Signaling Gateway Client User Manual

Part No. 160 - 3001 -01

Release 1.6.0

April 08, 2009

35 Nutmeg Drive
Trumbull, CT 06611

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

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 relies on the specifications supplied by the OEM vendors.

ALL RIGHTS RESERVED

Table of Contents

Chapter 1: Introduction 1

1.1 Document Overview 1
1.2 Scope 1
1.3 How to Use this Manual.. 2
1.4 Notations and Conventions 2
1.4.1 Revisions and Updates 2
1.4.2 Alert Messages 2
1.5 Related Documents and References.. 3

Chapter 2: Overview 5

2.1 Chapter Overview 5
2.1.1 SS7 Overview 5
2.1.1.1 SS7 Protocol Stack.. 7
2.1.2 SIGTRAN Overview.. 10
2.1.2.2 Signaling Gateway Process (SGP) 10
2.1.2.3 Application Server (AS).. 10
2.1.2.4 Application Server Process (ASP) 10
2.1.2.5 IP Server Process (IPSP).. 10
2.1.2.6 IP Application Server (IPAS) 11
2.1.2.7 SCTP Association 11
2.1.2.8 Signaling Point Management Cluster (SPMC) 11
2.1.3 NewNet Communication Technologies' Signaling Gateway Suite 11
2.2 Signaling Gateway Client Features and Capabilities.. 13
2.2.1 SS7-IP Interworking 13
2.2.2 Message Distribution 13
2.2.2.2 Traffic Distribution 14
2.2.3 Reliability.. 16
2.2.3.2 Fault Tolerance 16
2.2.4 Fault Management.. 17
2.2.5 Troubleshooting Tools 17
2.2.6 Security 17
2.3 Signaling Gateway Client Interfaces 18
2.3.7 User Application Interface 19
2.3.8 SCTP Interface 19
2.3.9 OAM Interface 19
2.3.10 Cluster LAN Interface.. 19
2.4 Software Components 20
2.5 Capacity 21

Chapter 3: Distributed7 Overview... 23

Table of Contents

3.1 Chapter Overview 23
3.2 General Description 23
3.3 Features 24
3.3.1 UNIX Open-Architecture 24
3.3.2 Common Channel Signaling System No. 7 25
3.3.2.2 Signaling Connection Control Part 25
3.3.2.3 Transaction Capabilities Application Part 25
3.3.2.4 ISDN User Part 25
3.3.2.5 Operations, Maintenance, and Application Part 25
3.3.2.6 Distributed SS7 Stack Operations 26
3.3.3 Platform Services 26
3.3.3.2 Distributed Process Management 28
3.3.3.3 Distributed Shared Memory Management 29
3.3.3.4 Distributed Kernel Memory Management 29
3.3.3.5 Distributed Node/Configuration Management 30
3.3.3.6 Distributed Alarm/Event Management 31
3.3.3.7 Redundant LAN Support 32
3.3.3.8 Network Clock Synchronization 32
3.3.3.9 Application Programming Guides 32
3.3.3.10 Backward Compatibility 32
3.3.4 Intelligent Network Emulation (INE) 33
3.4 Architecture 34
3.4.1 SPM API Library 36
3.4.2 APM API Library 36
3.4.3 DSM API Library 36
3.4.4 OA&M API Library 36
3.4.5 Alarm API Library 36
3.4.6 MTP API Library 37
3.4.7 SCCP API Library 37
3.4.8 TCAP API Library 37
3.4.9 Raw TCAP API Library 37
3.4.10 ISUP API Library 37
3.4.11 ISUP Advice of Charge (AoC) API Library 37
3.4.12 Gateway API Library 37
3.4.13 IS41-D API Library 37
3.4.14 GSM MAP API Library 38
3.4.15 GSM A-Interface API Library 39
3.5 Distributed7 System Applications 39
3.5.1 Media Server Network Signaling Interface 39
3.5.2 Customer Routing Point 41

Table of Contents

3.5.3 Short Message Server 42
3.5.4 Home Location Register 44
3.5.5 Visitor Location Register (VLR) 45
3.6 Major Standards Compliance 46
3.7 Capacity and Configuration Options 47
3.7.1 SS7 Database Capacity 47
3.7.2 Host Platform Options 47
3.7.3 SS7 Controller Options 48
3.7.4 External Dependencies 49
Chapter 4: Concepts 51
4.1 Chapter Overview 51
4.2 Managed Objects and the Object Server 51
4.2.1 Object Server 51
4.2.2 Managed Objects 56
4.3 Distributed System Characteristics 68
4.3.1 Resource Sharing 69
4.3.2 Reliability 70
4.3.2.2 Fault Tolerance 71
4.3.3 Scalability 73
4.3.4 Transparency 74
4.3.5 Performance 75
4.4 Core Product Specifications 75
4.4.1 Resource Sharing 75
4.4.2 Reliability 76
4.4.2.2 Reliability through Fault Tolerance 77
4.4.2.3 Failure Semantics 77
4.4.2.4 LAN Failures 77
4.4.3 Scalability 79
4.4.4 Transparency 80
4.4.4.2 Location Transparency 80
4.4.4.3 Concurrence Transparency 80
4.4.4.4 Replication Transparency 81
4.4.4.5 Failure Transparency 81
4.4.4.6 Growth/Retrofit Transparency 81
4.5 Product Specifications 82
4.5.1 MTP Layer Product Specifications 82
4.5.1.2 Reliability 82
4.5.1.3 Scalability 84
4.5.1.4 Transparency 84

Table of Contents

4.5.1.5 Performance Considerations 86
4.5.2 SCCP Layer Product Specifications.......... 87
4.5.2.2 Reliability 87
4.5.2.3 Scalability.......... 88
4.5.2.4 Transparency 88
4.5.3 TCAP Product Specifications.......... 90
4.5.3.2 Reliability 90
4.5.3.3 Scalability.......... 91
4.5.3.4 Transparency 91
4.5.4 ISUP Product Specifications 93
4.5.4.2 Reliability 93
4.5.4.3 Scalability.......... 94
4.5.4.4 Transparency 94

