIMESLOT MO instance

<ERROR>::HDLC timeslot (port) listens to NOCONNECT

<ERROR>::HDLC timeslot (port) not listened by a timeslot

<ERROR>::No timeslot record(s) found, timeslot(s) in NOCONNECT state

EXAMPLES

MODIFY-TIMESLOT:HOSTNAME=host-A***,BOARDNM=****sbs334****,INST=****0***, ***DEST-TYPE***=*HDLC****,DESTSLOT=****0****,ORIGTYPE=****LINEB***, ***ORIGSLOT***=0***;***
DISPLAY-TIMESLOT:;

SAMPLE OUTPUT

```
MML_TH>dis-timeslot:;
--------------------------------------------------------------------------------------------------------------------
HOSTNAME BOARDNM INST DESTTYPE DESTSPAN DESTSLOT CLASS ORIGTYPE ORIGSPAN ORIGSLOT
--------------------------------------------------------------------------------------------------------------------
ziwo     pmc8260 0    LINE      1         1       IV    HDLC      -       0
ziwo     pmc8260 0    LINE      1         2       IV    HDLC      -       1
ziwo     pmc8260 0    LINE      2         1       IV    HDLC      -       2
ziwo     pmc8260 0    LINE      2         2       IV    HDLC      -       3
ziwo     pmc8260 0    HDLC      -         0       IV    LINE      1       1
ziwo     pmc8260 0    HDLC      -         1       IV    LINE      1       2
ziwo     pmc8260 0    HDLC      -         2       IV    LINE      2       1
ziwo     pmc8260 0    HDLC      -         3       IV    LINE      2       2
ziwo     pmc8260 0    HDLC      -         4       IV    CTBUS     0       0
ziwo     pmc8260 0    HDLC      -         5       IV    CTBUS     0       1
ziwo     pmc8260 0    CTBUS     0         0       IV    HDLC      -       4
ziwo     pmc8260 0    CTBUS     0         1       IV    HDLC      -       5
```

Table 9-17: TIMESLOT Display Values

CLASS
This read-only parameters identifies the preassigned hardware class type of the board to which this time slot belongs. Board hardware determines the class parameter value, which can be one of the following: II : class 2 E1 type line III : class 3 T1 type line IV : class 4 E1/T1 configurable line

9.4.18 MTP Level-2 Status (L2CS)

NAME

L2CS Displays MTP Level-2 status information.

COMMANDS

DISPLAY Displays mtp-l2 cumulative status information.

DISPLAY-L2FLOW:[LINK=link***];***

PARAMETERS

link Link nam. It is a string of 1 to 12 alphanumeric characters maximum, or an * to display all links.

ERRORS

<ERROR>::LINK MO instance does not exist

<ERROR>::nothing to list

EXAMPLES

DISPLAY-L2CS:LINK=l1***;***

SAMPLE OUTPUT

```
MML_TH> DISPLAY-L2CS:LINK=l1;
----------------------------------------------------------------------------------------------------
LINK STAT TMINSRV SUERM ALGNF LINKF  RSU_E   D_RXL   D_TXL   D_BO TXFRAMES RXFRAMES TXOCTETS RSOCTETS
----------------------------------------------------------------------------------------------------
l11  IS   6         0   2     0       15     0       0       0    3        3        71         71
<SUCCESS>:: 1 records found
```


9.4.19 Line Statistics (LINESTAT)

NAME

LINESTAT Modifies or displays statistics and performance parameters of an SS7 line on a board.

COMMANDS

MODIFY Modifies the statistics and performance parameters of an SS7 line on a board.

*Note: **errevents** is the only parameter that can be modified.*

MODIFY-LINESTAT: HOSTNAME=hostname***,***
BOARDNM=boardnm***, INST***=inst***, SPAN***=span***,***
ERREVENTS=errevents***;***

DISPLAY Displays statistics and performance parameters of an SS7 line on a board.

DISPLAY-LINESTAT:[HOSTNAME=hostname***]***
[,BOARDNM=boardnm***][,INST***=inst***][,SPAN***=span***];***

PARAMETERS

hostname Name of host. It is a string of 1 to 15 alphanumeric characters maximum.

boardnm Board type, entered as one of the following values:

- **pci3xpq** common name for 24-port PCI bus boards (pci3xpq)
- **pci3xapq** common name for 24-port PCI bus boards (pci3xapq)
- **cpc3xpq** common name for 24-port CompactPCI bus boards (cpc3xpq)
- **adaxm** common name for 124 port PCIe bus boards (HDCII-LPe)

***Note**: Although PCI3xPQ, PCI3xAPQ and CPC3xPQ boards allow configuration of up to 24 links, use of more than 16 for PCI3xPQ boards is not recommended for systems requiring full band-width on all configured links.*

inst Identifies the SS7 board driver instance number. The getcfg command provides the configured SS7 board slot, driver, and instance information.

span Identifies the span number; **1** or **2** for class II (E1), III (T1) boards and **1,2,3,4,5,6,7,** or **8** for class IV (E1/T1) boards.

errevents Error events counter, entered as **0** to reset the counter.

ERRORS

<ERROR>:: No such SS7BOARD MO instance

<ERROR>:: Board name must be one of class II or III boards

<ERROR>:: Missing BOARDNM attribute

<ERROR>:: Missing INST attribute

<ERROR>:: Missing SPAN attribute

<ERROR>:: Can not get line statistics

<ERROR>:: Can not modify line statistics

EXAMPLE

MODIFY-LINESTAT:HOSTNAME=ultra5***,*** ***BOARDNM***=pci3xpq***,*** ***INST***=0***,*** ***SPAN***=1***,*** ***ERREVENTS***=0***;***

DISPLAY-LINESTAT:HOSTNAME=ultra5***,*** ***BOARDNM***=pci3xpq***,*** ***INST***=0***,*** ***SPAN***=1***;***
DISPLAY-LINESTAT:HOSTNAME=ultra5***,*** ***BOARDNM***=pci3xpq***,*** ***INST***=0***;***
DISPLAY-LINESTAT:HOSTNAME=ultra5***,*** ***BOARDNM***=pci3xpq***;***
DISPLAY-LINESTAT:HOSTNAME=ultra5***;***
DISPLAY-LINESTAT:;

SAMPLE OUTPUT

```
MML_TH>display-linestat:;
-----------------------------------------------------------------------------------------------
HOSTNAME BOARDNM INST SPAN ERREVENTS CURSTATUS CURTIMER CUR-ES CUR-UAS 24H-ES 24H-UAS VLDINTTOTAL
-----------------------------------------------------------------------------------------------
  ultra5   pci3xpq    0     1 6018    SIG-AV      266      0       0      4      15          96
  ultra5   pci3xpq    0     2 979    SIG-AV      265      0       0      4      0          96
  ultra5  pci3xapq    0     1 0      SIG-AV      263      0       0      1      15          96
  ultra5  pci3xapq    0     2 4      SIG-AV      263      0       0      1      0          96
<SUCCESS>:: 4 records found
```

Table 9-18: LINESTAT Display Values

ERREVENTS	CURSTATUS	CURTIMER
Error events counter. An error event is defined as the occurrence of a CRC6 error or an Out of Frame error.	Current signal status. An unavailable signal state is declared after 10 consecutive seconds each has 320 or more CRC6 error events OR one or more Out of Frame error. • SIG-AV =Available • SIG-UNAV= Unavailable	Current 15-minute interval timer.

Table 9-18: LINESTAT Display Values

CUR-ES	CUR-UAS	24H-ES
Number of errored seconds in current 15-minute interval. An errored second is a second with one or more error events.	Number of unavailable seconds in current 15-minute interval. An unavailable second is a second when the signal state is unavailable.	Number of errored seconds in previous 24-hour period.

24H-UAS	VLDINTTOTAL
Number of unavailable seconds in previous 24-hour period.	Number of valid 15-minute intervals in previous 24-hour period.

9.4.20 Line 24-Hour Performance Data (LINEHIST)

NAME

LINEHIST Modifies or displays 24-hour performance data of an SS7 line on a board.

COMMANDS

MODIFY Modifies the 24-hour performance data of an SS7 line on a board.

MODIFY-LINEHIST: HOSTNAME=hostname**,**
BOARDNM=boardnm**,** ***INST***=inst**,** ***SPAN***=span**,**
RESET=reset***;***

DISPLAY Displays the 24-hour performance data of an SS7 line on a board.

DISPLAY-LINEHIST:[HOSTNAME=hostname***]***
[,BOARDNM=boardnm***][,INST***=inst***][,SPAN***=span***];***

PARAMETERS

hostname Name of host. It is a string of 1 to 15 alphanumeric characters maximum.

boardnm Board type, entered as one of the following values:

- **pci3xpq** common name for 24-port PCI bus boards (pci3xpq)
- **pci3xapq** common name for 24-port PCI bus boards (pci3xapq)
- **cpc3xpq** common name for 24-port CompactPCI bus boards (cpc3xpq)
- **pmc4539** common name for 128-port CompactPCI/PCI bus boards (PMC4539F)
- **adaxm** common name for 124 port PCIe bus boards (HDCII-LPe)

***Note**: Although PCI3xPQ, PCI3xAPQ and CPC3xPQ boards allow configuration of up to 24 links, use of more than 16 for PCI3xPQ boards is not recommended for systems requiring full bandwidth on all configured links.*

inst	Identifies the SS7 board driver instance number. The getcfg command provides the configured SS7 board slot, driver, and instance information.
span	Identifies the span number; **1** or **2** for class II (E1), III (T1) boards and **1,2,3,4,5,6,7,** or **8** for class IV (E1/T1) boards.
reset	Reset option, entered as either **YES** or **NO**. If entered as **YES** all of the the history will be deleted.

ERRORS

<ERROR>:: No such SS7BOARD MO instance

<ERROR>:: Board name must be one of class II or III boards

<ERROR>:: Missing BOARDNM attribute

<ERROR>:: Missing INST attribute

<ERROR>:: Missing SPAN attribute

<ERROR>:: Can not get 24-Hour performance data

<ERROR>:: Can not modify 24-Hour performance data

EXAMPLE

MODIFY-LINEHIST:HOSTNAME=ultra5***, BOARDNM***=pci3xpq***, INST***=0***, SPAN***=1***, RESET***=YES***;***
DISPLAY-LINEHIST:HOSTNAME=ultra5***, BOARDNM***=pci3xpq***, INST***=0***, SPAN***=1***;***
DISPLAY-LINEHIST:HOSTNAME=ultra5***, BOARDNM***=pci3xpq***, INST***=0***;***
DISPLAY-LINEHIST:HOSTNAME=ultra5***, BOARDNM***=pci3xpq***;***
DISPLAY-LINEHIST:HOSTNAME=ultra5***;***
DISPLAY-LINEHIST:;

SAMPLE OUTPUT

```
MML_TH>display-linehist: hostname=ultra5, boardnm=pci3xpq, inst=0, span=1;
-------------------------------------------------------
         HOSTNAME   BOARDNM INST SPAN INTERVAL   ES UAS
-------------------------------------------------------
           ultra5   pci3xpq    0    1        1    0   0
           ultra5   pci3xpq    0    1        2    0   0
           ultra5   pci3xpq    0    1        3    0   0
           ultra5   pci3xpq    0    1        4    0   0
           ultra5   pci3xpq    0    1        5    0   0
           ultra5   pci3xpq    0    1        6    0   0
           ultra5   pci3xpq    0    1        7    0   0
           ultra5   pci3xpq    0    1        8    0   0
           ultra5   pci3xpq    0    1        9    0   0
```



```
                    ultra5   pci3xpq    0     1        10     0    0
                    ultra5   pci3xpq    0     1        11     0    0
                    ultra5   pci3xpq    0     1        12     0    0
                    ultra5   pci3xpq    0     1        13     0    0
                    ultra5   pci3xpq    0     1        14     0    0
                    ultra5   pci3xpq    0     1        15     0    0
                    ultra5   pci3xpq    0     1        16     0    0
                    ultra5   pci3xpq    0     1        17     0    0
                    ultra5   pci3xpq    0     1        18     0    0
                    ultra5   pci3xpq    0     1        19     0    0
                    ultra5   pci3xpq    0     1        20     0    0
                    ultra5   pci3xpq    0     1        21     4   15
          <SUCCESS>:: 21 records found
```

Table 9-19: LINESTAT Display Values

INTERVAL	ES	UAS
15-minute interval index. Interval 1 represents the most recent 15-minute interval.	Number of errored seconds. An errored second is a second with one or more error events.	Number of unavailable seconds. An unavailable second occurs when the signal state is unavailable.

9.4.21 CT Bus (CTBUS)

NAME

CTBUS Modifies or displays the configuration of the CT Bus logic on a board. For more information about CT Bus (H.110/H.100), please refer to documents 'H.110/H.100 Hardware Compatibility Specification: CT Bus, revision 1.0, 1997, Enterprise Computer Telephony Forum (ECTF)'. These documents can be downloaded from web site of ECTF at http://www.ectf.org/.

COMMANDS

MODIFY Sets the configuration of the CT Bus on the board. The CT Bus cannot be modified until the SS7 board configuration is set to OFF (MODIFY-SS7BOARD:CONF=OFF;).

CTBUS MO applies to class IV boards like PMC8260, ARTIC1000 and ARTIC2000. Use of MODIFY-CTBUS command for boards other than the mentioned above will return an error.

MODIFY-CTBUS:HOSTNAME=hostname***,BOARDNM***=boardname***, INST***=instance***[,REFCLK***=reflck***][,REFINV***=refinv***][,FBMODE***=fbmode***] [,FBSPAN***=fbspan***][,COMP***=comp***][,C8A***=c8a***][,C8B***=c8b***] [,NRMODE***=nrmode***][,NRPSAN***=nrspan***][,NR8KHZ***=nr8khz***][,NRINV***=nrinv***] [,NR1***=nr1***][,NR2***=nr2***][,GRP_A***=grp_a***][,GRP_B***=grp_b***][,GRP_C***=grp_c***]; [,GRP_D***=grp_d***][,GRP_E***=grp_e***][,GRP_F***=grp_f***][,GRP_G***=grp_g***] [,GRP_H***=grp_h***];***

DISPLAY Displays the configuration settings and status for ctbus. The fallback status and the netref clock output status are shown if the board is configured ON otherwise a '-' will be printed. When fallback status ***FB*** column is **ON** the fallback clock selection is active. When netref clock output status **NRACT** is **OFF** netref output has been shut off by the board due to a reference clock loss.

DISPLAY-CTBUS:[HOSTNAME=hostname***][,BOARDNM***=boardnm***][,INST***=inst***];***

PARAMETERS

hostname Name of the host. It is a string of 1 to 15 alphanumeric characters maximum.

boardnm Board type, entered as one of the following values:

- **pmc8260** - common name for 64-port CompactPCI bus boards
- (pmc8260)
- **artic8260** - common name for 64-port CompactPCI bus boards
- (artic8260)

inst Identifies the SS7 board driver instance number. The **getcfg** command provides the configured SS7 board slot, driver, and instance information.

refclk Identifies primary reference clock source when board is in ***REMOTE*** clock mode. Refclk must be one of the following values:

- **C8A,C8B** ECTF H.1x0 bus 8.192Mhz A or B clocks.
- **NETREF1,NETREF2** ECTF H.1x0 bus network reference clocks.
- **SCSA2,SCSA4,SCSA8** SCSA bus 2.048, 4.096 or 8.192 Mhz clocks.
- **MVIP,HMVIP** MVIP or H-MVIP bus clocks, H.100 bus only.

refinv Identifies primary reference clock inversion switch value. Refinv must be one of the following values:

- **ON** invert primary reference clock.
- **OFF** do not invert primary reference clock.

fbmode Identifies fallback clock mode activated when primary reference clock failed. Fbmode must be one of the following values:

- **C8A,C8B** ECTF H.1x0 bus 8.192Mhz A or B clocks.
- **NETREF1,NETREF2** ECTF H.1x0 bus network reference clocks.
- **INTERNAL** Internal board clock.
- **LINE** Line recovered clock, requires FBSPAN to be entered.

fbspan Identifies fallback clock span when FBMODE value is LINE. Fbspan must be one of the following values: **1,2,3,4,5,6,7,8.**

comp Identifies compatibility clocks output mode selection. Comp must be one of the following values:

- **ECTF** disables compatibility clocks.
- **SCSA2,SCSA4,SCSA8** enables 2.048, 4.096 or 8.192 MHz SCSA clocks.
- **MVIP,HVMIP** enables MVIP or HMVIP clocks, H.100 bus only.

c8a Identifies ECTF H.1x0 C8A clock output enable/disable switch value. C8a must be one of the following values:

- **ON** enable C8A clock output.
- **OFF** disable C8A clock output.

c8b Identifies ECTF H.1x0 C8B clock output enable/disable switch value. C8b must be one of the following values:

- **ON** enable C8B clock output.
- **OFF** disable C8B clock output.

nrmode Identifies network reference clock source mode. Nrmode must be one of the following values:

- **NETREF1,NETREF2** ECTF H.1x0 bus network reference clocks.
- **INTERNAL** Internal board clock.
- **LINE** Line recovered clock, requires NRSPAN to be entered.

nrspan Identifies network reference clock span when NRMODE value is LINE. Nrspan must be one of the following values: **1,2,3,4,5,6,7,8**.

nr8khz Identifies network reference clock 8KHz output switch value. Nr8khz must be one of the following values:

- **ON** enable 8KHz netref clock output.
- **OFF** disable 8KHz netref clock output.

nrinv Identifies network reference clock output inversion switch value. Nrinv must be one of the following values:

- **ON** enable netref clock output inversion.
- **OFF** disable netref clock output inversion.

nr1 Identifies network reference clock 1 output enable/disable switch value. Nr1 must be one of the following values:

- **ON** enable netref 1 clock output.
- **OFF** disable netref 1 clock output.

nr2 Identifies network reference clock 2 output enable/disable switch value. NR2 must be one of the following values:

- **ON** enable netref 2 clock output.
- **OFF** disable netref 2 clock output.

grp_a..grp_h Identifies span group A to H data rate, each group consists of 4 ctbus spans. Group A is 0-3, group B is 4-7, etc. Grp_x must be one of the following values:

- **OFF** disable (tri-state) spans in a group.
- **2048** span group rate is 2.048 Mbps, 32 timeslots.
- **4096** span group rate is 4.096 Mbps, 64 timeslots.
- **8192** span group rate is 8.192 Mbps, 128 timeslots.