Chapter 5: Installation 97

5.1 Overview 97
5.2 Software Installation 98
5.2.1 Disk Space Requirements.......... 99
5.2.2 Installation Steps 100
5.2.2.2 Installation 104
5.2.3 Live Upgrade.......... 108
5.2.3.2 Live Upgrade Steps 109
5.2.4 Software Removal 111

Chapter 6: Operations, Administration, and Maintenance 113

6.1 Overview 113
6.2 Operating Environment 113
6.2.1 User Account.......... 113
6.2.2 Directory Structure and Files 114
6.2.3 Environment Variables.......... 115
6.2.4 Application Processes 116
6.3 General Operations.......... 117
6.3.1 Starting the Software 117
6.3.2 Stopping the Software 118
6.3.3 Automatic Startup 119
6.3.4 Using MML for Configuration and Query 120
6.4 Getting Started.......... 121
6.5 Software Configuration 123
6.5.1 Network Configuration 123
6.5.1.2 IP Security Configuration.......... 126
6.5.1.3 SCTP Configuration 126

Table of Contents

6.5.1.4 Cluster LAN Configuration ... 133
6.5.2 Signaling Network Configuration ... 136
6.5.3 Signaling Gateway Client Configuration ... 138
6.5.3.2 Local ASP ... 139
6.5.3.3 Remote SG ... 139
6.5.3.4 Remote SGP ... 140
6.5.3.5 SS7 Destination Point Code ... 140
6.5.3.6 Configuring a Route Key ... 140
6.5.3.7 Configuring an AS ... 141
6.5.3.8 Defining AS Traffic Control ... 142
6.5.3.9 IP Server Process ... 142
6.5.3.10 IP Application Server ... 143
6.5.4 Configuring SCCP ... 143
6.5.4.2 Subsystem Provisioning ... 144
6.5.4.3 Concerned SP Provisioning ... 144
6.5.4.4 Global Title (GT) Database Provisioning ... 144
6.5.4.5 Global Title Entry Table (GTENTRY) Provisioning ... 145
6.5.4.6 Mated Subsystem Provisioning ... 145
6.5.5 Configuring ISUP ... 145
6.5.5.2 Adding ISUP Circuit Groups ... 146
6.5.5.3 Adding ISUP Circuits ... 147
6.5.6 Alarm Configuration ... 148
6.5.6.2 Alarm Groups ... 148
6.5.6.3 Alarm Display ... 149
6.5.7 SNMP Configuration ... 149
6.5.7.2 Starting the SNMP Agent ... 164
6.6 Using AccessMOB ... 164
6.6.1 Introduction ... 164
6.6.1.2 Environment Variable Settings ... 165
6.6.1.3 Conventions ... 165
6.6.1.4 Starting the Managed Object Browser ... 166
6.6.2 Managed Object Browser ... 167
6.6.2.2 Accessing Menus ... 167
6.6.2.3 Using the Mouse ... 168
6.6.2.4 Entering Data in the Dialog Box ... 168
6.6.2.5 Managed Objects Parameters ... 170
6.6.3 Managed Object Browser Windows and Operation ... 173
6.6.3.2 Selecting an Operation Mode ... 174
6.6.3.3 Selecting Managed Objects ... 175

Table of Contents

6.6.3.4 Operation Dialog Boxes 176
6.6.4 Error Messages 183

Chapter 7: MML Commands.................. 187

7.1 Chapter Overview 187
7.1.1 MML Commands 187
7.1.2 Managed Objects.................. 187
7.1.3 MML Network Element Labels 188
7.1.4 Syntax Rules for the Command Line 189
7.1.5 String-Constant Data Entry Method.................. 189
7.1.6 Case Sensitivity 189
7.1.7 Command Syntax 190
7.1.8 Output Messages 191
7.2 SCCP MML Commands 204
7.2.1 Concerned Point Code (CPC) 204
7.2.2 Global Title (GT) 206
7.2.3 Global Title Entry (GTENTRY) 208
7.2.4 Mate (MATE).................. 210
7.2.5 SCCP (SCCP).................. 212
7.2.6 SCCP Signaling Point (SNSP) 214
7.2.7 Subsystem (SUBSYS).................. 216
7.2.8 Local Subsystem (LOCALSUBSYS) 218
7.2.9 Connection (CONNECTION).................. 219
7.3 ISUP MML Commands 220
7.3.1 ISUP Circuits (ISUPCCT) 220
7.3.2 ISUP Circuit Group (ISUPCGRP) 224
7.3.3 ISUP Signaling Node (ISUPNODE).................. 227
7.3.4 ISUP Configuration (ISUP) 230
7.3.5 ISUP Timer (ISUPTMR) 233
7.4 System MML Commands 237
7.4.1 Host (HOST) 237
7.4.2 Network (NTWK) 239
7.5 Signaling Gateway Client MML Commands.................. 241
7.5.1 Application Server (SGCAS) 241
7.5.2 Application Server Process (SGCASP).................. 243
7.5.3 AS-ASP Traffic Control (SGCASTFC) 245
7.5.4 Destination Point Code (SGCDPC) 247
7.5.5 Routing Key (SGCRK) 249
7.5.6 Routing Key Range (SGCRKRNG).................. 251
7.5.7 Signaling Gateway (SGCSG).................. 253

Table of Contents

7.5.8 Signaling Gateway Process (SGCSGP) 255
7.5.9 Signaling Point to Network Appearance Mapping (SGCSPNA).......... 258
7.5.10 IP Server Process 261
7.5.11 IP Application Server (SGCIPAS).......... 263

Chapter 8: System Processes 267

8.1 Chapter Overview 267
8.2 Process Listing 268
8.2.1 AccessAlarm (alarmd) 269
8.2.2 AccessMOB (mob) 271
8.2.3 AccessOMAP (omapd) 272
8.2.4 AccessSNMP 274
8.2.5 AccessStatus.......... 277
8.2.6 AccessMonitor 279
8.2.7 apmd.......... 280
8.2.8 dkmd.......... 283
8.2.9 dsmd 285
8.2.10 isupd 287
8.2.11 logd.......... 289
8.2.12 mlogd 290
8.2.13 mml 292
8.2.14 mmi 293
8.2.15 netd.......... 294
8.2.16 rtc_agent.......... 296
8.2.17 scmd 298
8.2.18 spmd 300
8.2.19 tcmd.......... 302
8.2.20 upmd.......... 303
8.3 Configuration Files 305
8.3.1 apmconfig.......... 305
8.3.2 apmconfig.old 318
8.4 Signaling Gateway Client Processes.......... 323
8.4.1 aspd 323