Important*: In order to be able to do the TIMESLOT configuration for CTBUS spans user must first enable corresponding CTBUS span with the GRP_x attribute by setting it to an appropriate value the application demands. Otherwise, the user will get an error stating that the timeslot value is out of the acceptable range.*

ERRORS

<ERROR>::Missing BOARDNM attribute

<ERROR>::Missing INST attribute

<ERROR>::Can not add ctbus MO

<ERROR>::Can not delete ctbus MO

<ERROR>:: Cannot modify FALLBACK attribute

<ERROR>:: Can not modify ctbus MO

<ERROR>:: Can not get ctbus status

<ERROR>::No such a ctbus MO instance

EXAMPLES

MODIFY-CTBUS:HOSTNAME=host-A***,BOARDNM***=artic8260***,INST***=0***,***
REFCLK=C8A***,FBMODE***=C8B***,COMP***=ECTF***,GRP_A***=2048***;***
DISPLAY-CTBUS:;

SAMPLE OUTPUT

```
MML_TH>dis-ctbus:;
----------------------------------------------------------------------------------------------------
HOSTNAME BOARDNM INST REFCLK REFINV FBMODE FBSPAN FB COMP C8A C8B NRMODE NRSPAN NR8KHZ NRINV NRACT NR1
----------------------------------------------------------------------------------------------------
diablo    pmc8260 0   C8A    OFF  DISABLED   -    -  ECTF OFF OFF  INTERNAL -   ON     OFF   -    OFF
diablo    pmc8260 1   C8A    OFF  DISABLED   -    -  ECTF OFF OFF  INTERNAL -   ON     OFF   -    OFF
diablo    pmc8260 2   C8A    OFF  DISABLED   -    -  ECTF OFF OFF  INTERNAL -   ON     OFF   -    OFF
diablo    pmc8260 3   C8A    OFF  DISABLED   -    -  ECTF OFF OFF  INTERNAL -   ON     OFF   -    OFF
diablo  artic8260 0   C8A    OFF  DISABLED   -    -  ECTF OFF OFF  INTERNAL -   ON     OFF   -    OFF
-----------------------------------------------------
NR2 GRP_A GRP_B GRP_C GRP_D GRP_E GRP_F GRP_G GRP_H
-----------------------------------------------------
OFF   2048  OFF   OFF   OFF   OFF   OFF   OFF   OFF
OFF   2048 4096  8192   OFF   OFF   OFF   OFF   OFF
OFF   2048  OFF   OFF   OFF   OFF   OFF   OFF   OFF
OFF   2048  OFF   OFF   OFF   OFF   OFF   OFF   OFF
OFF   2048  OFF   OFF   OFF   OFF   OFF   OFF   OFF
<SUCCESS>:: 5 records found
```


9.5 SCCP MML Commands

9.5.1 Concerned Point Code (CPC)

NAME

CPC — Adds, Deletes, or Displays a Concerned Point Code, or information about a Concerned Point Code.

COMMANDS

ADD — Adds a new Concerned Point Code (CPC) to the subsystem of a Signalling Point Code (SPC) defined in the SCCP network database. The SP and the subsystem must already exist.

ADD-CPC:SPC=spc***,SSN***=ssn***,CPC***=cpc***;***

DELETE — Deletes the CPC from the Subsystem of the SPC defined in the SCCP network database.

DELETE-CPC:SPC=spc***,SSN***=ssn***,CPC***=cpc***;***

DISPLAY — Displays the CPC for the subsystems defined for the SPC from the SCCP network database.

DISPLAY-CPC:[SPC=spc***,SSN***=ssn***,CPC***=cpc***;]***

PARAMETERS

spc — Signalling point code entered as one of the following:

- Zone-Network-SPid (3-8-3) for CCITT networks
  Example:1-222-3
- Network-Cluster-Member (8-8-8) for ANSI networks
  Example:10-20-30
- 5-4-7 bit format for Japanese networks
  Example: 31-25-127

ssn — A subsystem number with a range of 2 to 255.

cpc — Concerned point code entered as one of the following:

- Zone-Network-SPid (3-8-3) for CCITT networks
  Example:1-222-3
- Network-Cluster-Member (8-8-8) for ANSI networks
  Example:10-20-30
- 5-4-7 bit format for Japanese networks
  Example: 31-25-127
- Asterisk (*****) for the entire list

ERRORS

<ERROR>:: Missing SPC parameter.

<ERROR>:: Missing SSN parameter.

<ERROR>:: Missing CPC parameter.

<ERROR>:: Wildcard cannot be used with this command.

<ERROR>:: SP not defined in sccp network.

<ERROR>:: Subsystem not defined for sp.

<ERROR>:: SP cannot be concerned for itself.

<ERROR>:: CPC not defined in sccp network.

<ERROR>:: Subsystem has a mate at cpc.

<ERROR>:: CPC already defined for subsystem.

<ERROR>:: CPC not defined for subsystem.

EXAMPLE

ANSI:	***ADD-CPC:SPC=0-23-255, SSN=4, CPC=224-245-123;***
CCITT:	***ADD-CPC:SPC=3-125-6, SSN=4, CPC=1-2-3;***
ANSI:	***DELETE-CPC:SPC=0-23-255, SSN=4, CPC=224-245-123;***
CCITT:	***DELETE-CPC:SPC=3-125-6, SSN=4, CPC=1-2-3;***
ANSI:	***DISPLAY-CPC:SPC=0-23-255, SSN=4, CPC=224-245-123;***
CCITT:	***DISPLAY-CPC:SPC=3-125-6, SSN=4, CPC=1-2-3;***
Both:	***DISPLAY-CPC:SPC=3-125-6, SSN=4, CPC=*;***
	DISPLAY-CPC:SPC=0-23-255, SSN=4, CPC=*;

SAMPLE OUTPUT

```
------------------------------
SSN   SPC          CPC
------------------------------
254   1-1-1        2-2-2
<SUCCESS>:: 1 records found.
```


9.5.2 Global Title (GT)

NAME

GT — Adds, Deletes, Modifies or Displays a Global Title in the SCCP database.

COMMANDS

ADD — Provisions a global title into the SCCP database.

ADD-GT:GT=gt***,GTIE***=gtie***,TRTYPE***=trtype ***[, NUMPLAN***=numplan***, NATOFADDR***=natofaddrr,***LOADSHARE***=loadshare***], ADDRINFO***=addrinfo***;***

DELETE — Deletes or removes one or more global titles from the SCCP database.

DELETE-GT: GT=gt***,GTIE***=gtie***,TRTYPE***=trtype***, [NATOFADDR***=natofaddr,***NUMPLAN=numplan],ADDRINFO***= addrinfo***;***

MODIFY — Modifies the LOADSHARE parameter of GT.

MODIFY-GT: GT=gt***,LOADSHARE***=loadshare***;***

DISPLAY — Displays one or more global titles in the SCCP database.

DISPLAY-GT:[GT=gt***,GTIE***=gtie***,TRTYPE***=trtype***, [NATOFADDR***=natofaddr,***NUMPLAN=numplan],ADDRINFO***= addrinfo***];***

PARAMETERS

gt — Global title alias (8 bits).

gtie — Global Title Encoding (from 1 to 15).

trtype — Translation type (from 0 to 255).

numplan — Numbering plan (optional for ADD command, default value=1)

natofaddr — Nature of address indicator (optional) replaces ***trtype*** field when gti=1 or 4. (for ITU only)

addrinfo — Addressing information. Enter as a character string (each digit = 1 byte). All global titles which begin with or are exactly equal to this string of digits will be translated to the specified SPC. Global titles with fewer digits will not be translated/deleted/displayed by this entry. To delete/ display all, enter ‘*’.

loadshare — Loadsharing option for global title translation. It is a character string that accepts the following values:

- **ON**
- **OFF**

If load-share is ON, GT related traffic is shared among related gt-entries If set to OFF gt-entries are used in an active/standby manner

ERRORS

<ERROR>:: Missing GTIE parameter.

<ERROR>:: Missing TRTYPE parameter.

<ERROR>:: TRTYPE undefined for GTIE=1 (CCITT).

<ERROR>:: Missing NATOFADDRIND parameter.

<ERROR>:: NATOFADDRIND only defined for GTIE=1 (CCITT).

<ERROR>:: Missing ADDRINFO parameter.

<ERROR>:: ADDRINFO too long.

<ERROR>:: Wildcard cannot be used with this command.

<ERROR>:: address specified already provisioned.

<ERROR>:: trtype specified not provisioned.

<ERROR>:: Invalid ADDRINFO.

<ERROR>:: address specified not provisioned.

<ERROR>:: Only LOADSHARE can be modified.

<ERROR>:: DB inconsistency for GT and GTENTRY.

EXAMPLES:

ADD-GT:GT=GT1,***GTIE***=4,***TRTYPE***=0,***ADDRINFO***=12039251111;

DELETE-GTENTRY:GT=2,***GTIE***=4,***TRTYPE***=253, ***ADDRINFO***=8001234567;

DELETE-GTENTRY:GT=3,***GTIE***=4,***TRTYPE***=253,***ADDRINFO***=*;

MODIFY-GT:GT=GT1,***LOADSHARE***=ON;

DISPLAY-GT:IO=OUTGOING,**GTIE**=4,***TRTYPE***=253, ***ADDRINFO***=8001234567;

DISPLAY-GT:IO=INCOMING,**GTIE**=4,***TRTYPE***=253,***ADDRINFO***=*;

DISPLAY-GT:IO=OUTGOING,***GTIE***=4,***TRTYPE***=253,***ADDRINFO***=8003;

SAMPLE OUTPUT

```
------------------------------------------------------------------
GT        GTIE  TRTYPE NATOFADDRIND      ADDRINFO
------------------------------------------------------------------
1          1     N/A    4             0f3a(HEX)
2          1     N/A    4             8006661234
<SUCCESS>:: 2 records found.
```


9.5.3 Global Title Entry (GTENTRY)

NAME

GTENTRY Adds, Deletes, or Displays a Global Title entry.

COMMANDS

ADD Provisions a global title into the SCCP database.

ADD-GTENTRY:IO=io***,GT***=gt***,SPC***=spc***[,SSN***=ssn***][,NEWGT***=newgt***], [ENTRYTYPE***=entrytype***] [,WILDCARD***=wildcard***][,XLATE_ID***=xlate_id***;***

DELETE Removes a global title entry.

DELETE-GTENTRY:IO=io***,GT***=gt***,[ENTRYTYPE***=entrytype***], [XLATE_ID***=xlate_id***]***

MODIFY Modifies SPC, SSN or NEWGT parameters of global title entry.

MODIFY-GTENTRY:IO=io***,GT***=gt***, ENTRYTYPE***=entrytype ***[,XLATE_ID***=xlate_id***] [,SPC***=spc***][,SSN***=ssn***][,NEWGT***=newgt***];***

DISPLAY Display one or more global title entries.

DISPLAY-GTENTRY:IO=io***,GT***=gt***,[ENTRYTYPE***=entrytype***[,XLATE_ID***=xlate_id***]];***

PARAMETERS

io Incoming or outgoing table. It must be either INCOMING or OUTGOING.

gt Global Title index (1 to 131,072)

entrytype Priority of SCCP entity set in the global translation table. It is a character string that accepts the following values:

- **PRIMARY** (default)
- **SECONDARY**

This attribute is optional in all command types.

xlate_id Defines a unique name for the gtentry. It can only be specified for gt-entries of entrytype **SECONDARY**. When the load-share attribute of the global title is set to **OFF**, gt-entries are used in an active/multi-standby manner and the xlate-id defines the order in which secondary gt-entries are used for translation (when the translation -spc, ssn- in the primary record becomes unavailable, secondary entries are used in alphabetical order to provide another available translation for the gt).

This attribute is optional in all command types and the default value is empty string.

spc	Signalling Point Code entered as one of the following: • Zone-Network-SPid (3-8-3) for CCITT networks Example:1-222-3 • Network-Cluster-Member (8-8-8) for ANSI networks Example:10-20-30 • 5-4-7 bit format for Japanese networks Example: 31-25-127
ssn	Subsystem number, value from **2** to **255** (optional parameter for ADD command).
newgt	New Global Title for translation
wildcard	Indicates whether the entry should be used for wildcard matches: • **YES** • **NO** (default)

ERRORS

<ERROR>:: Missing IO parameter.

<ERROR>:: IO must be either INCOMING or OUTGOING.

<ERROR>:: Wildcard cannot be used with this command.

<ERROR>:: Missing SPC parameter.

<ERROR>:: SP not defined in sccp network.

<ERROR>:: Subsystem not defined for sp.

<ERROR>:: Address specified already provisioned.

<ERROR>:: Primary GTENTRY not defined.

<ERROR>:: Secondary record exists.

<ERROR>:: WILDCARD can NOT be modified.

EXAMPLES:

ADD-GTENTRY:IO=INCOMING***,GT***=GT1***,ENTRYTYPE***=SECONDARY***,***
XLATE_ID=xlate1***,SPC***=1-1-2***,SSN***=254***;***
DELETE-GTENTRY:IO=INCOMING***,GT***=GT1***;***
MODIFY-GTENTRY:IO=INCOMING***,GT***=GT1***,ENTRYTYPE***=SECONDARY***,***
XLATE_ID=xlate1***,SPC***=1-2-3***;***
DISPLAY-GTENTRY:IO=INCOMING***,GT***=GT1***;***

9.5.4 Mate

NAME

MATE Adds, Deletes, or Displays two subsystems of different SPs as mates.

COMMANDS

ADD Mates two subsystems at different SPs in the SCCP network database. Both of the SPs and the subsystems must exist in the database.

ADD-MATE:SPC=spc***,SSN***=ssn***,MSPC***=mspc***,MSSN***=mssn***;***

DELETE Deletes the mate relationship between the SSNs.

DELETE-MATE:SPC=spc***,SSN***=ssn***,MSPC***=mspc***,MSSN***=mssn***;***

DISPLAY Displays the mate of a subsystem defined for the Signalling Point Code (SPC) from the SCCP network database.

DISPLAY-MATE:[SPC=spc***,SSN***=ssn***];***

PARAMETERS

spc Signalling point code entered as one of the following:

- Zone-Network-SPid (3-8-3) for CCITT networks
  Example:1-222-3
- Network-Cluster-Member (8-8-8) for ANSI networks
  Example:10-20-30
- 5-4-7 bit format for Japanese networks
  Example: 31-25-127

ssn A subsystem number with a range of **2** to **255**.

mspc Mate signalling point code entered in the same format as ***spc***.

mssn A subsystem number with a range from **2** to **255**.

ERRORS

<ERROR>:: Missing SPC parameter.

<ERROR>:: Missing SSN parameter.

<ERROR>:: Missing MSPC parameter.

<ERROR>:: Missing MSSN parameter.

<ERROR>:: SP not defined in sccp network.

<ERROR>:: Subsystem not defined for sp.

<ERROR>:: Mate sp not defined in sccp network.

<ERROR>:: Mate subsystem not defined for mate sp.

<ERROR>:: Subsystems of same sp cannot be mated.

<ERROR>:: Subsystem of sp already has a mate.

<ERROR>:: Mate sp defined as own id

<ERROR>:: Subsystem of mate sp already has a mate

<ERROR>:: Subsystems are not mated.

EXAMPLE

ANSI: ***ADD-MATE:SP***=0-23-255***,SSN***=4***,MSPC***=224-245-123***,MSSN***=8***;***

CCITT: ***ADD-MATE:SP***=3-125-6***,SSN***=4***,MSPC***=1-2-3***,MSSN***=8***;***

ANSI: ***DELETE-MATE:SPC***=0-23-255***,SSN***=4***,MSPC***=224-245-123***,***
MSSN =8***;***

CCITT: ***DELETE-MATE:SPC***=3-125-6***,SSN***=4***,MSPC***=123***,MSSN***=8***;***

ANSI: ***DISPLAY-MATE:SPC***=0-23-255***,SSN***=4***;***

CCITT: ***DISPLAY-MATE:SPC***=3-125-6***,SSN***=4***;***

SAMPLE OUTPUT

```
------------------------------------
SSN   SPC         MSSN  MSPC
------------------------------------
111   3-3-3       254   1-1-1
<SUCCESS>:: 1 records found.
```


9.5.5 SCCP

NAME

SCCP Displays or modifies information of the SCCP.

COMMANDS

DISPLAY Displays protocol-specific information of the working SCCP. This includes sp, variant, management address, and timer information.

DISPLAY-SCCP:;

MODIFY Modifies protocol-specific information of the working SCCP. Allowed fields consist of management format and timer values.

MODIFY-SCCP:[PCIND=pcind***][,PROTOCOL***=protocol***]***
[,VARIANT=variant***][,T_IAS***=t_ias***][,T_CONN_EST***=t_conn_est***]***
[,T_IAR=t_iar***][,T_REL***=t_rel***][,T_INT***=t_int***]***
[,T_GUARD=t_guard***][,T_RESET***=t_reset***]***
[,T_SEGMENT=t_segment***][,T_A***=t_a***][,T_D=t_d]***
[,T_CON=t_con***];***

PARAMETERS

PCIND Include point code for SCCP management messages.

- **YES**
- **NO**

PROTOCOL Protocol of the SCCP.

- **DEFAULT**
- **ANSI_92**
- **ANSI_96**
- **ITU_93**
- **ITU_97**
- **REDKNEE**

VARIANT Variant of the SCCP.

- **NONE**
- **ATT**
- **APLUS**
- **SNET**

T_CONN_EST Connection T_CONN_EST timer value.

T_IAS Connection T_IAS timer value

T_IAR Connection T_IAR timer value

T_REL Connection T_REL timer value

T_INT Connection T_INT timer value

T_GUARD	Connection T_GUARD timer value
T_RESET	Connection T_RESET timer value
T_SEGMENT	Segmented messsage T_SEGMENT timer value
T_A	Restriction level T_A timer value (ITU only)
T_D	Restriction level T_D timer value (ITU only)
T_CON	SCCP/subsystem congestion level T_CON timer value (ITU only)

Note: *Timer values are decimal values in milliseconds. For an exact description of the timers, refer to the SCCP specifications.*

EXAMPLE

ANSI ***MODIFY-SCCP:PCIND***=YES***,T_IAS***=300***,T_REL***=700***,***
T_GUARD=500***;***

ITU ***MODIFY-SCCP:PCIND***=YES***,T_IAS***=300***,T_REL***=700***,***
T_GUARD=500***;***

SAMPLE OUTPUT

```
---------------------------------------------------------------------------------------------------------
 SPNO PROTOCOL VARIANT PCIND T_CONN_EST T_IAS T_IAR T_REL T_INT T_GUARD T_RESET T_SEGMENT T_A T_D T_CON
---------------------------------------------------------------------------------------------------------
 0    ANSI_96  NONE    NO    27000      45000 99000 1500  6000  6000    3000    3000      300  5000 5000
<SUCCESS>:: 1 record found
```


9.5.6 SCCP Signalling Point

NAME

SNSP Adds, Deletes, or Displays a Signalling Point in the SCCP network.