Chapter 9: User Commands 325

9.1 Chapter Overview 325
9.2 Platform Utilities.......... 329
9.2.1 ebs_alarm 329
9.2.2 ebs_apidemo.......... 331
9.2.3 ebs_audit 332

Table of Contents

9.2.4 ebs_brdfinfo 333
9.2.5 ebs_cfgbrd 335
9.2.6 ebs_config 337
9.2.7 ebs_dbconfig 339
9.2.8 ebs_dnlbrd 341
9.2.9 ebs_explain........ 343
9.2.10 ebs_hbeat 344
9.2.11 ebs_ipcbm........ 346
9.2.12 ebs_log 347
9.2.13 ebs_loopback 349
9.2.14 ebs_mngbrd 351
9.2.15 ebs_oldapidemo........ 353
9.2.16 ebs_modinstall........ 354
9.2.17 ebs_modremove 355
9.2.18 ebs_modunload........ 356
9.2.19 ebs_mtpglobal 357
9.2.20 ebs_pkgrm 358
9.2.21 ebs_ps 359
9.2.22 ebs_qinfo 363
9.2.23 ebs_qlist........ 365
9.2.24 ebs_qstat 367
9.2.25 ebs_report 370
9.2.26 ebs_setrelease 372
9.2.27 ebs_showlink 373
9.2.28 ebs_shutdown 376
9.2.29 ebs_start........ 377
9.2.30 ebs_stop 378
9.2.31 ebs_sync 379
9.2.32 ebs_sysinfo 381
9.2.33 ebs_tasklist 382
9.2.34 ebs_tune........ 383
9.2.35 getcfg 384
9.3 APM Utilities 388
9.3.1 apm_audit 388
9.3.2 apm_getstate........ 389
9.3.3 apm_kill 390
9.3.4 apm_killall........ 392
9.3.5 apm_ps 394
9.3.6 apm_report 397
9.3.7 apm_setstate 399

Table of Contents

9.3.8 apm_start 401
9.3.9 apm_stop 403
9.3.10 apm_trcapture 405
9.3.11 apm_trclear 408
9.3.12 apm_trgetmask 409
9.3.13 apm_trinit 411
9.3.14 apm_trsetmask 413
9.3.15 apm_trshow 415
9.3.16 apm_update 417
9.4 DSM Utilities 418
9.4.1 dsm_apidemo 418
9.4.2 dsm_audit 419
9.4.3 dsm_bm 421
9.4.4 dsm_list 422
9.4.5 dsm_rm 425
9.4.6 dsm_stat 426
9.5 DKM Utilities 428
9.5.1 dkm_apidemo 428
9.5.2 dkm_bm 429
9.5.3 dkm_dump 430
9.5.4 dkm_list 431
9.5.5 dkm_rm 436
9.5.6 dkm_sar 437
9.5.7 dkm_stat 439
9.5.8 dratest 441
9.6 TCAP Utilities 448
9.6.1 rtc_dump 448
9.6.2 rtc_stat 450
9.6.3 tcm_apidemo 451
9.6.4 tcm_list 452
9.6.5 tcm_stat 454
9.6.6 tcm_tune 455
9.7 Virtual Board Utilities 457
9.7.1 vb_addhost 457
9.7.2 vb_bridge 458
9.7.3 vb_config 460
9.7.4 vb_connhosts 462
9.7.5 vb_connports 463
9.7.6 vb_discport 464

Table of Contents

9.7.7 vb_lhosts........ 465
9.7.8 vb_lports........ 466
9.7.9 vb_reset........ 467
9.7.10 vb_startup........ 468
9.7.11 snmptest........ 469
9.7.12 snmptrapd........ 470
9.7.13 snmpwalk........ 471
9.7.14 snmpget........ 472
9.7.15 AccessStatus........ 473
9.7.16 db2date........ 475
9.7.17 db2text........ 476
9.8 Signaling Gateway Client Application Utilities........ 478
9.8.1 sgc_pkgrm........ 478
9.8.2 sgc_start........ 479
9.8.3 sgc_stop........ 481
9.8.4 sgc_setrelease........ 482
9.8.5 sgc_trace........ 482

Chapter 10:Maintenance and Troubleshooting........ 485

10.1 Overview........ 485
10.2 Software Maintenance........ 485
10.2.1 Configuration Backup........ 485
10.2.2 Monitoring Alarms........ 486
10.2.3 Monitoring Logs........ 487
10.3 Troubleshooting Tools........ 487
10.3.1 Alarms........ 487
10.3.2 Log Files........ 492
10.3.3 Runtime Tracing........ 493

Appendix A:Glossary........ 495

List of Figures

Figure 2-2: SS7 Protocol Stack 8
Figure 2-3: SS7 Stack Working with Applications 9
Figure 2-4: NewNet Communication Technologies' Signaling Gateway Suite 12
Figure 2-5: SS7-IP Interworking 13
Figure 2-6: Signaling Gateway Client Interfaces 18
Figure 2-7: Signaling Gateway Client Software Architecture 20
Figure 3-2: Distributed Memory Management.. 30
Figure 3-3: Distributed7 Software Architecture.. 34
Figure 3-4: Media Server Network Interface 40
Figure 3-5: Distributed7 as a Customer Routing Point 41
Figure 3-6: SMServer Software Architecture... 42
Figure 3-7: HLR in the Wireless Intelligent Network.. 44
Figure 3-8: VLR in the Wireless Intelligent Network.. 45
Figure 4-2: MTP Managed Object Containment Structure 53
Figure 4-3: SCCP Managed Object Containment Structure... 54
Figure 4-4: ISUP Managed Object Containment Structure.. 54
Figure 4-5: SPM Managed Object Containment Structure 55
Figure 4-6: Network Managed Object Containment Structure 55
Figure 4-7: Alarm Managed Object Containment Structure 55
Figure 6-2: Sample Network Configuration.. 137
Figure 6-3: SNMP Architecture 150
Figure 6-4: Internet MIB Tree 151
Figure 6-5: Distributed7 MIB - MTP Tree 1.. 152
Figure 6-6: Distributed7 MIB - MTP Tree 2.. 153
Figure 6-7: Distributed7 MIB - MTP Tree 3.. 154
Figure 6-8: Distributed7 MIB - Hardware Tree 155
Figure 6-9: Distributed7 MIB - Network Tree 156
Figure 6-10: Distributed7 MIB - Alarm Tree.. 157
Figure 6-11: Distributed7 MIB (SCCP Layer - 1)... 158
Figure 6-12: Distributed7 MIB (SCCP Layer - 2)... 159
Figure 6-13: Distributed7 MIB (ISUP Layer - 1).. 160
Figure 6-14: Distributed7 MIB (ISUP Layer - 2).. 160
Figure 6-15: MOB Main Window 173
Figure 6-16: Key Selection Dialog Box 176
Figure 6-17: Add Dialog Box.. 177
Figure 6-18: Modify Dialog Box... 178
Figure 6-19: Delete Dialog Box 179
Figure 6-20: View Dialog Box 181

List of Figures

List of Tables

Table 2-1: Capacity 21
Table 3-2: Additional Standards Compliance 46
Table 3-3: Interfaces 46
Table 3-4: Standard SS7 Database Capacity.......... 47
Table 3-5: Available SS7 Controller Options 48
Table 3-6: Executable External Dependencies 49
Table 3-7: Library External Dependencies 50
Table 4-2: NETWORK (NTWK) Branch Managed Object Descriptions 60
Table 4-3: ALARM Branch Managed Object Descriptions 61
Table 4-4: MTP Managed Object Descriptions 62
Table 4-5: SCCP Branch Managed Object Descriptions 66
Table 4-6: ISUP Branch Managed Object Descriptions 67
Table 6-2: Environment Variables 115
Table 6-3: Application Processes.......... 116
Table 6-4: SNMP Configuration Directories and Files.......... 161
Table 7-2: SCCP Configuration Managed Objects 192
Table 7-3: ISUP Configuraton Managed Objects 196
Table 7-4: System Configuration Managed Objects 199
Table 7-5: Signaling Gateway Client Configuration Managed Objects 200
Table 7-6: SCCP Display Values 213
Table 7-7: SNSP Display Values 215
Table 7-8: SUBSYS Display Values.......... 217
Table 7-9: LOCALSUBSYS Display Values 218
Table 7-10: CONNECTION Display Values.......... 220
Table 7-11: ISUP Circuit Display Report 222
Table 7-12: ISUP Circuit Display Values 223
Table 7-13: ISUP Circuit Group Display Report– Pre-Call Ctrl & Maintenance Activation..... 225
Table 7-14: ISUP Circuit Group Display Report – Post-Call Ctrl & Maintenance Activation .. 226
Table 7-15: ISUP Timers 234
Table 7-16: ISUP Related Timers to Modify 236
Table 7-17: SGCASTFC Display Values 246
Table 7-18: SGCSPN Display Values 260
ebs_ps Output Description.......... 360
Table 9-2: ebs_qinfo Output Description.......... 364
Table 9-3: ebs_qlist Output Description 365
Table 9-4: ebs_qstat Output Description.......... 368
Table 9-5: ebs_showlink Output Description 374
Table 9-6: apm_ps Output Description 395
Table 9-7: dsm_list Output Column Description 423
Table 9-8:

List of Tables

Table 9-9: dsm_stat Output Column Description.. 426
Table 10-2: OAM Alarm Group.. 488
Table 10-3: M3UA Alarm Group.. 489
Table 10-4: M3UA Error Group 490
Table 10-5: Program IDs for Tracing... 493

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

Date	Pages Replaced	Description of Changes	Assoc SW Release
7/13/08	Chapter 6	Section 6.5.3.10: updated five IPASs to 8 IPASs.	1.5.0
	Chapter 7	Section 7.5.4: Updated NAID parameter value and added note. Section 7.5.9: Updated NAID parameter value and added note.	
5/23/08	All	Updated the company name, logo, address and contact information. Updated the copyright and trademark information.	1.5.0
7/6/07	Chapter 6	Updated section 6.5.1.3.	1.5.0 GA
6/15/07	Chapter 1	Updated section 1.5, Documents and References.	1.5.0 beta
	Chapter 7	Updated sections 7.5.3, 7.5.10 and 7.5.11	
2/07/07	All	Replaced Solaris 2.8 with Solaris 10. Removed most references to sctpd.	1.5.0 beta
	Chapter 1	Added a new reference in the Related Documents and References section.	
	Chapter 2	Edits were made to the following sections: 2.1.2.5, 2.1.3, 2.2.1, 2.2.2.2, 2.2.3.1, and 2.5	
	Chapter 3	Edited section 3.7.2.	
	Chapter 4	Edited section 4.2.1 diagram.	
	Chapter 5	Edited section 5.2.1.	
	Chapter 6	Edited section 6.5.1.3. Added sections 6.5.3.9 and 6.5.3.10.	
	Chapter 7	Edited section 7.5.3. Added sections 7.5.10 and 7.5.11.	
	Chapter 8	Deleted section 8.4.1 sctpd.	
	Chapter 9	Edited sections 9.8.3 and 9.8.5.	
	Chapter 10	Edited Table 10-5.	
	Glossary	Added IPSP and IPAS definitions.	
7/31/06	Chapter 1	Section 1.2 Scope: removed reference to gnu facility.	1.4.0
	Chapter 3	Table 3-5 Host Platform Options: removed references to gnu.	
	Chapter 4	Section 4.1 Chapter Overview: removed reference to gnu.	
	Chapter 5	Section 5.2.2.2 Live Upgrade Steps: updated version numbers. Section 5.2.3.2 Live Upgrade Steps: updated version numbers.	
	Chapter 6	Table 6-1 Directories and Files: removed rows for MML command log file and MML command history file.	
	Chapter 7	Section 7.1.4: Syntax Rules for the Command Line: removed reference to History command.	
	Chapter 8	Section 8.2.13 mml: removed out-of-date listings from Files. Section 8.2.14 mmi: removed out-of-date listings from Files.	
7/22/05	All	Updated versions and dates for D7 1.4.0.	1.1.1
	Chapter 2	Updated Table 2-1: Capacity.	
	Chapter 3	Updated Table 3-4: Host Platform Options. Updated Table 3-5: Available SS7 Controller Options.	