COMMANDS

ADD Adds a new signalling point to the SCCP network.The SPC must already be provisioned in the MTP network. When an SPC is added to the SCCP network, the SCCP management subsystem (SSN=1) is automatically created by the SCCP in order to display remote SCCP status in ITU WHITEBOOK networks. When a remote user part (SCCP) is unavailable, only one SST message is sent to the remote SCCP for SSN=1 until the remote SCCP is up. Subsystem SSN=1 can only be displayed by users to monitor the remote SCCP's status. It cannot be modified by users.

ADD-SNSP:SPC=spc;

DELETE Deletes the Signalling Point Code (SPC) from the SCCP network database. This command fails if the SPC does not exist in the database. The SCCP management subsystem (SSN=1) was automatically created by the SCCP when the first SPC was added. Subsystem SSN=1 exists to display remote SCCP status in ITU WHITEBOOK networks. When a remote user part (SCCP) is unavailable, only one SST message is sent to the remote SCCP for SSN=1 until the remote SCCP is up. Subsystem SSN=1 can only be displayed by users to monitor the remote SCCP's status. It cannot be modified by users. When the last SPC is removed from the SCCP network, the management subsystem (SSN=1) is also removed automatically.

DELETE-SNSP:SPC=spc***;***

DISPLAY Displays the Signalling Point Codes (SPC) from the SCCP network database. In ITU WHITEBOOK networks, the management subsystem (SSN=1) always exists and can only be displayed. When subsystem (SSN=1) is PROHIBITED, it means that the remote SCCP user part is unavailable.

DISPLAY-SNSP:[SPC=spc***];***

PARAMETERS

spc Signalling point code entered as one of the following:

- Zone-Network-SPid (3-8-3) for CCITT networks
  Example:1-222-3
- Network-Cluster-Member (8-8-8) for ANSI networks
  Example:10-20-30

- 5-4-7 bit format for Japanese networks
  Example: 31-25-127
- Asterisk (*****) for the entire list

ERRORS

<ERROR>:: Missing SPC parameter.

<ERROR>:: SPC not defined in MTP network.Add routeset first.

<ERROR>:: SP already defined in sccp network.

<ERROR>:: SP defined as own ID.

<ERROR>:: Wildcard cannot be used with this command.

<ERROR>:: SP has defined subsystems.

<ERROR>:: SP not defined in sccp network.

<ERROR>:: SP defined as concerned.

<ERROR>:: Nothing to list.

EXAMPLE

	ADD-SNSP:SPC=0-3-2;
ANSI:	***DELETE-SNSP:SPC=0-23-255;***
CCITT:	***DELETE-SNSP:SPC=3-125-6;***
ANSI:	***DISPLAY-SNSP:SPC=0-23-255;***
CCITT:	***DISPLAY-SNSP:SPC=3-125-6;***
Both:	***DISPLAY-SNSP:SPC=*;***

SAMPLE OUTPUT

```
-------------------------------------------------------------------------
SPC       STATUS       XLATE        CONCERNED          SUBSYSTEMS
-------------------------------------------------------------------------
3-125-6   ACCESSIBLE   PRIMARY      NO                 YES
<SUCCESS>:: 1 records found.
```

Table 9-20: SNSP Display Values

SPC	STATUS	XLATE	CONCERNED	SUBSYSTEMS
See description in Section 9.2.1 on page 9-2	Status of the signalling point: ACCESSIBLE INACCESSIBLE	PRIMARY SECONDARY	Whether the signalling point is a concerned point code.	Whether subsystems are provisioned.

RL	RSL	CLS
SCCP restriction level	SCCP restriction sub-level	SCCP congestion level

9.5.7 Subsystem

NAME

SUBSYS — Adds, Deletes, or Displays a subsystem or subsystem information for a Signalling Point Code (SPC)

COMMANDS

ADD — Adds a new subsystem to a SPC defined in the SCCP network database.

ADD-SUBSYS:SPC=spc***,SSN***=ssn***;***

DELETE — Deletes the subsystem from the SPC defined in the SCCP network database. The command fails if the subsystem or SPC does not exist in the database.

DELETE-SUBSYS:SPC=spc***,SSN***=ssn***;***

DISPLAY — Displays the subsystems defined for the SPC from the SCCP network database.

DISPLAY-SUBSYS:[SPC=spc***,[SSN***=ssn***]];***

PARAMETERS

spc — Signalling point code entered as one of the following:

- Zone-Network-SPid (3-8-3) for CCITT networks
  Example:1-222-3
- Network-Cluster-Member (8-8-8) for ANSI networks
  Example:10-20-30
- 5-4-7 bit format for Japanese networks
  Example: 31-25-127

ssn — Subsystem number entered as one of the following

- number in the range of **2** to **255**
- asterisk (*****) for the entire list

ERRORS

<ERROR>:: Missing SPC parameter.

<ERROR>:: Missing SSN parameter.

<ERROR>:: Wildcard cannot be used with this command.

<ERROR>:: SP not defined in sccp network.

<ERROR>:: Subsystem already defined for sp.

<ERROR>:: SP defined as own id.

<ERROR>:: Subsystem not defined for sp.

<ERROR>:: Subsystem has defined cpc's.

<ERROR>:: Subsystem has a mate.

<ERROR>:: Invalid SSN.

<ERROR>:: Given element not in the sccp database.

EXAMPLE

ANSI:	***ADD-SUBSYS:SPC=0-23-255,SSN=4;***
CCITT:	***ADD-SUBSYS:SPC=3-125-6,SSN=4;***
ANSI:	***DELETE-SUBSYS:SPC=0-23-255,SSN=4;***
CCITT:	***DELETE-SUBSYS:SPC=3-125-6,SSN=4;***
ANSI:	***DISPLAY-SUBSYS:SPC=0-23-255,SSN=4;***
CCITT:	***DISPLAY-SUBSYS:SPC=3-125-6,SSN=4;***
Both:	***DISPLAY-SUBSYS:SPC=1-1-1,SSN=*;***

SAMPLE OUTPUT

```
------------------------------------------------------------------------
SSN       SPC    MSSN   MSPC   SSN_STATUS   XLATE   CONCERNED
------------------------------------------------------------------------
253       1-11-1 0      0-0-0 ALLOWED      PRIMARY      NO
254       1-1-1  0      0-0-0 PROHIBITED   PRIMARY      NO
<SUCCESS>:: 1 records found.
```

Table 9-21: SUBSYS Display Values

SSN, SPC	MSSN	MSPC	SSN_STATUS	XLATE	CONCERNED
See description in synopsis	Mate SSN, if any.	Mate point code, if any.	Status of the SSN: ALLOWED PROHIBITED	Translation: PRIMARY SECONDARY	Whether the signalling point is a concerned point code.

9.5.8 Local Subsystem

NAME

LOCALSUBSYS Displays a local subsystem or local subsystem information for a Signalling Point Code (SPC)

COMMANDS

DISPLAY Displays the local subsystems defined for the SPC from the SCCP network database.

DISPLAY-LOCALSUBSYS:;

PARAMETERS

none

EXAMPLE

ANSI: ***DISPLAY-LOCALSUBSYS:;***

CCITT: ***DISPLAY-LOCALSUBSYS:;***

Table 9-22: LOCALSUBSYS Display Values

SSN, SPC	MSSN	MSPC	SSN_STATUS	XLATE	CONCERNED
See description in synopsis	Mate SSN, if any.	Mate point code, if any.	Status of the SSN: UNEQUIPPED, ALLOWED PROHIBITED	Translation: PRIMARY SECONDARY	Whether the signalling point is a concerned point code.

9.5.9 Connection

NAME

CONNECTION Displays the state of a connection-oriented SCCP connection.

COMMANDS

DISPLAY Displays the state of a connection-oriented SCCP connection.
DISPLAY-CONNECTION:ID=id***;***

PARAMETERS

id Connection ID ranging from **0** up to **16383**, or asterisk (*****) for all. If * entered, then all the connection states but the IDLE ones. are displayed.

ERRORS

<ERROR>:: Missing ID parameter.

<ERROR>:: All connections are in IDLE state.

EXAMPLES

DISPLAY-CONNECTION:ID=126;

DISPLAY-CONNECTION:ID=*;

SAMPLE OUTPUT

```
------------------
ID    STATUS
------------------
126   IDLE
<SUCCESS>:: 1 records found.
```

The connection states can be one of following:

- IDLE
- CONNECTION_PENDING_OUTGOING
- CONNECTION_PENDING_INCOMING
- CONNECTION_PENDING
- WAIT_CONNECTION_CONFIRM
- ACTIVE
- DISCONNECT_PENDING
- DISCONNECT_PENDING_BOTHWAY
- DISCONNECT_PENDING_INCOMING
- DISCONNECT_PENDING_OUTGOING

- MAINTENANCE_BLOCKING
- RESET_OUTGOING
- RESET_INCOMING
- BOTHWAY_RESET
- WAIT_FOR_SENDING_EA_MESSAGE

9.6 ISUP MML Commands

9.6.1 ISUP Circuits

NAME

ISUPCCT — Adds, Deletes, Displays, or Modifies circuits and circuit status information in the ISUP database.

COMMANDS

ADD — Adds one or more circuits to the ISUP database.

ADD-ISUPCCT:PCNO=pcno***,GRPID***=grpid***,***
CCTNUM=cctnum***[,RANGE***=range***];***

DELETE — Deletes a circuit of a circuit group in the ISUP database.

DELETE-ISUPCCT:PCNO=pcno***,GRPID***=grpid***,***
CCTNUM=cctnum***,[RANGE***=range***];***

DISPLAY — Displays the circuit and status information of ISUP circuits.

DISPLAY-ISUPCCT:PCNO=pcno***,GRPID***=grpid***, CCTNUM***=cctnum***;***

MODIFY — Modifies a circuit state to initiate or terminate circuit supervision events in the ISUP database. All events except STOP will cause an ISUP_MML_INITIATED indication to Call Control. The STOP event sends the ISUP_MML_INITIATED_STOP indication.

MODIFY-ISUPCCT:PCNO=pcno***,GRPID***=grpid***,***
CCTNUM=cctnum***,OPERSTATE***=operstate***,RANGE***=range***;***

***Important**: Due to ACK latency or loss, the result of the circuit supervision events may not be observed immediately after executing the command.*

PARAMETERS

pcno — The unique pointcode index number which refers to the Destination Point Code (Signalling Point Code)

grpid — ISUP circuit group ID. It was mapped to a Call Control trunk group ID (*trkgrpid*) by ISUP in the ADD-ISUPCGRP command. This value is used in the CIC field of ISUP messages sent to the network. It is an unsigned integer from **0** to **3039.**

cctnum — The circuit number. It is a number between 0 and the maximum circuits per span defined for the node specified by the pcno

range — Optional field entered as integer value that creates circuits within the specified range, starting from the ***cctnum***.

operstate	Operation state of the circuit. When the state of the circuit is changed, an indication with the appropriate primitive and message type is sent to Call Control. Valid states are: • **BLO:** Initiates a block event on the specified circuit. • **GRS:** Initiates a group reset event, starting with the specified circuit and including the circuits in the range. • **HCGB:** Initiates a hardware group block event, starting with the specified circuit and including the circuits in the range. (In ANSI ISUP, this is a *block with immediate release*.) • **HCGU:** Initiates a hardware group unblock event on the specified circuit and including the circuits in the range. • **MCGB:** Initiates a maintenance group block event, starting with the specified circuit and including the circuits in the range. In ANSI ISUP, this is a *block without release*. • **MCGU:** Initiates a maintenance group unblock event on the specified circuit and including the circuits in the range. • **RSC:** Initiates a reset event on the specified circuit. • **UBL:** Initiates an unblock event on the specified circuit. • **STOP:** Stops all supervision events on a circuit.

ERRORS

<ERROR>::Nonapplicable command.

<ERROR>::Internal database error.

<ERROR>::Missing PCNO parameter.

<ERROR>::Missing GRPID parameter.

<ERROR>::Missing CCTNUM parameter

<ERROR>::PCNO does not exist.

<ERROR>::GRPID does not exist.

<ERROR>::CCTNUM does not exist.

<ERROR>::CCTNUM already exists.

<ERROR>::CCTNUM out of range.

<ERROR>::Nothing to list.

<ERROR>::Call Control not activated.

<ERROR>::Missing OPERSTATE parameter.

<ERROR>::Missing RANGE parameter.

<ERROR>::ISUPCCT is in use.

EXAMPLE

ADD-ISUPCCT:PCNO=1,GRPID=1,CCTNUM=2;
ADD-ISUPCCT:PCNO=1,GRPID=1,CCTNUM=0,RANGE=24;

DELETE-ISUPCCT:PCNO =1,GRPID=5,CCTNUM=2,RANGE=2;
DELETE-ISUPCCT:PCNO =1,GRPID=5,CCTNUM=12;
DISPLAY-ISUPCCT:PCNO=1,GRPID=1,CCTNUM=1;
DISPLAY-ISUPCCT:PCNO=1,GRPID=1,CCTNUM=*;
MODIFY-ISUPCCT:PCNO=1,GRPID=1,CCTNUM=0,OPERSTATE=RSC;
MODIFY-ISUPCCT:PCNO=1,GRPID=1,CCTNUM=0,OPERSTATE=BLO;
MODIFY-ISUPCCT:PCNO=1,GRPID=1,CCTNUM=0,OPERSTATE=UBL;
MODIFY-ISUPCCT:PCNO=1,GRPID=1,CCTNUM=0,OPERSTATE=MCGB, RANGE=7;
MODIFY-ISUPCCT:PCNO=1,GRPID=1,CCTNUM=0,OPERSTATE=HCGB, RANGE=7;
MODIFY-ISUPCCT:PCNO=1,GRPID=1,CCTNUM=0,OPERSTATE=MCGU, RANGE=7;
MODIFY-ISUPCCT:PCNO=1,GRPID=1,CCTNUM=0,OPERSTATE=HCGU, RANGE=7;
MODIFY-ISUPCCT:PCNO=1,GRPID=1,CCTNUM=0,OPERSTATE=STOP;

SAMPLE OUTPUT

Table 9-23: ISUP Circuit Display Report

PCNO	DPC	GRPID (group ID)	CCTNUM (circuit number)	STATUS (circuit status)	MNTCSTATUS (maintenance)	HWDSTATUS (Hardware)	SUSSTATUS (Suspend)	CIC (circuit ID code)
1	5-100-5	1	2	NO-IND	UN-BLK	UN-BLK	NOT-SUS	26 (ansi) 34 (itu)
1	5-100-5	1	5	IN-BUSY	L-BLK	L-BLK	ORG-SUS	29 (ansi) 37 (itu)
1	5-100-5	1	10	IDLE	LR-BLK	LR-BLK	NOT-SUS	34 (ansi) 42 (itu)

***Note**: For ANSI variants, the hardware status of a circuit is not meaningful, so the* **HWDSTATUS** *column is not printed.*

Table 9-24: ISUP Circuit Display Values

Circuit Status	Maintenance/Hardware Status	Suspend Status	Circuit Identification Code
IDLE=No call on circuit IN-BUSY=Incoming call on circuit OUT-BUSY=Outgoing call on circuit NO-IND=A reset message was sent to network, but no acknowledgment (RLG or GRA) arrived. The circuit state is unknown.	UN-BLK=Unblocked L-BLK=Locally blocked R-BLK=Remotely blocked LR-BLK=Locally and remotely blocked	NOT-SUS=Not suspended. Valid for any circuit state except NO-IND. ORG-SUS=Suspended by originator side. Valid for one of the busy circuits. TRM-SUS=Suspended by terminating side. Valid for one of the busy circuits. BOTH-SUS=Suspended by both sides. Valid for one of the busy circuits.	A circuit identification code (CIC) is assigned to each trunk and is known at both ends of the trunk. CICs are unique between two signaling points.
Maintenance status is related to a voluntary setting. Hardware status is related to an automatic setting due to hardware failure.			

9.6.2 ISUP Circuit Group

NAME

ISUPCGRP Adds, Deletes, Displays, or Modifies a circuit group or circuit group information.

COMMANDS

ADD Adds a circuit group to the ISUP database.

ADD-ISUPCGRP:PCNO=pcno***,GRPID***=grpid***,CCTNUM***=cctnum***,*** ***TRNKGRPID***=trnkgrpid***[,SCGA***=scga***];***

DELETE Deletes a circuit group of an ISUP node from the database.

DELETE-ISUPCGRP:PCNO=pcno***,GRPID***=grpid***;***

DISPLAY Displays information on ISUP circuit groups.

DISPLAY-ISUPCGRP:PCNO=pcno***,GRPID***=grpid***;***

MODIFY Modifies a circuit group in the ISUP database.

MODIFY-ISUPCGRP:PCNO=pcno***,GRPID***=grpid***,*** ***CCTNUM***=cctnum***[,TRNKGRPID***=trnkgrpid***][,SCGA***=scga***];***

PARAMETERS

pcno The unique pointcode index number which refers to the Destination Point Code (Signalling Point Code).

grpid Unique identifier of an ISUP circuit group, which is mapped to the Call Control trunk group ID (*trkgrpid*) by ISUP in the ADD-ISUPCGRP command. Call Control sends *trkgrpid* to ISUP, and ISUP sends the *groupid* to the far end ISUP, which maps *groupid* to its own trunk group ID. This value is used in the CIC field of ISUP messages sent to the network. It is an unsigned integer from **0** to **3049**.

cctnum Number of circuits that will be in the ISUP circuit group. It is an unsigned integer between **0** and the maximum circuit groups per span defined for the node by the *pcno*.

trnkgrpid Unique identifier of the Call Control trunk group ID, which is mapped by the ISUP layer to a *destination* and *groupid*. ISUP uses the group ID and the circuit number to calculate the CIC. It is an unsigned integer from **0** to **8191**.

scga For ANSI variants only, Software Carrier Group Alarm (SCGA) protection indication. This parameter is optional. Valid values are:

- **ON**
- **OFF** (*default*)

ERRORS

<ERROR>::Nonapplicable command.

<ERROR>::PCNO does not exist.

<ERROR>::Internal database error.

<ERROR>::Missing PCNO parameter.