Date	Pages Replaced	Description of Changes	Assoc SW Release
2/21/03	Chapter 3	Table 3-5 Available SS7 Controller Options: Updated footnote 9 to include ARTIC1000/2000 boards and updated table	1.1.1
	Chapter 4	Table 4-1 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 4-4 MTP Managed Object Descriptions: added L2ECM, PCRN1 and PCRN2 to ***mtp*** object values; added artic8260 board set value to link boardnum parameter	
		Table 4-5 SCCP Branch Managed Object Descriptions: Updated ***gtentry*** and ***gt*** managed objects	
		Section 4.5.3.3 Scalability: Updated text	
		Section 4.5.4.1 Flexibility: Added bullet item	
	Chapter 5	Updated D7 release to 1.3.0; updated Disk Space Requirements, Table 5-1; deleted note after step 11 in Installation, Section 5.2.2.2	
	Chapter 6	Figure 6-7 Distributed7 MIB MTP Tree 3: Updated to include new link parameters. L2ECM, PCRN1 and PCRN2	
		Figure 6-8 Distributed7 MIB Hardware Tree: Added pmlink and ctbusTable branches from ss7board	
	Chapter 7	Table 7-2 SCCP Configuration Managed Objects: Modified GTENTRY object	
		Section 7.2.2 Global Title (GT): Edited names, commands, parameters, errors and examples	
		Section 7.2.3 Global Title Entry (GTENTRY): Edited commands, parameters, errors and examples	
		Section 7.5.1 Application Server (SGCAS): Modified ADD command and examples	
		Section 7.5.2 Application Server Process (SGCASP): Modified ***sgredmode*** paramter and examples	
		Section 7.5.3 AS-ASP Traffic Control (SGCASTFC): Modified ADD command, deleted ***operst*** parameter and modified examples	
		Section 7.5.7 Signaling Gateway (SGCSG): Modified ADD command, ***mode*** parameter and examples	
		Section 7.5.8 Signaling Gateway Process (SGCSGP): Modified ***sctpport*** parameter and examples	
	Chapter 8	Section 8.2.4 AccessSNMP: Updated daemon	
		Section 8.2.13 MML: Updated errors	
		Section 8.2.14 MMI: Updated errors	
		Section 8.2.15 netd: Updated ***netd*** daemon, which now takes '-i' command-line option for Solaris IP network multipathing (IPMP) support	
2/21/03	Chapter 8	Section 8.2.18 spmd: Added artic8260 to the list of boards in the third paragragh of DESCRIPTION	1.1.1
	Chapter 9	Artic8260 board references added throughout chapter (Sections 9.2.4, 9.2.5, 9.2.8, 9.2.14, 9.2.35 and Table 9-6)	
		Table 9-1 User Command Summary: Added ***ebs_dbconfig*** to table	
		Section 9.2.7 ebs_dbconfig: Added new section	
		Section 9.2.32 ebs_sysinfo: Updated ***ebs_sysinfo*** utility, which now provides '-l' command-line option	

Date	Pages Replaced	Description of Changes	Assoc SW Release
11/25/02	Chapter 1	Changed product name to Signaling Gateway Client; added bullets to Section 1.5, Related Documents and References	1.1.0
	Chapter 2	Changed product name to Signaling Gateway Client	
	Chapter 3	Changed product name to Signaling Gateway Client	
	Chapter 4	Changed product name to Signaling Gateway Client	
	Chapter 5	Changed product name to Signaling Gateway Client; updated patch in Caution statement under Section 5.2.2, Installation Steps; updated D7 release to 1.2.1, directories and steps in Section 5.2.2.2, Installation	
	Chapter 6	Changed product name to Signaling Gateway Client; modified the procedures in Section 6.3.1, Starting the Software and Section 6.3.2, Stopping the Software; updated the SCTP Daemon Configuration File in Section 6.5.1.3, SCTP Configuration; changed ANSI_92 to ANSI_96 in Section 6.5.2.1, MTP Configuration; updated Section 6.5.3, Signaling Gateway Client Configuration; updated Figure 6-4 Internet MIB Tree	
	Chapter 7	Changed product name to Signaling Gateway Client; added HOSTNAME parameter to Table 7-4 Signaling Gateway Client Configuration Managed Objects SGCSGP and SGCASP; updated Signaling Gateway Client Managed Objects in Section 7.5; changed name of	
	Chapter 8	Section 7.5.9 to Signaling Point to Network Appearance Mapping	
	Chapter 9	Changed product name to Signaling Gateway Client; updated directory paths in Section 8.4.1, sctpd and Section 8.4.2, sgpd	
	Chapter 10	Changed product name to Signaling Gateway Client; updated utility information in Section 9.8.2, sgc_start amd Section 9.8.3 sgc_stop; changed sg to asp in Section 9.8.5, sg_trace Changed product name to Signaling Gateway Client; changed directory path in Section	
11/24/02		10.2.1, Configuration Backup; removed alarms 020108 and 020109 from Table 10-1 ASP Alarm Group; corrected Caution statement in Section 10.3.3, Runtime Tracing Removed all the SS8 Connect7 information in the Overview, MML and Configuation	1.1.Ø Beta
11/24/02	Chapter 6:	chapters and replaced it with the SS8 Distributed7 information. Chapter 3 is a new chapter with an overview of Distributed7.	1.1.Ø Beta
11/24/02	Chapter 7:	Update the Configuration Chapter with the new steps and commands. Also added ISUP and SCCP configuration, and AccessMOB from Distributed7. Updated the Signaling Gateway Client MML commands as follows: NA is now SGCSPNA,	1.1.Ø Beta
11/24/02	Chapter 8:	AS is now SGCAS, ASP is now SGCASP, DPC is now SGCDPC, SG is now SGCSGP, SGC, SPC and CONN were deleted, and the new SGCRK, SGCRKRNG, and SGCASTFC	1.1.Ø Beta
11/24/02	Chapter 9	were added. Added a new chapter, System Processes, that has the Distributed7 and Signaling Gateway	1.1.Ø Beta
4/3Ø/Ø2		Client system processes.	1.Ø.Ø
4/3Ø/Ø2	Chapter 3	Replaced the SS8 Connect7 alarms with the Distributed7 alarms and updated the Signaling	1.Ø.Ø

Gateway Client alarms.

Updated all beta information to GA text.

Updated the Installation steps to reflect the GA changes.

Date	Pages Replaced	Description of Changes	Assoc SW Release
4/3Ø/Ø2	Chapter 5	Correct the first bullet in Section 5.1.3 to state that only the ASID is a number less than 11Ø. Added a note to the Network Appearance managed object stating that the PROTOCOL parameter must be set to ITU93 and the PCSIZE parameter must be set to 24 for a network appearance of China. Corrected the range for the NAID parameter in the RK MO from -65535 to 65535 to Ø to 65535. Corrected the range for the SCTPPORT parameter in the SG MO from Ø to 99999 to Ø to 32768.	1.Ø.Ø
2/15/Ø2		Initial release	1.Ø.Ø Beta

This page is intentionally blank.

Chapter 1: Introduction

1.1 Document Overview

This document describes the architecture and interface of Signaling Gateway Client, also known throughout this manual as SGC.

1.2 Scope

The manual is organized into the following chapters describing the Signaling Gateway Client:

You are Here!

- **Chapter 1: Introduction** - introduces this manual, including specific notations and conventions used in the manual, and related documents and references.
- Chapter 2: Overview - provides an overview of the Signaling Gateway Client that describes the architecture, key concepts, and features of the Signaling Gateway Client.
- Chapter 3: Distributed7 Overview - provides an overview of the Distributed7 software platform.
- Chapter 4: Concepts - provides a general overview of SS7, the Signaling Gateway Client managed object concepts, and the Product Specifications.
- Chapter 5: Installation - has the steps to install the Distributed7 and Signaling Gateway Client software.
- **Chapter 6: Operations, Administration, and Maintenance** - provides an in depth discussion of how the Signaling Gateway Client system operates. Steps are included for the initial usage prior to application development. Also included is a sample configuration section that describes and provides examples of the usage of MML sessions and commands for SS7 and M3UA configuration.
- **Chapter 7: MML Commands** - provides reference information for the Signaling Gateway Client service provisioning, statistics, general system, and Managed Objects including syntax and usage and a set of common UNIX commands
- Chapter 9: User Commands - documents the user commands (scripts) that come with the Signaling Gateway Client software.
- **Chapter 10: Maintenance and Troubleshooting** - identifies the Signaling Gateway Client alarms and troubleshooting tools.
- **Appendix A:** - lists Signaling Gateway Client abbreviations, common SS7 and telecommunication industry acronyms.

1.3 How to Use this Manual

The *Signaling Gateway Client User Manual* is intended for use by the operator during operation, configuration, and maintenance of the software and to aid with application development. This manual and all documents referenced within are required for proper operation.

1.4 Notations and Conventions

This paragraph describes the notations and conventions that are used in this manual. Conventions have been established for commands and file names, window names and heading names, acronyms, mnemonics, and alert messages. Each of these conventions is described in a subsection that follows.

1.4.1 Revisions and Updates

NewNet Communication Technologies seeks to provide total quality and customer satisfaction through continuous improvement. Toward that goal, revisions to the manual will occur from time to time due to software enhancements or documentation enhancements. Documentation changes will be included in the release notes for software point releases.

1.4.2 Alert Messages

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

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

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

1.5 Related Documents and References

This paragraph lists the related documents that are beyond the scope of the Signaling Gateway Client manual set. The documents listed below are referenced from this manual, and contain additional information that may be helpful to the user:

- Framework Architecture for Signaling Transport, RFC 2719, Oct 1999
- SS7 MTP3-User Adaptation Layer (M3UA), RFC 3332, September 2002
- SS7 MTP3-User Adaptation Layer (M3UA), RFC 4666, September 2006
- M3UA Implementor’s Guide, draft-ietf-sigtran-m3ua-implementors-guide-01.txt
- M3UA SG-SG Communication, draft-bidas-sigtran-sgsg-01.txt, September 2002
- Stream Control Transmission Protocol (SCTP), RFC 2960, Oct 2000
- Stream Control Transmission Protocol (SCTP) Implementers Guide, draft-ietf-tsvwg-sctpimpguide-06.txt, May 2002
- Stream Control Transmission Protocol (SCTP) Checksum Change, RFC 3309, September 2002
- SNMPv1, RFC 1157
- SNMPv2 RFC 1905, RFC 1906
- Distributed7™ Manual Set CD-ROM, 27ØØ-1796-Ø1
- Signaling Gateway™ User Manual, 160-2001-01
- Draft RFC 3332 bis, Signaling System 7 (SS7) Message Transfer Part 3 (MTP3)--User Adaptation Layer (M3UA), September 2004.

Chapter 2: Overview

2.1 Chapter Overview

This chapter presents an overview of SS7, SIGTRAN, and Signaling Gateway Client’s architecture and features.

2.1.1 SS7 Overview

Signaling System Number 7 (SS7) is the protocol for sending signals in a digital network instead of using voice connections. This protocol is a set of rules that define the exchange of Switched Circuit Network (SCN) native signaling information between SS7 network elements and the Public Switched Telephone Network (PSTN). The SS7 protocol defines the sequence in which SCN signaling is sent using transmission paths called *links*. Figure 2-1 shows the SS7 network architecture and is followed by brief explanations of the labels shown in the diagram.