<ERROR>::Missing GRPID parameter.

<ERROR>::Missing CCTNUM parameter.

<ERROR>::Missing TRNKGRPID parameter.

<ERROR>::CCTNUM out of range.

<ERROR>::CCTNUM cannot be modified.

<ERROR>::TRNKGRPID out of range.

<ERROR>::TRNKGRPID already exists.

<ERROR>::TRNKGRPID is in use.

<ERROR>::GRPID already exists.

<ERROR>::GRPID out of range.

<ERROR>::GRPID contains CCTs.

<ERROR>::GRPID does not exist.

<ERROR>::Database inconsistency.

<ERROR>::Nothing to list.

EXAMPLE

ADD-ISUPCGRP:PCNO=1,GRPID=1,CCTNUM=2,TRNKGRPID=1,SCGA=ON;
DELETE-ISUPCGRP:PCNO=1,GRPID=5;
DISPLAY-ISUPCGRP:PCNO=1,GRPID=*;
DISPLAY-ISUPCGRP:PCNO=1,GRPID=1;
MODIFY-ISUPCGRP:PCNO=1,GRPID=1,CCTNUM=2,TRNKGRPID=3, SCGA=ON;
MODIFY-ISUPCGRP:PCNO=1,GRPID=1,CCTNUM=2,SCGA=OFF;

SAMPLE OUTPUT

Table 9-25: ISUP Circuit Group Display Report—Pre-Call Control and Maintenance Activation

PCNO	DPC	GRPID (Group ID)	CCTNUM (# of circuits)	TRNKGRPID (Trunk Group)	SCGA	CCNAME	MNTCNAME
1	5-100-5	1	10	2	OFF	UNKNOWN	UNKNOWN
1	5-100-5	2	2	6	ON	UNKNOWN	UNKNOWN
1	5-100-5	10	5	9	ON	UNKNOWN	UNKNOWN

***Important**: SCGA column is printed onlyfor ANSI variants.*

Table 9-26: ISUP Circuit Group Display Report—Post-Call Control and Maintenance Activation

PCNO	DPC	GRPID (group ID)	CCTNUM (# of circuits)	TRNKGRPID (Trunk group)	SCGA	CCNAME	MNTCNAME
1	5-100-5	1	10	2	OFF	CC1	CC1
1	5-100-5	2	2	6	ON	CC1	MNTC1
1	5-100-5	10	5	9	ON	CC5	MNTC5

Important*: CC1, CC5, MNTC1, and MNTC5 are the registration names of the named objects that have been defined as the Call Control or Maintenance Module for ISUP.*

9.6.3 ISUP Signaling Node

NAME

ISUPNODE Adds, Deletes, Displays, or Modifies a node or node information in the ISUP database.

COMMANDS

ADD Adds a signalling node to the ISUP database.

ADD-ISUPNODE:PCNO=pcno***,DPC***=dpc***[,ANMOFF***=anoff***]*** ***[,ACMOFF***=acmoff***][,CRGOFF***=crgoff***][,CFNOFF***=cfnoff***]*** ***[,CICCONTROL***=ciccontrol***][,LOCATION***=location***]*** ***[,MAXCCT***=maxcct***][,FIRSTCIC***=firstcic***];***

DELETE Deletes an ISUP node from the database.

DELETE-ISUPNODE:PCNO=pcno***;***

DISPLAY Displays the status information of the ISUP node. This command displays ISUP office information and its status report in the ISUP network. The ISUP office information code abbreviations for the ISUP office info display are listed below.

Table 9-27: Display Values for ISUP Office Information

Abbreviation.	Meaning	Valid Value
ANMOFF	Answer Message Office	ON-OFF
ACMOFF	Answer Complete Office	ON-OFF
CRGOFF	Charge Office	ON-OFF

DISPLAY-ISUPNODE:PCNO=pcno***;***

MODIFY Modifies a signalling point code in the ISUP database.

MODIFY-ISUPNODE:PCNO=pcno***[,DPC***=dpc***]*** ***[,ANMOFF***=anoff***][,ACMOFF***=acmoff***][,CRGOFF***=crgoff***]*** ***[,CFNOFF***=cfnoff***][,CICCONTROL***=ciccontrol***];***

PARAMETERS

pcno The unique point code index number assigned by the user that refers to the Destination Point Code (Signalling Point Code). It is an unsigned integer from **0** to **2047**.

dpc Destination Point Code (Signalling Point Code) entered as:

- Zone-Network-SPid for CCITT networks
  Example:1-222-3
- Network-Cluster-Member for ANSI networks
  Example:10-20-30

anmoff	Answer Message Office*
acmoff	Address Complete Office*
crgoff	Charge Office*****
cfnoff	CFN Office*****

***Note:** The value of an office information parameter, i.e., **anmoff**, **acmoff**, **crgoff**, or **cfnoff**, is a character string, either **OFF**, to indicate it is not the office type, or **ON** to identify that it is the office type.*

ciccontrol Optional parameter to specify the local exchange's control of circuits (CICs) to the DPC for resolving dual seizures. Values can be:

- **ODD**: Local exchange controls odd CICs
- **EVEN**: Local exchange controls even CICs
- **ALL**: Local exchange controls all CICs
- **NONE**: Local exchange controls none of the CICs
- **DEFAULT**: Exchange with the higher point code controls the even CICs.

newdpc New Signalling Point Code that replaces the old DPC:

- Zone-Network-SPid for CCITT networks
  Example:1-222-3
- Network-Cluster-Member for ANSI networks
  Example:10-20-30

location Loaction field in cause parameter. It is a character string that accepts the following values:

- For ITU:
  - **locuser**
  - **prvnetlocuser**
  - **pubnetlocuser**
  - **transnet pubnetremuser**
  - **prvnetremuser**
  - **locinter**
  - **internatnet beyintworkpnt**
- For Spain:
  - **locuser**
  - **prvnetlocuser**
  - **pubnetlocuser**
  - **transnet pubnetremuser**
  - **prenetremuser**
  - **locinter**
  - **internatnet beyintworkpnt**
  - **pckhndnat**
- For ANSI:
  - **locuser**
  - **prvnetlocuser**
  - **loclocnet**
  - **transnet**

maxcct Maximum number of circuits per circuit group to this node. It is an unsigned integer from **1** to **32**.

firstcic Value of the first CIC to this node. It is an unsigned integer from **0** to **65535**.

cfnoff Makes CFN sending a configurable option. Valid values are:

- ON
- OFF

ERRORS

<ERROR>::Nonapplicable command.

<ERROR>::DPC not defined in MTP network. Add route set first.

<ERROR>::Internal database error.

<ERROR>::DPC already exists.

<ERROR>::PCNO already exists.

<ERROR>::PCNO does not exist.

<ERROR>::Invalid CICCONTROL value.

<ERROR>::Missing DPC parameter.

<ERROR>::Missing PCNO parameter.

<ERROR>::ISUPNODE contains CCTGRPs.

<ERROR>::Nothing to list.

EXAMPLE

ADD-ISUPNODE:PCNO=1,DPC=1-122-1;
ADD-ISUPNODE:PCNO=1,DPC=1-122-1,ANMOFF=ON,CICCONTROL=ODD;
ADD-ISUPNODE:PCNO=1,DPC=1-122-1,ANMOFF=OFF,CRGOFF=ON;
DELETE-ISUPNODE:PCNO=1;
DISPLAY-ISUPNODE:PCNO=6-111-6;
DISPLAY-ISUPNODE:PCNO=*;
MODIFY-ISUPNODE:PCNO=1,DPC=5-100-5;
MODIFY-ISUPNODE:PCNO=1,ANMOFF=ON,CICCONTROL=ODD;
MODIFY-ISUPNODE:PCNO=1,DPC=5-100-5,CICCONTROL=EVEN;

SAMPLE OUTPUT

MML_TH>dis-isupnode:;

Table 9-28: ISUP Node Display Report

PCNO	DPC	CONGESTION STATUS	ACCESSIBILITY STATUS	ANMOFF	ACMOFF	CRGOFF	CICCONTROL
1	5-100-5	0	ACCESSIBLE	ON	OFF	OFF	EVEN
2	6-111-6	0	INACCESSSIBLE	OFF	ON	OFF	ODD
3	2-200-2	1	CONGESTED	ON	OFF	ON	DEFAULT

<SUCCESS>:: 1 record found

9.6.4 ISUP Configuration

NAME

ISUP Displays or Modifies the current ISUP configuration.

COMMANDS

DISPLAY Displays the current ISUP configuration name. The name starts with CF and ends with the SP number, as assigned by the system, e.g. CF0, CF1.

DISPLAY-ISUP:[CFGNAME=cfgname***];***

MODIFY Modifies the current ISUP configuration.

MODIFY-ISUP:CFGNAME=cfgname***[,VARIANT***=variant***]***
[,MNTCIND=mntcind***][,CONGES***=conges***]***
[,RECMODE=recmode***][,AUTORESP***=autoresp***]***
[,EXCHODC=exchodc***][,UPMIND***=upmind***];***

PARAMETERS

cfgname The configuration name, or asterisk (*****) to display all configurations. A configuration name always starts with CF and ends with the signalling point number of the logical node. The configuration name is assigned by the system automatically, e.g., CF0, CF1. The CFGNAME attribute for this command can be omitted.

variant The ISUP variant name. ISUP initially starts with the GENERIC variant. It is a character string that accepts the following values:

- For ANSI: **ANSI92**, **ANSI96**, **BELL**, **DSC**, and **MCI**

Note: *Syntax for BELL MML requires the following:*

If the value is...	Then,
– a decimal	type it directly
– an octal	precede the value with an O’
– a hex	precede the value with an H’

- For ITU: **AUSTRALIA**, **BELGIUM, CHILE**, **CHI24**, **CZECH**, **ETSI97**, **FINLAND**, **FRANCE**, **GERMANY**, **HONGKONG**, **ITALY**, **ITU93**, **ITU97**, **MEXICO**, **NEW_ZEALAND**, **NORWAY**, **PHILIPPINES**, **Q767**, **RUSSIA**, **SINGAPORE**, **SPAIN**, **SWEDEN**, **SWEDENV1, SWITZERLAND, THAILAND**, **TURKEY**, **UAE,** and **UNIPAC**

Note: *The variant change takes more time than other commands. Therefore, it is advised to increase your MML’s TIMEOUT with the MML-CONFIG command before executing this command with a variant change.*

***Important**: When a variant is changed, all the ISUPNODE, ISUPCGRP, ISUPCCT, and ISUPTMR configurations are erased. These objects must be configured again.*

mntcind The Maintenance Indication status. It is a character string that accepts the following values:

- **ON**: Maintenance indications are sent to the Maintenance module.
- **OFF**: Maintenance indications are not sent to the Maintenance module, e.g., Call Control and Maintenance are the same application.
- **GRPINDON**: Only one indication per circuit group will be sent to the maintenance module when circuit group supervision events occur. For ITU releases only.

conges The Congestion Indication status of whether outgoing calls are limited when the destination is congested (according to the ISUP database). Valid for ITU only. It is a character string that accepts the following values:

- **ON**: If a congestion due to a link failure or other circumstance exists, then ISUP rejects outgoing call attempts from Call Control by sending it a RELEASE message in a CALL FAILURE primitive. The cause value of the RELEASE message is *switching equipment congested*.
- **OFF**: ISUP does not reject any outgoing calls because of the congestion at the destination. (*default*)

recmode Software recovery policy mode. It is a character string that accepts the following values:

- **RESCALL –** resume calls
- **RELCALL –** release calls

autoresp Auto response mode. Valid for ITU only. It is a character string that accepts the following values:

- **ON** (*default*)
- **OFF**

exchodc Operator Digital Center (ODC). Valid for Mexico variant only. It is a character string that accepts the following values:

- **ON**
- **OFF** (*default*)

upmind An indicator of whether ISUP should send / receive User Part Test (UPT) and User Part Available (UPA) messages. This is valid for ITU only. It is a character string that accepts the following values:

- **Off** - ISUP does not send a UPT message to the network, and does not start the T4 timer. It sends ISUP_UP_INACCESSIBLE indication to the Call Control. It also does not acknowledge with UPA message on the receipt of a UPT message from the network.

- **On** - ISUP sends an UPT message to the network and also starts the T4 timer. It sends ISUP_UP_INACCESSIBLE indication to the Call Control. It also acknowledges with an UPA message when it receives a UPT message from the network. (*default*)

ERRORS

<ERROR>::Nonapplicable command.

<ERROR>::CFGNAME does not exist.

<ERROR>::Internal database error.

<ERROR>::Nothing to list.

<ERROR>::Invalid VARIANT value.

<ERROR>::MNTCIND value is denied in ANSI variants.

<ERROR>::Feature not enabled.

EXAMPLE

DISPLAY-ISUP:CFGNAME=CF0;
DISPLAY-ISUP:CFGNAME=*;
DISPLAY-ISUP:;
MODIFY-ISUP:CFGNAME=CF0,VARIANT=BELL,MNTCIND=ON;
MODIFY-ISUP:MNTCIND=OFF,RECMODE=RELCALL;
MODIFY-ISUP:VARIANT=TURKEY;
MODIFY-ISUP:CONGES=OFF;

SAMPLE OUTPUT

MML_TH>dis-isup::

Table 9-29: ISUP Configuration Display Report (ANSI)

CFGNAME	VARIANT	MNTCIND	RECMODE	MBGIND	AUTORES P
CF0	BELL	ON	RESCALL	OFF	ON

<SUCCESS>:: 1 record found

Table 9-30: ISUP Configuration Display Report (ITU)

CFGNAME	VARIANT	MNTCIND	CONGESTION	RECMODE
CF0	TURKEY	OFF	ON	RELCALL

9.6.5 ISUP Timer

NAME

ISUPTMR Displays or Modifies ISUP protocol timer values.

COMMANDS

DISPLAY Displays the ISUP protocol timer values.

DISPLAY-ISUPTMR:TIMERID=tmrid***;***

MODIFY Modifies the protocol timer values in ISUP database.

MODIFY-ISUPTMR:TIMERID=tmrid***,VALUE***=value***;***

PARAMETERS

tmrid Timer identifier; it is an unsigned integer from **1** to ***n*** for a specific timer, where *n* is the upper limit of timer numbers for the protocol being used. The asterisk (*****) wild-card character can also be used to display all timers. The timers, displayed in milliseconds, are shown in Table 9-31 on page 9-124, and defined in Table 9-32.

value Timer value in milliseconds. Timer value can be between **10** milliseconds and **24** hours.

ERRORS

<ERROR>::Nonapplicable command.

<ERROR>::TIMERID out of range.

<ERROR>::Missing TIMERID parameter.

<ERROR>::Nothing to list.

<ERROR>::Internal database error.

<ERROR>::Missing VALUE parameter.

<ERROR>::VALUE out of range.

EXAMPLE

DISPLAY-ISUPTMR:TIMERID=2;
DISPLAY-ISUPTMR:TIMERID=*;
MODIFY-ISUPTMR:TIMERID=2,VALUE=20000;

SAMPLE OUTPUT

Table 9-31: ISUP Timer Display Report

TIMERID	VALUE
1	20000
2	15000
⋮	
n	50000

Table 9-32: ISUP Timers

Timer ID	Description	ANSI/ITU Usage	Default in ITU (msec)	Default in ANSI (msec)
1	First RLC timer	BOTH	15000	15000
2	Suspend/Resume timer	ITU	180000	-
3	Overload	ITU	120000	-
4	User Part Test	ITU	300000	-
5	Second RLC timer	BOTH	300000	60000
6	RES timer (network)	BOTH	120000	30000
7	ACM timer	BOTH	30000	30000
8	COT timer	BOTH	15000	15000
9	ANM timer	BOTH	180000	180000
10	Unused	-	-	-
11	Unused	-	-	-
12	First BLA timer	BOTH	15000	15000
13	Second BLA timer	BOTH	300000	60000
14	First UBA timer	BOTH	15000	15000
15	Second UBA timer	BOTH	300000	60000
16	First RSC response timer	BOTH	15000	15000
17	Second RSC response	BOTH	300000	60000
18	First CGBA timer	BOTH	15000	15000
19	Second CGBA timer	BOTH	300000	60000
WB: White Book				

Table 9-32: ISUP Timers (Continued)

Timer ID	Description	ANSI/ITU Usage	Default in ITU (msec)	Default in ANSI (msec)
20	First CGUA timer	BOTH	15000	15000
21	Second CGUA timer	BOTH	300000	60000
22	First GRA timer	BOTH	15000	15000
23	Second GRA timer	BOTH	300000	60000
24	Continuity tone timer	BOTH	1000	1000
25	First CCR timing	BOTH	10000	2000
26	CCR response timer	BOTH	180000	180000
27	CCR receive timer	BOTH	240000	240000
28	CQR timer	BOTH	10000	10000
29	First congestion	ITU	300	-
30	First congestion indication	ITU	10000	-
31	Unused	-	-	-
32	Unused	-	-	-
33	Information Request	ANSI	-	15000
34	CCR timer	ANSI	-	15000
35	Unused	-	-	-
36	CCR response (Q767/WB)	ITU	15000	-
37	Unused	-	-	-
38	TACC	ANSI	-	5000
39	TCCR	ANSI	-	2000
40	TCCRr	ANSI	-	20000
41	TCGB	ANSI	-	5000
42	TCRA	ANSI	-	10000
43	TCRM	ANSI	-	4000
44	TCVT	ANSI	-	10000
45	TEXMd	ANSI	-	15000
46	TGRS	ANSI	-	5000
47	THGA	ANSI	-	300000
48	TSCGA	ANSI	-	120000
WB: White Book				

Table 9-32: ISUP Timers (Continued)

Timer ID	Description	ANSI/ITU Usage	Default in ITU (msec)	Default in ANSI (msec)
49	TSCGAd	ANSI	-	60000
51	Trunk Offering Timer	CZECH	180000	-
	Tcc for FINLAND	FINLAND	10000	
	Tcalloffer for MEXICO	MEXICO	360000	
52	Tchg1 for FINLAND	FINLAND	3000	-
53	Tchg2 for FINLAND	FINLAND	3000	-
54	Tchg3 for FINLAND	FINLAND	60000	-
55	Tchg4 for FINLAND	FINLAND	3000	-
56	Tx for FINLAND	FINLAND	60000	-
WB: White Book				

***Important**: When the maintenance messages listed in Table 9-33 are sent to the network, two associated timers are started - the second timer, and then the first timer. The response message indicated in the table is expected within these time periods. When the first timer (a 15-second timer) expires, the message is resent and the first timer is restarted. The message is resent each time the first timer expires until the second timer (a 1-minute timer) expires. When the second timer expires, the first timer is stopped, the maintenance system is alerted, and the second timer is restarted. The system begins to send the message in one-minute intervals.*

***Important**: Distributed7 ISUP starts the second timer before the first timer. However, the fourth expiration of the first timer can occur before the second timer expires because 1 minute is exactly four times 15 seconds and the operating system's timer is not a high precision one. If the fourth expiration of the first timer occurs exactly when the second timer expires, then two messages can be sent at the same time. To avoid this situation, the timers can be configured. For example, the first timer can be set to 15.1 seconds or the second timer can be set to 59.9 seconds.*

Table 9-33: ISUP Related Timers to Modify

First Timer 15 sec.	Second Timer 1 min.	Related Messages
T12	T13	BLO sent - BLA expected

Table 9-33: ISUP Related Timers to Modify (Continued)

First Timer 15 sec.	Second Timer 1 min.	Related Messages
T14	T15	UBL sent - UBA expected
T16	T17	RSC sent - RLC expected
T18	T19	CGB sent - CGBA expected
T20	T21	CGU sent - CGUA expected
T22	T23	GRS sent - GRA expected

9.7 System MML Commands

9.7.1 Host

NAME

HOST Adds, Deletes, Displays, or Modifies a host instance.