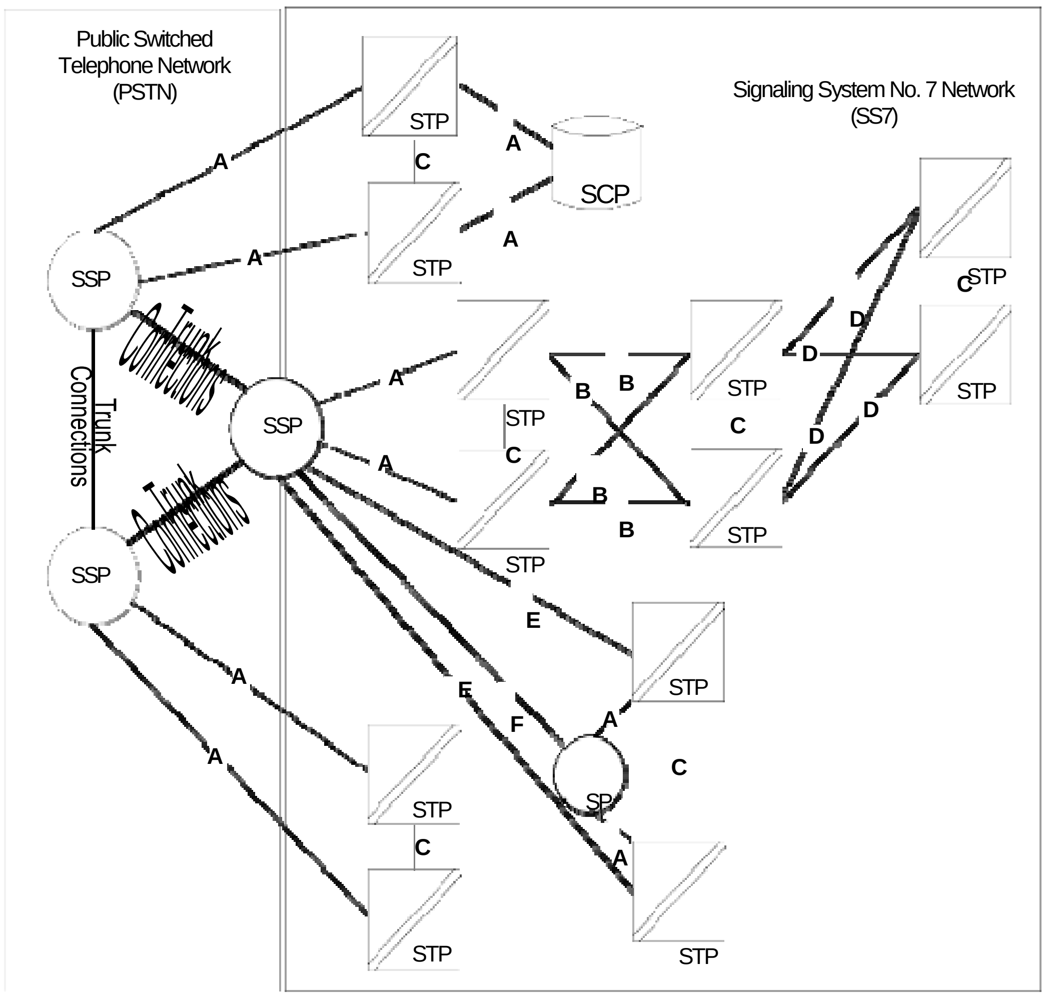

Public Switched Telephone Network (PSTN)
Signaling System No. 7 Network (SS7)
STP
SCP
SSP
SP
Trunk Connections
A
B
C
D
E
F

Figure 2-1: SS7 Network Architecture

The following describes the acronyms and labels used in Figure 2-1:

- **STP** or **Signaling Transfer Point** is the router in an SS7 network that sends and receives SS7 signals to and from other signaling points. STPs are always paired for redundancy, so that if one of the pair is not functioning then the other one handles the load.
- **SP** or **Signaling Point**, is any node in an SS7 network that is capable of sending and receiving SS7 signals. It is a generic term that is used when what a node does is not important. STP, SSP and SCP are all signaling points. STPs must be located in different geographic locations to ensure redundancy, but any other SP may be in the same geographic location as an STP. Each SP always has its own **S**ignaling **P**oint **C**ode (SPC), which is its network address.
- **SSP** or **Service Switching Point**, is a signaling point that handles call set up, stops call processing if necessary, queries databases (such as 8ØØ numbers or credit card verification), and responds appropriately to the result of a database query.

- **SCP** or **Service Control Point**, is a signaling point that acts as the front end of a database by handling queries and sending the answer back to the requesting node. An SCP has a mechanism to retrieve data from a database in a format that the requesting node can use. An SCP may or may not be in the same location as the database.

***Note**: A Signaling End Point, SEP, is any location in the SS7 network that originates and/or terminates SS7 messages. This generic term is useful only to discuss SS7 architecture without the need to define the specific functions of a node. Therefore, other terms (SSP, SCP, IP, etc.) more fully define what happens at the node, but all are SEPs.*

- **Links** are the transmission paths between locations in an SS7 network. They are referred to by the letters **A through F**, and these letters represent what the links do:
  - **A** Links **Access Links** give signaling points *access* to the SS7 network.
  - **B** Links **Bridge Links** connect pairs of STPs in every possible way to create a *bridge* between the two.
  - **C** Links **Cross Links** are the links that connect STP pairs and allow messages to *cross* over from one to the other. They are required to provide an STP's redundancy.
  - **D** Links **Diagonal Links** connect two different levels in an SS7 network in every possible way. The hierarchy in SS7 is conceptual and represented in diagrams by placing STPs at a higher level, above STPs at a lower level. Going from local to regional STPs, or regional to national STPs are examples of SS7's conceptual hierarchy. The links are always *diagonal* lines to connect the two levels, one higher than the other in drawings.
  - **E** Links **Extended Links** connect an SP to an STP in another area to *extend* a signaling point's routing flexibility. An E link provides access to the network like an A link, but is called an E link because it *extends* an SP's routing flexibility beyond the access an A Link provides.
  - **F** Links **Fully Associated Links** connect two locations to specifically isolate their communication from the network so that it is a direct connection.