COMMANDS

ADD Add a new host instance.

ADD-HOST:HOSTNAME=hostname***,RMTHOST***=rmthost ***[,ALIAS***=alias***,RMTHOSTTYP***=rmthosttyp***,CONF***=conf***];***

DELETE Delete a host instance.

DELETE-HOST:HOSTNAME=hostname***,RMTHOST***=rmthost***;***

DISPLAY Display a specific host instance or all instances.

DISPLAY-HOST:[HOSTNAME=hostname***,RMTHOST***=rmthost***];***

MODIFY Modify a host instance information.

MODIFY-HOST:HOSTNAME=hostname***,RMTHOST***=rmthost ***[,RMTHOSTTYP***=rmthosttyp***,CONF***=conf***];***

Note: *MML commands to disconnect host-B from host-A cannot be entered from host-A as follows:*
MML_TH> MODIFY-HOST: HOSTNAME=host-B, RMTHOST=host-A, CONF=OFF;

This command fails with the following error string:
<ERROR>:: MODIFY-HOST operation must be performed on local hosts

This means that the same command should have been issued from host-B. Only the connection from the host-A side can be disconnected from host-A, for example:
MML_TH> MODIFY-HOST:HOSTNAME=host-A,RMTHOST=host-B,CONF=OFF;
MML_TH> MODIFY-HOST:HOSTNAME=host-A,RMTHOST=host-D,CONF=OFF;

PARAMETERS

hostname Name of host, entered as a 15-character alphanumeric label.

rmthost Name of remote host.

rmtalias Remote system dual name if it is a multi-homed host.

rmthosttyp Remote system host type. Enter as either of the following values:

- **AMGR**: Distributed7-type system
- **OTHER**: other than Distributed7-type system

conf Determines the configuration of the host. Valid values are:

- **ON**: establish this connection

- **OFF**: wait for some other configurations

ERRORS

<ERROR>:: MO does not exist
<ERROR>:: Hostname is not defined in the network
<ERROR>:: No such an instance
<ERROR>:: Can not add host in standalone mode
<ERROR>:: Missing HOSTNAME parameter
<ERROR>:: Missing RMTHOST parameter
<ERROR>:: Local hostname can not be used as remote or alias
<ERROR>:: DUALHOST is not configured in NTWK MO
<ERROR>:: Alias host is not in the same network of DUALHOST
<ERROR>:: TCPCON entry does not exist
<ERROR>:: Missing CONF parameter
<ERROR>:: Same CONF value for this entry
<ERROR>:: CONF attribute of the entry is ON

EXAMPLES

ADD-HOST:HOSTNAME=uranium-1,RMTHOST=silicon-1,
ALIAS=silicon-2,CONF=ON;
DISPLAY-HOST:;
MODIFY-HOST:HOSTNAME=uranium-1,RMTHOST=silicon-1,CONF=OFF;

SAMPLE OUTPUT

```
MML_TH>DISPLAY-HOST:;
-------------------------------------------------------------------------
HOSTNAME  RMTHOST    ALIAS        RMTHOSTTYP  CONF
-------------------------------------------------------------------------
uranium-1 silicon-1  silicon-     AMG         ON
<SUCCESS>:: 1 record found
```

9.7.2 Stored Alarm

NAME

STRDALM Deletes, Displays alarms set in the system.

COMMANDS

DELETE Clears an alarm that is currently *set* in the system. Alarms that have occurred which are of type SET_ALARM are tracked by the alarm process until they are cleared. An alarm can be cleared when the CLR_ALARM type alarm that is associated with the alarm occurs or when this command is executed with the alarm specified individually or as part of a group. The current list of set alarms can be displayed with the DISPLAY-STRDALM command.

DELETE-STRDALM:[HOST_NAME=host_name***,] GROUP***=group
[, MODULE=module***, TYPE***=type***][,LAST_OCC***=last_occ***]***
[,FIRST_OCC=first_occ***][,NUM_OF_OCCUR***=num_of_occur***]***
[,ALM_TEXT=alm_text***];***

DISPLAY Displays the current alarms that are *set* in the system. Alarms that have occurred which are of type SET_ALARM are tracked by the alarm process until they are cleared. These alarms are the ones shown in the output of this command. EVENT and CLR_ALARM alarms are never shown in this output. All alarms are displayed to the console as they occur, depending on the display settings.

DISPLAY-STRDALM:[HOST_NAME=host_name***,] GROUP***=group
[, MODULE=module***, TYPE***=type***][,LAST_OCC***=last_occ***]***
[,FIRST_OCC=first_occ***][,NUM_OF_OCCUR***=num_of_occur***]***
[,ALM_TEXT=alm_text***];***

PARAMETERS

host_name Name of the host whose alarms are to be deleted or displayed. Default is the local host.

group Alarm group name. It can be asterisk **(*)** for all groups, or one of the following:

- DKM - Distributed kernel memory
- ISUP - ISUP management
- ISUPMOD - ISUP management, distributed
- MTPL1 - MTP Level 1
- MTPL2 - MTP Level 2
- APM - application process management
- NIMOD - connection management
- OMAP - operation, maintenance, and administration part

- SCCP - service connection control part management
- SPM - signalling point (SP) management
- TCAP - TCAP driver
- TCMOD - TCAP over TCP/IP connection management
- TRMOD - translation module
- UPM - user part management, such as MTP Level 3
- ETMOD - ethernet test module
- PMON - passive monitor
- PMMOD - passive monitor module

module Module number (middle two digits) of the alarm or alarms to be deleted or displayed for the specified ***GROUP***.

type Last two digits of the alarm number to be deleted or displayed for the specified ***GROUP*** and ***MODULE***.

last_occ Date and time of the last occurrence of the alarm number(s). It can be used with or without the other attributes. The timestamp must be specified in the format: **hh:mm:ss@MM/DD/YY.**

- **hh** (hour) can be from 01 to 24
- **mm** (minutes) can be from 00 to 59
- **ss** (seconds) can be from 00 to 59
- **MM** (month) can be from 01 to 12
- **DD** (day) must be a valid two-digit number for the given month
- **YY** (year) can be from 00 to 99

first_occ Date and time of the first occurrence of the alarm number(s). It can be used with or without the other attributes. The timestamp must be specified (as above) in the format: **hh:mm:ss@MM/DD/YY.**

num_of_occur Number of occurrences of the alarm number(s).

alm_text Text associated with the alarm number(s).

ERRORS

<ERROR>::Nothing to list

<ERROR>::Cannot update configuration

<ERROR>::Invalid subtype for this MO

EXAMPLES

DISPLAY-STRDALM:GROUP=*;
DISPLAY-STRDALM:GROUP=SPM,MODULE=2,TYPE=3;
DISPLAY-STRDALM:LAST_OCC=11:15:32@02/15/97;

Table 9-34: STRDALM Display Values

HOSTNAME GROUP MODULE TYPE	INST	SEVERITY	FIRST_OCC	LAST_OCC	NUM_OF_OCCUR	ALM_TEXT
See description under command syntax		Severity of the alarm: INFO MINOR MAJOR CRITICAL FATAL	Date and time of first occurrence of the alarm number in hh:mm:ss@ MM/DD/YY format.	Date and time of last occurrence of the alarm number in hh:mm:ss@ MM/DD/YY format.	Number of occurrences of alarm number.	Text associated with the alarm number.

9.7.3 Alarm

NAME

ALARM Displays or Modifies the alarm configuration.

COMMANDS

DISPLAY Displays the current alarm configuration, including the console threshold, external console threshold, current alarm log file name, and current configuration files. It also displays the lower severity thresholds of alarms to be received by console and user for each alarm group.

The output of this command (see Sample Output) displays the current settings for the console output and for the default severity-level threshold settings of alarm output to the external console and the external user. The full path and file names are shown for the alarm group definition file, the alarm configuration file, and the current log file for alarms. The alarm log file holds all the generated alarms and can be used for an extensive examination of the alarms that have occurred in the system.

The output also displays all the alarm group IDs (GR#), in decimal, from the alarm group definition file. The GRP_NAME column specifies the name associated with the group ID. The USR-THR column shows the minimum threshold of alarm severity that will be displayed to the user interface for that group (e.g. all those alarms from that group that are above INFO in severity). The CONS-THR column shows the minimum threshold of alarm severity that will be displayed to the console, if it is enabled.

More information on alarm groups, alarm severities, and alarm descriptions can be found in the *Distributed7 Installation and Maintenance Manual*.

DISPLAY-ALARM:;

MODIFY Changes the configuration of the alarm managed object to have new alarm-display characteristics, to be the global alarm process in the distributed system, or to be updated with configuration files. Regardless of display settings, alarms are always logged to a file in the ***$EBSHOME/access/AlarmLogs*** directory.

MODIFY-ALARM:[HOSTNAME=hostname***,DISPLAY***=display***,***
CONS_THRS=cons_thrs***,USER_THRS***=user_thrs***,***
REPEAT=repeat***, GLOBAL***=global***, UPDATE***=update***];***

PARAMETERS

hostname Name of the host to be modified, entered as an 15-character alphanumeric label.

display	Indicator of display status. Values can be ON, to display alarms to the console, or OFF, to turn the display of alarms off. At start-up, it is set to ON by default. The alarm process must be running in order to display alarms.
cons_thrs	Minimum severity level of alarms to be displayed to the console when used with CONS_THRS or to a user-defined external alarm interface function when used with USER_THRS. Valid values for severity are: • NONE • INFO (*default*) • MINOR • MAJOR • CRITICAL • FATAL
user_thrs	Minimum severity level of alarms to be displayed to a user-defined external alarm interface function. Valid values for severity are: • NONE • INFO (*default*) • MINOR • MAJOR • CRITICAL • FATAL
repeat	Counter threshold for displaying a repeated alarm. If an incoming alarm is exactly the same as the immediately preceding alarm, then the counter is increased incrementally but the alarm is not displayed unless the counter equals the setting specified in this command. Default is 3, i.e., for the default, if four identical alarms are received, then only two are displayed. The counter is reset to 0 when the threshold is reached. The value is ignored if the DISPLAY is set to OFF.
global	Indicator of an attempt to become the global alarm process. Value must be **ON** to be global.
update	If the update indicator is set to ON, ***alarm*** daemon will re-read alarm group definition and alarm text files and update its in-memory copy. Subsequently, it will set the update indicator to OFF.

ERRORS

<ERROR>::Network is down.

<ERROR>::No such process.

<ERROR>::Illegal address.

<ERROR>::Parameter value is out of range.

<ERROR>::Cannot update configuration.

EXAMPLE

MODIFY-ALARM:DISPLAY=ON;
MODIFY-ALARM:DISPLAY=OFF;
MODIFY-ALARM:CONS_THRS=MINOR;
MODIFY-ALARM:USER_THRS=MINOR;
MODIFY-ALARM:REPEAT=5;
MODIFY-ALARM:GLOBAL=ON;
MODIFY-ALARM:UPDATE=ON;

SAMPLE OUTPUT

Alarm output with DISPLAY on and REPEAT at a threshold of 3:

```
ALARM $880703 $00000000 $00000000 LVL: Info
MTP: Link Set LS-ls0 available
         - - - 3 alarms came - - -
   Fri Mar 27 14:31:03 1996
   Last alarm repeated 3 times more
         - - - 3 alarms came - - -
   Fri Mar 27 14:31:42 1996
   Last alarm repeated 3 times more
         - - - 3 alarms came - - -
   Fri Mar 27 14:32:22 1996
   Last alarm repeated 3 times more
```

9.7.4 Alarm Event

NAME

ALMEVENT Displays alarm events for all hosts or a specified host.

Note: *Application programs can use the* ***alm_notify()*** *function to express interest in a particular set of alarm events that can occur while operating under the Distributed7 (a.k.a. AccessMANAGER) environment, and specify what action should take place if and when one of the pending alarm events occurs on local, a remote, or all hosts.*

COMMANDS

DISPLAY Displays alarm events

DISPLAY-ALMEVENT:HOSTNAME=hostname**,**
REQ_HOSTNAME=req_hostname**,*****GROUP***=group**,**
MODULE=module**,*****TYPE***=type**,**
THRESHOLD=threshold***;***

ADD Priviliged operation – can only be used by MO server alarm daemon

DELETE Priviliged operation – can only be used by MO server alarm daemon

PARAMETERS

hostname Name of the host whose alarm event information is requested. Wildcard is not allowed. If HOSTNAME and other arguments are not specified, alarm event information for all hosts displayed.

req_hostname Name of the host from which an application wants to be notified when a specified alarm occurs; wildcard is allowed.

group ID of the alarm group; wildcard is allowed.

module Alarm module ID in alarm group; wildcard is allowed.

type Alarm type in alarm group and module. Wildcard is allowed.

threshold Severity of the alarm. If the specified alarm occurs with this severity, then the application is notified. Valid values for severity are, in ascending order:

- INFO
- MINOR
- MAJOR
- CRITICAL
- FATAL

Note: *Wildcard is not allowed for the* ***threshold*** *parameter.*

ERRORS

<ERROR>::Nothing to list

<ERROR>::Cannot update configuration

<ERROR>::ALMEVENT instance does not exist

<ERROR>::Invalid subtype for this MO

EXAMPLES

DISPLAY-ALMEVENT:;
DISPLAY-ALMEVENT:GROUP=84,MODULE=3,TYPE=*;

SAMPLE OUTPUT

```
MML_TH>DISPLAY-ALMEVENT:;
--------------------------------------------------------------------------
HOSTNAME   REQ_HOSTNAME   GROUP   MODULE   TYPE   THRESHOLD
--------------------------------------------------------------------------
galaxya-1      galaxya-1   131   2     2     MINOR
galaxya-1      galaxya-1   131   2     2     CRITICAL
galaxya-1      galaxya-1   132   3     2     INFO
galaxya-1      galaxya-1   132   3     2     MAJOR
galaxya-1      galaxya-1   131   2     2     INFO
galaxya-1      galaxya-1   132   3     2     FATAL
galaxya-1      galaxya-1   131   2     2     MAJOR
galaxya-1      galaxya-1   131   2     2     FATAL
galaxya-1      galaxya-1   132   3     2     MINOR
galaxya-1      galaxya-1   132   3     2     CRITICAL
<SUCCESS>:: 10 records found
```

9.7.5 Alarm Group

NAME

ALMGRP Displays or Modifies the alarm group settings.

COMMANDS

DISPLAY Displays the threshold settings for an alarm group.

DISPLAY-ALMGRP:GROUP=group***;***

MODIFY Changes the severity thresholds for the display of alarms for individual alarm groups.

MODIFY-ALMGRP:GROUP=group***, [CONS_THRS***=cons_thrs***, USER_THRS***=user_thrs***];***

PARAMETERS

groupname Alarm group name.

- When used with ***cons_thrs***, specifies the minimum severity threshold for displaying alarms of that group to the console.
- When used with ***user_thrs***, specifies the minimum severity threshold for sending alarms of that group to the user-defined external alarm interface function.

Group name can be asterisk (*****) for all groups or one of the following:

- DKM - Distributed kernel memory
- ISUP - ISUP management
- ISUPMOD - ISUP management - distributed
- MTPL1 - MTP Level 1
- MTPL2 - MTP Level 2
- APM - application process management
- NIMOD - connection management
- OMAP - operation, maintenance, and administration part
- SCCP - service connection control part management
- SPM - signalling point (SP) management
- TCAP - TCAP driver
- TCMOD - TCAP over TCP/IP connection management
- TRMOD - translation module
- UPM - user part management, such as MTP Level 3
- ETMOD - ethernet test module
- PMON - passive monitor
- PMMOD - passive monitor module

severity Minimum severity level of alarms to be displayed to the console when used with ***cons_thrs*** or to a user-defined external alarm interface function when used with ***user_thrs***. Valid values for severity are:

- INFO
- MINOR
- MAJOR
- CRITICAL
- FATAL

Note: *If the* ***cons_thrs*** *or* ***user_thrs*** *settings in MODIFY-ALARM are higher than the settings for individual groups, then the alarms with the lower severity specified by this command will not be displayed.*

ERRORS

<ERROR>::Network is down.

<ERROR>::No such process.

<ERROR>::Illegal address.

<ERROR>::Parameter value is out of range.

<ERROR>::Cannot update configuration.

EXAMPLE

MODIFY-ALMGRP:GROUP=SPM,CONS_THRS=MAJOR;
MODIFY-ALMGRP:GROUP=SPM,USER_THRS=MAJOR;

SAMPLE OUTPUT

```
MML_TH>DISPLAY-ALMGRP:;
------------------------------------------------------------------------
HOSTNAME    GROUP   CONS_THRS   USER_THRS   NUM_OF_ALMS
------------------------------------------------------------------------
   uranium-1 OMAP    NONE        NONE        0
   uranium-1 TRMOD   NONE        NONE        0
   uranium-1 ETMOD   NONE        NONE        0
   uranium-1 DKM     NONE        NONE        0
   uranium-1 UPM     NONE        NONE        0
   uranium-1 NIMOD   NONE        NONE        0
   uranium-1 SPM     NONE        NONE        3
<SUCCESS>:: 7 records found
```

9.7.6 Configuration

NAME

MMLCONF Displays or modifies the MML session configuration values.

COMMANDS

DISPLAY Displays the configuration values for the MML session. All settings can be displayed by just entering the command without parameters. Individual settings are displayed by specifying an attribute name.

DISPLAY-MMLCONF:[LOG=log***,TIMEOUT***=timeout***];***

MODIFY Configures the MML process with settings for command logging and response timeout. The configuration values are stored and maintained as the settings for MML on a particular signalling point even after the current session ends. Subsequent MML sessions, on the same signalling point, use the same configuration until it is modified. MML sessions for different signalling points use their own configurations, e.g., *mml 0* and *mml 1* may have different configurations.

DISPLAY-MMLCONF:[LOG=log***,TIMEOUT***=timeout***];***

PARAMETERS

log State of MML command logging. Values are:

- ON (*default*)
- OFF

While the LOG option is on, MML logs all commands that are issued, except DISPLAYs, into the file,
`$EBSHOME/access/RUN<sp#>/backup/MMLcmnds.current`.
The user name, user ID, and time of execution are included with the command in the log that is written to this file. The LOG option can be turned off and on at any time with this command.

timeout Value of timeout for communications with processes; it is an unsigned integer from **0** and **240000** milliseconds. The default is 15000 milliseconds.

When MML sends a message to the daemon processes, e.g., *upmd*, *isupd*, etc., it waits for acknowledgments until the TIMEOUT setting has expired. If an acknowledgment is not received, then MML displays an SPM error message.

ERRORS

<ERROR>::[LOG] allowed values: { ON, OFF }

<ERROR>::Parameter [TIMEOUT] must be in range 1 - 240000

<ERROR>::Cannot update configuration

EXAMPLE

MODIFY-MMLCONF:;
MODIFY-MMLCONF:LOG;
MODIFY-MMLCONF:LOG=OFF;
MODIFY-MMLCONF:TIMEOUT=3000;

SAMPLE OUTPUT

```
MML_TH>DISPLAY-MMLCONF:;
-------------------------------------------------------
CONFNAME TIMEOUT     LOG
-------------------------------------------------------
CFG0     15000       ON
<SUCCESS>:: 1 record found
```

9.7.7 Network

NAME

NTWK — Displays or Modifies the operation mode of hosts in the distributed network.

COMMANDS

DISPLAY — Displays the operation mode of hosts in the distributed network.

DISPLAY-NTWK:[HOSTNAME=hostname***];***

MODIFY — Configures the Distributed7 system as stand-alone or part of a distributed network.

MODIFY-NTWK:HOSTNAME=hostname***[,MODE=***mode***,***
CLOCKSYNC=clocksync***,FREQUENCY=***frequency***,***
DUALHOST=dualhost***,NETMASK1=***netmask1***,***
NETMASK2=netmask2***];***

PARAMETERS

hostname — Name of host, entered as a 15-character alphanumeric label.

mode — Mode of operation for the host, entered as:

- **STNDLN** stand-alone mode
- **DSTRBTD** distributed mode (*default*)

clocksync — Specifies whether the network clock synchronization logic is available or not. Settings are

- **ON**
- **OFF**

frequency — Specifies, in milliseconds, how often to check the system clock on all hosts if CLOCKSYNC is **ON**. It is an integer from 0 to 10000. The default value is 0 if running in the stand alone mode, or 1000 in the distributed mode. THe range for the distributed mode is 60 to 10000.

dualhost — Specifies the alternate host name, if any, of the local host on a secondary, i.e., dual, network. If dual-LAN is not in use, then this field should contain the local host name specified in the HOSTNAME field.

mask1 — 32-bit mask specified in hex format that is used to extract the network ID on the primary network. The default values are initialized on the basis of class type associated with the corresponding network, as follows:

- Class A – 7f000000
- Class B – 3fff0000
- Class C – 1fffff00

mask2	32-bit mask specified in hex format that is used to extract the network ID on the secondary network, if any. The default values are the same as those for ***mask1***, indicated above.

ERRORS

<ERROR>:: MO does not exist

<ERROR>:: Hostname is not defined in the network

<ERROR>:: No such an instance

<ERROR>:: NTWK MO cannot be modified - HOST entries exist

<ERROR>:: Product is not configured as distributed

EXAMPLES

DISPLAY-NTWK:;
MODIFY-NTWK:HOSTNAME=uranium-1,DUALHOST=uranium-2;

9.7.8 TCP/IP Connections

NAME

TCPCON Displays or Modifies the TCP/IP connection information.

COMMANDS

DISPLAY Displays information on TCP/IP connections.

DISPLAY-TCPCON: [HOSTNAME=hostname***,RMTHOST***=rmthost***];***

MODIFY Configures TCP/IP connections.

MODIFY-TCPCON:HOSTNAME=hostname***,***
RMTHOST=rmthost***[,MODE***=mode***,SERVICE***=service***,***
PROTO=proto***,MODULES***=modules***,HBEAT***=hbeat***,***
FREQU=frequ***,MAXTRIES***=maxtries***,ACT_EST***=act_est***,***
ACT_RMV=act_rmv***, HB_LOSS***=hb_loss***];***

PARAMETERS

hostname Name of host, entered as a 15-character alphanumeric label.

rmthost Name of remote host.

mode TCP/IP connection type, entered as one of the following values:

- **AUTO:** auto mode means system sets the mode, it's the default value.
- **MASTER:** master mode means local host always tries to establish the connection.
- **SLAVE:** slave mode means local host always waits for connection requests.

service Internet service name can only be entered as **NETDBASE**, which is the default.

proto Identifies transportation layer protocol, entered as **TCP**, which is the default.

modules Specifies the ordered list of STREAMS modules that should be pushed over this TCP/IP connection. (***nimod*** is the only one.)

hbeat Indicates whether the heartbeat mechanism should be activated/ deactivated for this TCP/IP connection. The settings are:

- **ON:** activate heartbeat
- **OFF:** deactivate heartbeat

frequ Specifies, in milliseconds, how often to check the TCP/IP connections if HBEAT is **ON**. It is integer from 0 to 1000. The default value is 0 if the HBEAT parameter is OFF, and 1000 if the HBEAT parameter is ON.

maxtries	Specifies the number of consecutive times that should be tried to establish the specified TCP/IP connection before giving up hope. A value of *-1* means try forever.
act_est	Specifies the action to take when the connection is established. Values are: • **IGNORE** • **INFORM** (*default*)
act_rmv	Specifies the action to take when the connection is broken. Values are: • **IGNORE** • **INFORM** (*default*)
hb_loss	Specifies the action to take when a remote host fails to respond to a heartbeat request. Values are: • **NOACTION** • **SYNCDATA**

ERRORS

<ERROR>:: MO does not exist

<ERROR>:: Hostname is not defined in the network

<ERROR>:: No such an instance

<ERROR>:: Missing HOSTNAME parameter

<ERROR>:: Missing RMTHOST parameter

<ERROR>:: TCPCON entry does not exist

<ERROR>:: CONF attribute of the entry is ON

EXAMPLES

DISPLAY-TCPCON:;
MODIFY-TCPCON:HOSTNAME=uranium-1,RMTHOST=silicon-1,MODE=AUTO,HBEAT=ON;

9.7.9 EXIT

NAME

EXIT

COMMANDS

EXIT	Exits from the MML session. ***EXIT:;***

ERRORS

None

EXAMPLE

EXIT:;

9.7.10 Help

NAME

HELP

COMMANDS

HELP	Switches to the help mode and displays names of commands or a man page for the specific command entered. ***HELP:CMD=command_name;*** ***HELP:;***

PARAMETERS

command_name name of the command for which help is needed.

ERRORS

None

EXAMPLE

HELP:;
HELP:CMD=ADD-LINK***;***

9.7.11 SET-LOG

NAME

SET-LOG

COMMANDS

SET-LOG Turns the generation of log messages for a specified process ON or OFF. After the *ON* command is processed, each message into or out of the named object or SS7 object is duplicated and saved to the LOG process. Logging must be disabled by entering the command with OFF.

SET-LOG:TO=to***,[NAME***=name***,][SPID***=spid***,UPID***=upid***,***
SSN=ssn***,INST***=inst***],LOG***=log***;***

PARAMETERS

object	Type of process, entered as **NMDOBJ** or **SS7OBJ.**
name	Character string name of any registered named object. This parameter is mandatory when logging to any **NMDOBJ** type of object.
spid	Signalling point number, is an integer value and a mandatory parameter for any SS7 object.
upid	User Part **(UP)** ID is an integer value as defined in SS7 protocol (0 for MTP, 3 for SCCP). This parameter must be supplied to log messages to an SS7 object.
ssn	Subsystem number of the SS7 object, or an asterisk (*****) for all SSNs.
inst	Instance number of the subsystem number of the SS7 object.
log	Indicator to turn logging ON or OFF.

ERRORS

<ERROR>::Network is down.

<ERROR>::No such process.

<ERROR>::Illegal address.

<ERROR>::Parameter value is out of range.

<ERROR>::MML syntax error.

EXAMPLE

SET-LOG:TO=NMDOBJ,NAME=XYZ,LOG=ON;
SET-LOG:TO=SS7OBJ,SP=0,UP=3,SSN=254,INST=2,LOG=ON;

9.8 Passive Monitor MML Commands

9.8.1 Passive Monitor Link (PMLINK)

NAME

PMLINK — Adds, modifies, deletes a passive monitor link, or displays information about a passive monitor link.

COMMANDS

ADD — Adds a passive monitor link to an SS7board. The SS7board managed object must be added as a passive monitor board (by setting the PM option to ON) and must be configured ON prior to this command.

ADD-PMLINK:HOSTNAME=hostname***,BOARDNM***=boardnm***,***
INST=inst***,PORT***=port***;***

Note*: Proper timeslot and clocking settings must be done before adding a passive monitor link. Please see MML command descriptions in Section 9.4.6, SS7 Board (SS7BOARD) and Section 9.4.17, Time Slot (TIMESLOT).*

MODIFY — Modifies the administrative state of a passive monitor link.

MODIFY-PMLINK:HOSTNAME=hostname***,BOARDNM***=boardnm***,***
INST=inst***,PORT***=port***,ADMINSTAT***=adminstat***;***

DELETE — Deletes a passive monitor link.

DELETE-PMLINK:HOSTNAME=hostname***,BOARDNM***=boardnm***,***
INST=inst***,PORT***=port***;***

DISPLAY — Displays passive monitor link attributes.

DISPLAY-PMLINK:[HOSTNAME=hostname***[,BOARDNM***=brdnm
[,INST=inst***[,PORT***=port***];***

PARAMETERS

boardnm — Board type, entered as one of the following values:

- **pci3xpq** common name for 24-port PCI bus boards (pci37xpq)
- **pci3xapq** common name for 24-port PCI bus boards (pci37xapq)
- **pmc8260** common name for 64-port CompactPCI bus boards (pmc8260)
- **artic8260** common name for 64-port CompactPCI bus boards (artic1000 and artic2000)

inst — Physical instance number of the board. It is an unsigned integer from 0 to 8.

port	Port number of the link, entered as a numerical value. Valid range depends on the board type: • pci3xpq 0 to 23 • pci3xapq 0 to 23 • pmc8260 0 to 63 • artic8260 0 to 63
adminstat	Passive monitor link administrative state. Possible values are: • DEACTIVATE : Link is deactivated. MSUs are not received on this link. • ACTIVATE: Link is activated and ready to capture and pass on MSUs.
operstat	Passive monitor link operational state. Possible values are: • SHUTOFF : Link is deactivated by the passive monitor layer due to an internal resource outage. • INACTIVE : Link is deactivated due to an application's request. • IDLE : Link is activated but not receiving any MSUs. Either the connection with the SS7 link is lost or the SS7 link is not operational. • OOS: Out of Service, link is active but only receiving LSSU SIOS signals. • ALIGNING: Link is active but only receiving LSSU-SIO, SIN, SIE. Link is in alignment period. • INSERVICE: Link is active and receiving FISU or MSU signals (SS7 link is in service). • PROC-OUT: Link is active and receiving LSSUs because the sending side of the SS7 link has processor outage.
linkf	Number of link failures.
rxframes	Number of received MSUs.
rxoctets	Number of received MSU octets
rsu_e	Number of received signal units in error
d_rxl	Number of discarded signal units due to invalid HDLC length
d_bo	Number of discarded signal units due to receiver buffer overflow

ERRORS

<ERROR>:: Can not add pmlink

<ERROR>:: Can not perform operation - board not configured

<ERROR>:: Can not delete pmlink

<ERROR>:: MO exists

<ERROR>:: Generic error
<ERROR>:: Can not get pmlink attributes
<ERROR>:: Missing ADMINSTAT attribute
<ERROR>:: Missing BOARDNM attribute
<ERROR>:: Missing INST attribute
<ERROR>:: Missing PORTNUM attribute
<ERROR>:: Missing parameter
<ERROR>:: No such a SS7BOARD MO instance
<ERROR>:: No such a pmlink instance
<ERROR>:: No such a PORT MO instance
<ERROR>:: Nothing to list
<ERROR>:: Link information can not be retrieved
<ERROR>:: Invalid stream no is retrieved from spmd
<ERROR>:: mtpl2 MO operation failed
<ERROR>:: LINK is activated.
<ERROR>:: Nothing to list.

EXAMPLES

ADD-PMLINK:HOSTNAME=chicago,BOARDNM=pci3xapq,INST=0,PORT=0;
MOD-PMLINK:HOSTNAME=chicago,BOARDNM=pci3xapq,INST=0,PORT=0, ADMINSTAT=ACTIVATE;
MOD-PMLINK:HOSTNAME=chicago,BOARDNM=pci3xapq,INST=0,PORT=0, ADMINSTAT=DEACTIVATE;
DISPLAY-PMLINK:HOSTNAME=chicago,BOARDNM=pci3xapq,INST=0;
DISPLAY-PMLINK:HOSTNAME=chicago,BOARDNM=pci3xapq;
DISPLAY-PMLINK:HOSTNAME=chicago;
DISPLAY-PMLINK:;

SAMPLE OUTPUT

```
MML_TH>dis-pmlink:;

---------------------------------------------------------------------------------------------------------
        HOSTNAME    BOARDNM INST PORT   ADMINSTAT   OPERSTAT LINKF RXFRAMES RXOCTETS RSU_E D_RXL   D_BO
---------------------------------------------------------------------------------------------------------
          chicago   pci3xpq    0    0    ACTIVATE  INSERVICE     0        0        0     0     0      0
          chicago   pci3xpq    0    1    ACTIVATE  INSERVICE     0        0        0     0     0      0
          chicago   pci3xpq    0    2    ACTIVATE  INSERVICE     0        0        0     0     0      0
          chicago   pci3xpq    0    3    ACTIVATE  INSERVICE     0        0        0     0     0      0
          chicago   pci3xpq    0    4    ACTIVATE  INSERVICE     0        0        0     0     0      0
          chicago   pci3xpq    0    5    ACTIVATE  INSERVICE     0        0        0     0     0      0
          chicago   pci3xpq    0    6    ACTIVATE  INSERVICE     0        0        0     0     0      0
          chicago   pci3xpq    0    7    ACTIVATE  INSERVICE     0        0        0     0     0      0
```

Chapter 10: Users Guide for Virtual SS7 Connections

10.1 Introduction

The Virtual SS7 connections environment is comprised of a pseudo driver—the Virtual Board (***vbrd***) driver—a utility program (***vb_config***) to configure the driver, a daemon (***vb_bridge***) to enable remote host operations, and some ksh scripts to make configuring the driver easier.

The Virtual Board is a pseudo driver that simulates a physical 32 port SS7 board. The ***vbrd*** driver is designed to be used with this Distributed7 release. Since this release introduces distribution, during use and especially testing of the release, a tremendous need has arisen for physical SS7 cards. This has been the driving force behind virtual SS7 connections. With this driver, even though there is no SS7 card in hand, a virtual SS7 board can be added and configured, links can be added and activated, and finally SS7 message signalling units (MSU's) can be transmitted and received. No instrumentation is required in the Distributed7 software for using the virtual board. The ability of virtual board relies on the fact that ***vbrd*** interacts with Distributed7 the same way as it does with the real SS7 cards. As an example, although there is no virtual board level-2 code, ***vbrd*** sends a positive acknowledgment to a download request, or any other Level-2 management messages that require acknowledgment, from Distributed7.

Link operations such as addition, deletion, and activation are handled by some internal port tables. Whenever the driver receives a message, it parses the message and takes an appropriate action, if any. The driver keeps track of the links added with MML commands, and their states (modified from MML). It sends Distributed7 some state transition information, (if the two ends of a link are started, ***vbrd*** sends link_in_service to Distributed7). As a result, links can be activated within a few seconds.

Actual message transmission in the Virtual SS7 connection environment is achieved by the ***vbrd*** driver. The ***vbrd*** driver keeps track of its 32 links, such as which port is connected to which port. When a message is received from Port A, and Port A is connected to Port B, ***vbrd*** just overwrites certain fields within the message (*linkid*) and sends it to the Distributed7 software. In this way although there is not a physical wiring between the two ports on same host, messages are transferred.

This method works fine if the link is connected between two ports on the same host. What if we want to connect ports on different hosts? Here we introduce the ***vb_bridge*** daemon to handle link connection on different hosts. The ***vb_bridge*** daemon creates a data transmission bridge between a user-specified pair of hosts. Before performing any operations on a remote host, a data bridge must be created between the two hosts. The ***vbrd*** driver on host A knows that if a message is received from Port 1 that is connected to Port 2 on host B it should relay the message to the data bridge that is maintained by the ***vb_bridge*** daemon. And respectively, when a driver receives a message from a ***vb_bridge*** daemon, it knows that this is a message received from a remote host, and handles the message accordingly.

10.2 An Architectural Overview

Virtual SS7 connections comprise a new device driver (called the ***vbrd*** driver) and a set of command line utilities.

The ***vbrd*** driver is a pseudo driver that has been designed to simulate the actions of a physical SS7 board. It supports up to a total of 32 device connections of which the very first one is reserved for communication with the Distributed7 software. The remaining connections are for the ***vbrd*** command line utilities.

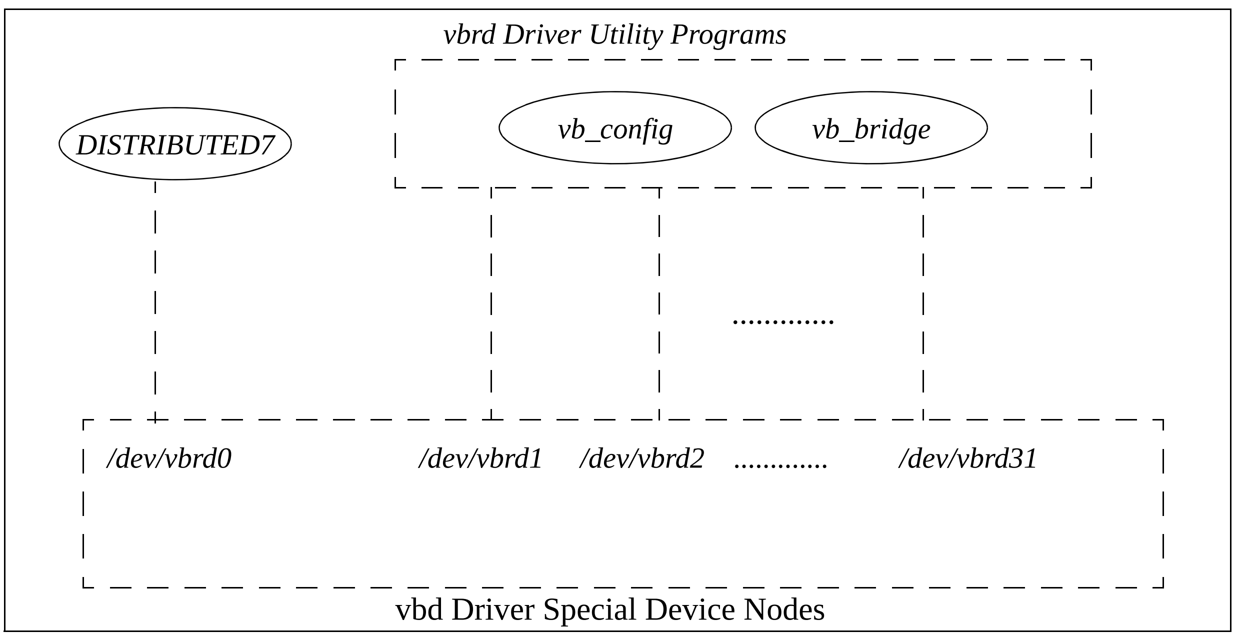


Figure 10-1: Architecture of the Virtual Board

10.3 Driver Installation/Removal

Before using any of the virtual connections functionality, the ***vbrd*** driver must be installed. When the Distributed7 product is installed (***ebs_modinstall***), ***vbrd*** driver is installed automatically. If the user wants to use Virtual Connection functions, then ***vbrd*** must be installed separately with *vbrd_install* script.

For installation and removal of this driver, the scripts *vbrd_install* and *vbrd_remove*, respectively, must be used. These scripts are under ***$EBSHOME/access/install*** directory.

To install the ***vbrd*** driver:

cd $EBSHOME/access/install

./vbrd_install

To remove the ***vbrd*** driver:

cd $EBSHOME/access/install

./vbrd_remove

Driver installation involves creation of special device files used by the ***vbrd*** driver.

vbrd_remove removes all special device nodes used by the ***vbrd*** driver. At the end of successful removal operation, *vbrd_remove* script does not return any error, and all device files created with *vbrd_install* script are deleted.

EBSHOME environment variable must be set for *vbrd_install* and *vbrd_remove* scripts. If EBSHOME variable is not set, the following error message is returned:

```
EBSHOME environment variable is not set!
```

If during driver removal operation, there is an open stream, (a process is already using the driver), *vbrd_remove* will fail with the following error message.

```
Cannot unload module: vbrd
Will be unloaded upon reboot.
```

This process can be a ***vbrd*** command line utility, or the Distributed7 itself. To enable removal of the ***vbrd*** driver, an ***ebs_stop*** must be issued, and all host connections must be reset. If driver is already removed, the following message will be printed:

```
Driver (vbrd) not installed.
```

10.4 New Concepts

The virtual SS7 board driver simulates a 32 port SS7 board. The functions of Virtual Connections are designed to realize this fact. Every operation on an actual SS7 card has a corresponding counterpart in a Virtual SS7 connections environment. New functions may be interrelated with a previously performed operation.

The Distributed7 interface to the virtual board is exactly the same as that of a physical SS7 board. Distributed7 does not differentiate whether the configured SS7 card is actually a physical board or a virtual one.

In following sections of this document, new concepts are introduced.

10.4.1 Port Connection

In traditional methods of communication, wiring must be done between two end points (between two ports). Connecting two ports of an SS7 board is the very first step of link alignment, message transmission, and reception. If the ports are not connected physically (wiring not done between ports), the MML command for link alignment fails, and messages can not be transferred in between. Whereas, cabling or wiring is not required in virtual connections.

In a Virtual SS7 connections environment, wiring between ports is considered as "port connection" and it is achieved with an option supplied to the *vb_config* utility. When a previously made connection is no longer needed, it should be removed, again using the *vb_config* utility.

Detailed description of how ports are connected in Virtual SS7 Connections is described in Section 10.5.2.1.

10.4.2 Host Connection

Ports to be connected can be on same host as well as on two different hosts. Since Virtual SS7 connections simulate a real wiring, there are no limitations for ***vbrd*** either. Here a need arises for some methods to transmit and receive data between two hosts since wiring between ports is lacking. The messages in ***vbrd*** environment travel over a software oriented data transmission bridge that is created previously. The creation of the bridge between a pair of hosts is called host connection operation. This operation is achieved by starting a daemon that runs on both of the hosts until the connection is no longer needed.

10.5 Virtual SS7 Connections Utilities

10.5.1 vb_bridge Utility

As previously mentioned, to perform any operation on remote host, (e.g. connecting a port to another port on a remote host or SS7 message transmission to a remote host), a data transmit/receive bridge must be available between the two hosts. This bridge is bi-directional and is setup and maintained by the ***vb_bridge*** daemon. The only function of ***vb_bridge*** daemon is to create a data bridge and transfer and receive messages between the two ends of the data bridge in both directions.

The ***vb_bridge*** creates a data bridge between two hosts, and informs ***vbrd*** driver that the connection to the remote host is up. In this way, the driver handles remote operations to the remote host easily.

The virtual board enables remote operations transparently both to the user and the Distributed7 with the ***vb_bridge*** daemon. When a physical link connection is considered, ***vb_bridge*** program basically enables port connections on different host machines. Starting ***vb_bridge*** daemon is the very first step of connecting ports between remote hosts (if this daemon is not running port connection between remote hosts is not allowed).

Every message destined to a remote host is transferred over the same data bridge.

Before using ***vb_bridge*** utility, ***vbrd*** driver must be installed.

The ***vb_bridge*** daemon starts itself on the remote host through 'ksh'. The user will need to have ksh permissions to remote host, if ***vb_bridge*** daemon is to be started.

When starting, the ***vb_bridge*** daemon full pathname shall be given. This is required since the ***vb_bridge*** daemon uses ksh, and remote shell PATH variable might be different from what is on the local host.

As an example, if a Virtual SS7 connections environment is to be established between hosts A, B and C, the following commands must be issued on one of these three hosts.

```
$EBSHOME/access/bin/vb_bridge A B
$EBSHOME/access/bin/vb_bridge A C
$EBSHOME/access/bin/vb_bridge B C
```

After those commands, any remote operations between hosts A, B and C can be done.

10.5.2 vb_config Utility

vb_config program is the user interface program for the ***vbrd*** driver.

Basically, configuring ***vbrd*** driver includes the following operations:

- connecting ports
- disconnecting ports
- listing port information
- listing host connections information
- resetting all host and port connections

10.5.2.1 Setting Up Port Connection

The concept of port connection is described in Section 10.4.1. In Virtual SS7 connections environment, a port is defined with a hostname and a port number information. Hence the parameters are in the form of:

vb_config -m host1:port1 host2:port2

The scope of the port connection setup operation is not limited to the local host. A port on local host can be connected to another port on remote host. Or a port on a remote host can be connected to another port on yet another remote host.

As an example, lets say we are issuing ***vb_config*** commands on Host A, and we have Host B and Host C in the virtual SS7 connections environment. All of the following examples are valid.

Example 1: Port Connection between ports of Host A. No host connection is required.

vb_config -m A:<port1> A:<port2>

Example 2: Port Connection between ports of Host A and Host B. Host connection between Hosts A and B is required.

vb_config -m A:<port1> B:<port2>

vb_config -m B:<port1> A:<port2>

Example 3: Port Connection between ports of Host B. Host connection between Hosts A and B is required.

vb_config -m B:<port1> B:<port2>

Example 4: Port Connection between ports of Host B and Host C. Host connections between Hosts A-B, A-C, and B-C is required.

vb_config -m B:<port1> C:<port2>

Above examples illustrate that in Virtual SS7 Connections environment an operation can be initiated through any host, as long as remote host connections are available to the remote hosts.

When ports on different hosts are connected, operation can succeed on one host, but the remote host may fail to connect its port. For this reason, an acknowledgment mechanism is implemented for the port connection procedure. Local host sends the port connection request to the remote host and waits for a response. Response can be one of positive or negative acknowledgment. If the remote host succeeds to connect its port, and performs its task successfully, it sends the originator of the port connection request a positive acknowledgment, and the originator connects its port too, and the command returns. If the remote host fails to connect its port for some reasons, it sends a negative acknowledgment to the originator, and originator returns an error to the user.

Due to ***vbrd*** driver limitations, at any time only one port connection can be performed. Hence until the port connection operation is completed, (positive or negative acknowledgment is received or acknowledgment timer expires) and ***vb_config*** program returns, another port connection request is rejected.

At the end of a successful port connection operation, no error message is returned. If operation fails, an error message indicating the error is printed on the console.

Possible error cases are:

- One of the local ports specified is already connected.
- Another port connection operation is in progress.
- There is no connection to the remote host specified.
- Operation on remote end failed for some reason.

At any time, the following command issued on a related host can be used to list the connected port information.

vb_config -l con

10.5.2.2 Breaking Port Connections

This operation exactly simulates removing wiring or cables between two ports. Hence this operation updates port tables to omit the port connection, and informs Distributed7 about the new state (if the links were aligned, after this operation, they become unavailable).

A link is composed of a connection between two ports. To break the link connection, only one port information is enough. The driver removes the port connection on the local host, and sends the related host a break connection command, so that the remote end will also reset its port.

The syntax is as follows:

vb_config -b <hostanme>:<portnumber>

No acknowledgment mechanism is available for this procedure since there is no error case. Even if the port to be disconnected is already disconnected, no error is returned.

As an example, lets say we are issuing the ***vb_config*** commands on host A, and Hosts A, B and C are connected to each other. All listed examples are valid:

Example 1: A:<port1> and A:<port2> are connected. No host connection required.

vb_config -b A:<port1>

vb_config -b A:<port2>

Example 2: A:<port1> and B:<port2> are connected. Hosts A and B must be connected.

vb_config -b A:<port1>

vb_config -b B:<port2>

Example 3: B:<port1> and B:<port2> are connected. Hosts A and B must be connected.

vb_config -b B:<port1>

vb_config -b B:<port2>

Example 4: B:<port1> and C:<port2> are connected. There must be host connection between each pair of hosts A, B and C.

vb_config -b B:<port1>

vb_config -b C:<port2>

If a remote operation is required in break port connection operation, but that host is not connected, an error message is printed on console, and ***vb_config*** program returns with an error.

10.5.2.3 Retrieving Port Related Information

The ***-l*** option is used to retrieve port information on the host that command is issued. User can not retrieve the port information of a remote operation. The syntax is:

vb_config -l con | dis | all

The argument of ***-l*** option is the filter defined for the port list output. The list can be filtered to include only the connected ports (***con***), or the disconnected ports (***dis***) or no filter that is all of the ports (***all***).

If the ***all*** option is used, all of the 32 ports can be displayed. If a port is not connected, it is marked as DISCONNECTED. If the port is connected, all the information related with that port is output.

As an example, let us say a link is created between two ports, and port information is listed with the following commands:

vb_config -m sparc4a:0 sparc4a:1

vb_config -l con

The `vb_config -l` command output looks something like:

```
---------------------------------------------------------------------------
LocalRemoteRemote  SP   Link   Link   Prot   Local  Remote  lpo   rpo    MSU
Port  Host   Port       Set    No            State  State               Count
---------------------------------------------------------------------------
0   sparc4a   1    -1   -1     -1     -1     OS     OS      0     0      0
1   sparc4a   0    -1   -1     -1     -1     OS     OS      0     0      0
```

For the detailed explanation of individual fields, refer to the manual page, or the User Command section of this manual, for the ***vb_config*** utility.

Note that all SP and link related information has been set to invalid values initially.

When links are added with the following MML commands:

MML_TH>add-link:link=l11,lset=ls1,hostname=sparc4a,BOARDNM=vbrd,INST=0, port=0,slc=0,priority=0;

MML_TH>add-link:link=l11,lset=ls1,hostname=sparc4a,BOARDNM=vbrd,INST=0, port=1,slc=0,priority=0;

the `vb_config -l` command output looks something like:

```
---------------------------------------------------------------------------
LocalRemoteRemote  SP   Link   Link   Prot   Local  Remote  lpo   rpo    MSU
Port    HostPort        Set    No            State  State               Count
```

```
---------------------------------------------------------------------------------
0   sparc4a  1      0    0      0      ansi   OS     OS      0      0      0
1   sparc4a  0      1    0      0      ansi   OS     OS      0      0      0
```

Again note that SP and link related fields now reflect correct values.

When link on port 0 is started with the following MML command:

MML_TH>modify-linkstat:link=l11,status=SET_ACT;

the **`vb_config -l`** command output looks like:

```
---------------------------------------------------------------------------------
LocalRemoteRemote  SP   Link   Link   Prot   Local  Remote   lpo  rpo    MSU
Port  Host   Port       Set    No            State  State                Count
---------------------------------------------------------------------------------
0   sparc4a   1     0    0      0      ansi   StartedOS        0    0      0
1   sparc4a   0     1    0      0      ansi   OS     Started   0    0      0
```

Note that '*`Local State`*' of Port 0, and '*`Remote state`*' of Port 1 are changed to *`Started`*.

When link on port 1 is also started with the following MML command:

MML_TH>modify-linkstat:link=l11,status=SET_ACT;

the **`vb_config -l`** command output looks like:

```
---------------------------------------------------------------------------
LocalRemoteRemote  SP   Link   Link   Prot   Local  Remote  lpo   rpo    MSU
Port   HostPort         Set    No            State  State                Count
---------------------------------------------------------------------------
0   sparc4a  1     0     0      0      ansi   IS     IS      0      0      0
1   sparc4a  0     1     0      0      ansi   IS     IS      0      0      0
```

Note that when Port 1 is also started, both port states are changed to *`IS`* (in-service).

When link on port 1 receives local processor outage, as a result of following MML command:

MML_TH>modify-linkstat:link=l11,status=SET_LPO;

the **`vb_config -l`** command output looks like:

```
---------------------------------------------------------------------------
LocalRemoteRemote SP    Link   Link   Prot   Local  Remote lpo    rpo    MSU
Port   HostPort         Set    No            State  State                Count
---------------------------------------------------------------------------
0   sparc4a  1     0     0      0      ansi   IS     IS     0      1      0
1   sparc4a  0     1     0      0      ansi   IS     IS     1      0      0
```

Note that `lpo` field for Port 1 and consequently `rpo` field for Port 0 are set.

10.5.2.4 Displaying Connections to Remote Hosts

The ***-h*** option is used to list the hosts that the local host has a connection with. The hosts are connected with the ***vb_bridge*** daemon, but the hostnames of the connected hosts are retrieved with ***vb_config*** utility program.

The output is a list of hostnames. If there is no host connection, nothing is printed.

10.5.2.5 Resetting Environment

The ***-r*** option is used to reset all the host and port connections in the Virtual SS7 connections environment. At the end of this call, all the port connections are cleared. The **`vb_config -l`** `all` command displays all ports as DISCONNECTED. Also all the ***vb_bridge*** daemons are terminated.

10.5.3 Shell Scripts

The purpose of ***vbrd*** ksh scripts is to create a simulation of physical wiring. That is, if a wiring connection between two ports is not affected from a reboot on one of the hosts, the virtual board environment shall not be affected either. But since all the units of Virtual SS7 connections are software oriented, and are obviously affected by a reboot, a new method is implemented. This method is based on keeping all the applied commands required to setup the environment in a snapshot file called the ***vb_startup*** file.

If a machine is re-booted or goes down for any reason, after all the machines in the environment are up again, the ***vb_startup*** file is executed. The ***vb_startup*** file is an executable shell script, which is executed when a host machine crashes, or for other reasons, the exact state of virtual link and host connection must be re-established.

All of the shell scripts perform an operation, and update the ***vb_startup*** file to reflect the current state of the virtual board environment. All ***vbrd*** shell scripts update the ***vb_startup*** file according to what they have changed. This guarantees that, if the user always uses ***vbrd*** shell scripts, the ***vb_startup*** file will be up-to-date, and is the snapshot of the environment.

A typical ***vbrd*** shell script has these two elements:

- perform required action
- update ***vb_startup*** file.

There is a script for most of operations in ***vbrd*** environment. The list is as follows:

Table 10-1: *vbrd* scripts

Script Name	Brief Description
vb_connhosts	Establishes connections between each pair of hosts in parameter list
vb_addhost	Adds a host to the established ***vbrd*** environment
vb_connports	Defines a link between two ports
vb_discport	Breaks a link connection
vb_lports	List port information on local host
vb_lhosts	List host connections information for the local host
vb_reset	Resets port and host connections on all hosts in the virtual board environment

10.5.3.1 vb_connhosts

vb_connhosts is a ksh script that updates ***vb_startup*** file and issues ***vb_bridge*** command(s). There must be at least two hosts in the parameter list. Host names in the parameter list are pinged to eliminate unreachable hosts. Host name duplication is also checked. When a valid host list, that can be pinged, is retrieved the list is printed to a host_list file. If a host can not be pinged, nothing is done for that host, as if it were not given as parameter.

Host connection process between multiple hosts requires starting the ***vb_bridge*** program between each combination of hosts. As an example, if the given command is:

vb_connhosts A B C

the ***vb_connhosts*** script, initiates ***vb_bridge*** program three times with the following parameters:

$EBSHOME/access/bin/vb_bridge A B
$EBSHOME/access/bin/vb_bridge A C
$EBSHOME/access/bin/vb_bridge B C

The executed ***vb_bridge*** commands are appended to end of the ***vb_startup*** file, so that the ***vb_startup*** file will be up-to-date.

A warning is printed in following cases:

- a host in parameter list is unreachable (ping failed).
- a host name is repeated in parameter list.
- any of ***vb_bridge*** commands fails (on console).

An error is returned in following cases:

- less then two parameters are given.

10.5.3.2 vb_addhost

When it is necessary to add a host to the established virtual board environment, the ***vb_addhost*** script is used. The ***vb_addhost*** script assumes that there are some hosts in the environment, and a new host is being added.

Since the host connection is done between two hosts, ***vb_addhost*** invokes the ***vb_bridge*** executable for each pair of hostnames and hosts in environment.

As an example, let us say the host connection is available between three hosts A, B, and C, and we are adding host D to the environment. The ***vb_addhost*** script then executes the following commands:

$EBSHOME/access/bin/vb_bridge A D
$EBSHOME/access/bin/vb_bridge B D
$EBSHOME/access/bin/vb_bridge C D

The ***vb_startup*** file is also updated, to include the executed commands.

An error is returned in following cases:

- hostname can not be pinged.
- hostname is already connected.

10.5.3.3 vb_connports

vb_connports is a ksh script that updates the ***vb_startup*** file and issues following command:

vb_config -m host1:port1 host2:port2

The ***vb_connports*** script is used for defining a link between two ports. Each port information consists of a hostname and a port number (range 0-31). There is no restriction for host names, except for being a valid system name in same network. Both host1 and host2 can be local, or both can be remote or a mixture. Even if the port is local, its host information must still be supplied. The connection only needs to be established once, as it is a bi-directional connection.

There must be at least two ports in the parameter list. Host names in the parameter list are pinged. If at least one host is unreachable, an error is returned, and the ***vb_startup*** file is not updated. The successfully executed `vb_connports host1:port1 host2:port2` command is appended to end of the ***vb_startup*** file, so that the ***vb_startup*** file will be up-to-date.

If one of the ports is on a remote host, make sure the ***vb_bridge*** daemon has been started for the remote host. The ***vb_connports*** script will not return until all remote operations are complete.

An error is received for following cases (printed on the console):

- first port is already in connected state.
- second port is already in connected state.
- a remote operation is needed, but ***vb_bridge*** daemon is not running.
- remote end is too late to acknowledge.

10.5.3.4 vb_discport

vb_discport is a ksh script that updates the ***vb_startup*** file and issues following command:

vb_config -b host1:port1

vb_discport is used for breaking a link connection. Only one port of the link to be broken is given as parameter. Port information is defined with a hostname and a portnumber combination. host1 can be any host (local or remote). The other port of link is also broken (whether on local, remote or a third host).

The host name in the parameter list is pinged. If the host is unreachable, an error is returned, and the ***vb_startup*** file is not updated. At end of successful operation, the `vb_discport host1:port1` command is appended to end of the ***vb_startup*** file so that the ***vb_startup*** file will be up-to-date.

If at least one port of link is on a remote host, make sure the ***vb_bridge*** daemon has been started for the remote host, so that there is a bridge to that host for message transmission.

An error is returned if remote operation is required, but no host connection is available (no *vb_bridge* daemon for remote host).

10.5.3.5 vb_lports

The ***vb_lports*** function is used when user wants to list port information on local host.

A filter can be defined for the retrieved output. If the ***con*** option is selected, only the connected ports are displayed.

If the ***dis*** option is selected, the list of idle port numbers is printed. This information is used to see which ports can be used.

If the ***all*** option is used, information for all of the ports is printed.

10.5.3.6 vb_lhosts

The ***vb_lhosts*** function is used when user wants to list host connections information for the local host.

The output is a list of host names, that there is a bridge connection between that host and local host. Consequently, any host in the output can be used for remote operations. Before performing a remote operation, user can use ***vb_lhosts*** command, to see whether a message transfer path (bridge) is available to that host.

vb_lhosts script, performs the following command:

vb_config -h

10.5.3.7 vb_reset

The ***vb_reset*** script is used when we need to reset all port and host connections on all hosts in the virtual board environment.

The ***vb_reset*** script performs `vb_config -r` command and clears the ***vb_startup*** file. Resetting of remote hosts is handled by the ***vbrd*** driver.

10.6 Setting Up Virtual SS7 Connections Environment

Setting up a virtual SS7 connections environment requires Distributed7. This driver is been intended for, and designed to work with, Distributed7. As a result, Distributed7 must be installed on the all the machines involved in the virtual SS7 connections.

The Virtual Board Driver does not have an interface with a real SS7 board. If a virtual link is to be connected between two hosts, ***vbrd*** driver must be installed on both hosts.

10.6.1 Configuring *vbrd* Driver

When we consider that the virtual board replaces real SS7 boards, it will be more instructive if we first explain how a physical SS7 card (or board) behaves and then explain the ***vbrd*** driver behavior.

The operations on a real SS7 board are as follows:

1. Add SS7 board instance from MML (command: ADD-SS7BOARD:...)
2. Configure the board (command: MODIFY-SS7BOARD:...)
3. Add links on the previously added board.

A physical SS7 board is downloaded when it is configured. So is ***vbrd*** driver, but with an important difference, a physical board has board software, named Level-2 software, that is mandatory so that the SS7 card will function. However, a virtual board driver does not need this software, since it is already a software driver. From the point of view of Distributed7 there is no difference between a physical SS7 card and a virtual board, so level-2 code is downloaded to the ***vbrd*** driver as well. The ***vbrd*** driver does not use the downloaded code, but since Distributed7 expects an Acknowledgment message, the driver sends an acknowledgment message to Distributed7 in response.

Although the ***vbrd*** driver does not have Level-2 software, Distributed7 sends a download code to the driver when it is configured (CONF=ON). The following ***vbrd*** Level2 code files must exist under ***$EBSHOME/access/drv***:

- mtpl2.vbrd.rel
- sal.vbrd.rel

The contents of these files is not important. When a release is generated, any level2 code files of a physical board (e.g. *mtpl2.sbs334.rel* and *sal.sbs334.rel* files) are copied to *mtpl2.vbrd.rel* and *sal.vbrd.rel* files respectively.

The instance number of the ***vbrd*** driver is 0, the class of ***vbrd*** is "I", and the maximum number of ports on the ***vbrd*** driver is 32. These values must be used throughout ***vbrd*** operations.

To add a ***vbrd*** board, the MML command is as follows:

MML_TH>ADD-SS7BOARD:HOSTNAME=<hostname>,BOARDNM=vbrd,INST=0, PROTOCOL=<protocol>,CLASS=I,PORTS=32;

To configure a ***vbrd*** driver, the following MML command must be issued:

MML_TH>modify-SS7BOARD:HOSTNAME=<hostname>,BOARDNM=vbrd, INST=0,conf=ON

If the ***vbrd*** download files, i.e., *mtpl2.vbrd.rel* and *sal.vbrd.rel*, do not exist under ***$EBSOME/access/drv***, configuring the ***vbrd*** driver fails.

Now we have an SS7 board with name ***vbrd*** (just the same as **sbs332** etc.). Only one ***vbrd*** board with instance number 0 can be added on a host. When ***spmd*** opens the board device, it appends the instance to the boardname parameter. In this example, the resulting device becomes `/dev/vbrd0`. At MML commands, giving instance of the ***vbrd*** as 0 is a must. Since `/dev/vbrd0` is reserved for Distributed7, only Distributed7 can use `/dev/vbrd0`.

10.6.2 Link Creation/Activation - *vbrd* driver vs. Actual SS7 Card

After the ***vbrd*** board is configured, links can be added on ports of ***vbrd*** driver. The MML commands required to add a link is same with traditional methods. Listed below:

MML_TH>add-link:link=<link>,lset=<lset>,hostname=<hostname>,

BOARDNM=vbrd,INST=0,port=<portnumber >,slc=<slc>, priority=<priority>;

As stated above, **INST** parameter is 0, **BOARDNM** is "vbrd", **portnumber** can be a value in range (0-31), **slc** and **priority** can be any value in valid slc and priority range.

When a link is added, Distributed7 sends ***vbrd*** driver some messages to inform ***vbrd*** about some parameters of the added link, such as *sp*, *linkset*, *linkno*, and *protocol*. Those values are retrieved from the message and stored in a port table. Those values can be displayed with the `vb_config -l con` command on the related host. Those values are crucial for Distributed7 interaction, since Distributed7 expects to find correct values when it receives a message from the ***vbrd*** driver. Hence correct values must be observed after links are added.

After the links are added, they can be activated in the same way.

MML_TH>modify-linkstat:link=<link>,status=SET_ACT;

When the remote end of the link is also started, almost simultaneously the links are aligned. Link activation in a Virtual SS7 connections environment takes less time than in a physical connection.

After that point, Distributed7 may send or receive MSU’s over the aligned link.

10.7 On-Line Manual Pages

The Virtual SS7 Connections contains on-line reference manuals on all utility programs, and shell scripts. These reference manuals are provided in the form of *manual pages*. The user can invoke the UNIX standard *man(1)* tool to access them.

The source files for the Distributed7 manual pages are provided in the ***$EBSHOME/access/manpages*** directory. Manual pages for Virtual SS7 Connections functions are under ***man1v***. The user must expand the system’s ***$MANPATH*** environment variable setting to include the above directory in the search path, as follows:

```
setenv manpath ${manpath}:$EBSHOME/access/manpages
```


This page is intentionally blank.

Chapter 11: Glossary

The following table lists D7 abbreviations, and common SS7 and telecommunications industry acronyms.

Brief definitions are included for frequently used terms found in the D7 manuals.

Table G-1: Glossary of Terms

Acronym or Term	Description
ACG	Automatic Code Gapping
ACM	Address Complete Message
ADC	Automatic Call Distributor
AFR	Automatic Flexible Routing
AHT	Average Handle Time
AIN	Advanced Intelligent Network
AIOD	Automatic Identified Outward Calling
AM/MSC	Access MANAGER/Mobile Switching Center
AMA	Automatic Teleprocessing System
AMATPS	AMA Teleprocessing System
AMP	AIN Maintenance Parameter
ANI	Automatic Number Identification
ANM	Answer Message
ANSI	American National Standards Institute
API	Application Programming Interface
API	Application Programming Interface
ARP	Address Resolution Protocol
ASA	Average Speed of Answer
ASE	Application Service Element
ASN.1	Abstract Syntax Notation 1
ATB	All Trunks Busy
ATP	Acceptance Test Procedure
AUI	Attachment Unit Interface
AW	Admin Workstation
BAF	Bellcore AMA Format
BBG	Basic Business Group
BCC	Bellcore Client Company
BCD	Binary Coded Decimal

Table G-1: Glossary of Terms (Continued)

Acronym or Term	Description
BCI	Backward Call Indicators
BCLID	Bulk Calling Line Identification
BCM	Basic Call Model
BER	Basic Encoding Rules
BG	Business Group
BGID	Business Group Identification
BRI	Basic Rate Interface
BSN	Backward Sequence Number
CAC	Carrier Access Code
CAP	Competitive Access Provider
CC	Call Control
CCA	Call Control Adjunct
CCITT	Consultative Committee on International Telephone & Telegraph
CCS	Common Channel Signaling
CDAR	Customer Dialed Account Recording
CDP	Customized Dialing Plan
CDPD	Cellular Digit al Packet Data
CED	Call Entered Digits
CGB	Circuit Group Blocking Message
CGU	Circuit Group Unblocking Message
CIC	Circuit Identification Code
CIDS	Calling Identity Delivery & Suppression
CLID	Calling Line ID
CLLI	Common Language Location Identification
CMC	Cellular Mobile Carrier
CMS	(AT&T) Call Management System
CNAB	Call Name Delivery Blocking
CO	Central Office
COT	Continuity Test Message
CPC	Call Processing Control
CPE	Customer Premises Equipment
CPG	Call Process Message
CR	Conditional Requirement
CRA	Circuit Reservation Acknowledgment Message
CRM	Circuit Reservation Message
CS-1	Capability Set 1
CSC	Circuit Supervision Control
CSU	Channel Service Unit
CT	Call Type

Table G-1: Glossary of Terms (Continued)

Acronym or Term	Description
CVR	Circuit Validation Response Message
CVT	Circuit Validation Test Message
DACS	Digital Access Cross-Connect System
DCE	Data Circuit Equipment
DMP	Device Management Protocol
DN	Dialed Number
DNIS	Dialed Number Identification Service
DP	Dial Pulse
DPC	Destination Point Code
DSVD	Digital Simultaneous Voice and Data
DTE	Data Terminal Equipment
DTMF	Dial Tone Multifrequency
DUP	Data User Part
DXI	Data Exchange Interface
EA	Equal Access
EADAS	Engineering & Administration Data Acquisition System
EADASNM	EADAS Network Administration
EAEO	Equal Access End Office
EAMF	Equal Access Multifrequency
EBCDIC	Extended Binary Coded Decimal Interchange Code
EDP	Event Detection Point
EIA	Electronic Industries Association
EIR	Equipment Identification Register
EKTS	Electronic key Telephone Service
EMS	Event Management Service
EO	End Office
ESN	Electronic Serial Number
EXM	Exit Message
FCS	Frame Check Sequence
FISU	Fill-in Signal Unit
FRAD	Frame Relay Access Device
FRL	Facility Restriction Level
FUNI	Frame User Network Interface
FSD	Feature Specific Document
FSN	Forward Sequence Number
FSS	Facility Selective Service
FTE	Full Time Equivalent
FTP	File Transfer Protocol
FX	Foreign Exchange

Table G-1: Glossary of Terms (Continued)

Acronym or Term	Description
GN	Generic Name
GRS	Group Reset Message
GSC	Gateway Switching Center
GSM	Groups Special Mobile
GTT	Global Title Translations
GTV	Global Title Value
GUI	Graphical User Interface
HDLC	High Level Data Link Control
HFC	Hybrid Fiber Coaxial Cable
HLR	Home Location Register
HSL	High Speed Link
IAM	Initial Address Message
IC	Inter-exchange Carrier
ICP	Intelligent Call Processing
ICR	Intelligent Call Router
IDLC	Integrated Digital Loop Carrier
ISP	Intermediate Service Part
IDT	Integrated Digital Terminal
INR	Information Request Message
IP	Intelligent Peripheral **or** Internet Protocol
IPC	Interprocess Communication
IPI	Intelligent Peripheral Interface
ISP	Intermediate Service Part
ISPC	International Signaling Point Code
ISDN	Integrated Services Digital network (Used with CPE)
ISDNUP	ISDN User Part
ISUP	ISDN User Part (Used with Circuit Oriented)
IWX	Interworking Function
IXC	Inter-exchange Carrier
LAA	Longest Available Agent
LAN	Local Area Network
LATA	Local Access & Transport Area
LI	Length Indicator
LSL	Low Speed Link
LSSU	Link Status Signal Unit
LSSGR	LATA Switching & Signaling Generic Requirements
LOCREQ	Location Request
MAP	Mobility Application Part
MBG	Multi-switch Business Group

Table G-1: Glossary of Terms (Continued)

Acronym or Term	Description
MCC	Mobile Country Code
MIN	Mobile Identification Number
MGW	Mini-Gateway Prototype
MLHG	Multi-Line Hunt Group
MMI	Man-Machine Interface
MSC	Mobile Switching Center
MSISDN	Mobile Station ISDN Number
MSU	Message Signal Unit
MUX	Multiplexor
MTP	Message Transfer Part
NAA	Next Available Agent
NCA	Non-Call Associated
NCP	Network Control Point
NDC	National Destination Code
NIC	Network Interface Controller
NNI	Network Node Interface
NPA	Numbering Plan Area
NSP	Network Services Part
ODBC	Open Database Connectivity
OE	Office Equipment
OMAP	Operations & Maintenance Application Part
OPC	Origination Point Code
OPI	Open Peripheral Interface
OS	Operations System
OSI	Open Systems Interface
OTGR	Operations Technology Generic Requirement
PBX	Private Branch Exchange
PCS	Personal Communications Services
PG	Peripheral Gateway
PIC	Point In Call
PIM	Peripheral Interface Manager
PPP	Point-to-Point Protocol
PRI	Primary Rate Interface
PROFREQ	Profile Request
PSN	Alternative to PSTN (Public Switched Telephone Network)
PSTN	Public Switched Telephone Network
REGNOT	Registration Notification
RISC	Reduced Instruction Set Computing
ROUTREQ	Routing Request

Table G-1: Glossary of Terms (Continued)

Acronym or Term	Description
SANC	Signaling Area Network Code
SCCP	Signaling Connection Control Part
SCP	Service Control Point
SDLC	Synchronous Data Link Control
SEP	Signaling Endpoint
SF	Status Field
SI	Service Indicator
SIF	Signaling Information Field
SIO	Signaling Information Octet/Service Information Octet
SLC	Signaling Link Code
SLIP	Serial Line Internet Protocol
SLS	Signaling Link Selection
SLP	Service Logic Program
SMDS	Switched Multi-megabit Digital Service
SMH	Signaling Message Handling – MTP functions comprised of routing (HMRC), discrimination (HMDC), and distribution (HMDT)
SMP	Symmetric Multiprocessor
SMS	Service Management System
SN	Services Node
SNA	Systems Network Architecture
SNM	Signaling Network Management
SNMP	Simple Network Management Protocol
SNT	Signaling Network Testing
SP	Signaling Point
SPC	Signaling Point Code
SPID	Service Provider Identifier
SPM	Signaling Point Manager
SQL	Structured Query Language
SPR	Signaling Point w/SCCP Relay
SPRC	Signaling Procedure Control
SRTC	Subrate Channel
SS7	Signaling System 7
SSF	Sub-Service Field
SSN	Sub-System Number
SSP	Service Switching Point
STP	Signaling Transfer Point
SU	Signal Units
TA	Technical Advisory
TC	Transaction Capabilities

Table G-1: Glossary of Terms (Continued)

Acronym or Term	Description
TCAP	Transaction Capabilities Application Part
TCM	Traveling Class Mark
TCP/IP	Transmission Control Protocol/Internet Protocol
TDM	Time Division Multiplexor
TDP	Trigger Detection Point
TLDN	Temporary Local Directory Number
TR	Technical Reference
TUP	Telephone Users Part
UDP	User Datagram Protocol
UDT	Unitdata
UDTS	Unitdata Service
VAD	Voice Activated Dialing
VANC	Voice Activated Network Control
VLR	Visitor Location Register
VPN	Virtual Private Network
VRU	Voice Response Unit
WAN	Wide Area Network
WATS	Wide Area Telephone Service
XUDT	Extended Unitdata

This page is intentionally blank.

Index

A

B

C

Index

Index

Index

Index

Index

N

Index

Index

Index

Index