2.1.1.1 SS7 Protocol Stack

Figure 2-2 is a standard diagram of the first three functional layers of SS7 that deal with a message just about to be transmitted over the links, or one just received from the links. These three levels are Message Transfer Part (MTP) Levels 1, 2 and 3, or MTP1, MTP2 and MTP3. Stacked above the MTP 1, 2 and 3 layers of the User Parts are the Level 4 User Parts, as shown in Figure 2-2:

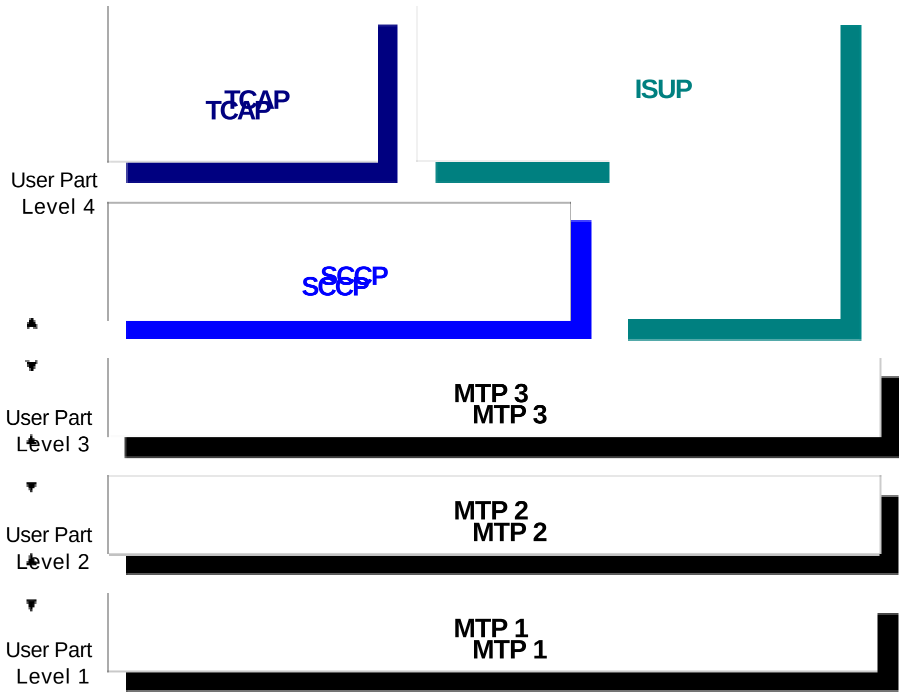


Figure 2-2: SS7 Protocol Stack

Message Transfer Part (MTP)

SS7's MTP layers are the base of the SS7 stack:

- **MTP Level 1** defines the physical, electrical and functional characteristics of the signaling link. This level requires that a link must be bi-directional.
- **MTP Level 2** provides the reliable transfer of signaling information between two adjacent signaling points. It is the last to handle messages being transmitted and the first to handle messages being received. It monitors links and reports on their status, checks the integrity of messages, discards bad messages, requests copies of discarded messages, acknowledges good messages, places links in service and restores links that have been taken out of service. It reports most of what it does to MTP Level 3.
- **MTP Level 3** defines signaling message handling procedures and signaling network management functions. It is responsible for:
  - *Signal Message Handling* controls message discrimination (routing to and from MTP Level 2) and message distribution (routing to and from Level 4 User Parts).
  - *Signaling Network Management* manages traffic, links, routing and congestion flow control.

Level 4 User Parts

The SS7 Level 4 User Parts define the messages and procedures for a particular application running on top of MTP. The following describes the standard Level 4 User Parts:

- **Signaling Connection Control Part** (SCCP) - extends the addressing capability of MTP and establishes non-circuit related signaling connections. It also controls database redundancy.
- **Transaction Capabilities Application Part** (TCAP) - defines non-circuit related protocol over SCCP, such as database queries.
- **Integrated Services Digital Network User Part** (ISUP) - defines the protocol to control circuit-switched services in the Integrated Services Digital Network (ISDN). This part provides the functions for handling telephone-call-related messaging that is sent from switch to switch in a voice network.

The following diagram illustrates one example of how the SS7 stack works with applications.

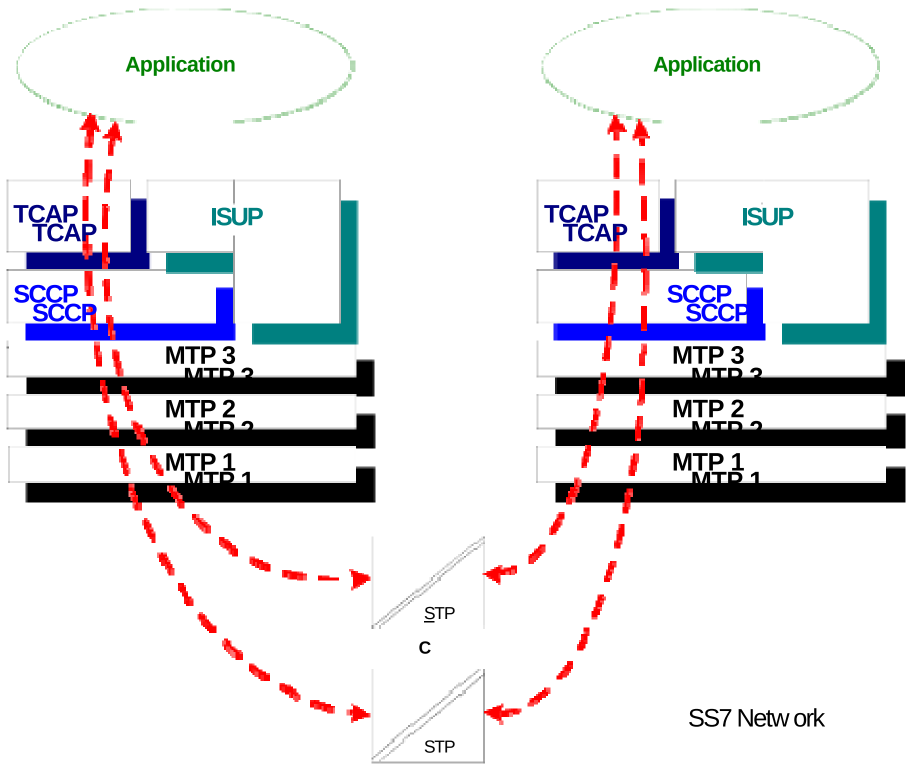


Figure 2-3: SS7 Stack Working with Applications

Note: *Figure 2-3 does not show the message flow passing through ISUP because the example does not involve switching in a voice network.*

2.1.2 SIGTRAN Overview

The Internet Engineering Task Force (IETF) SIGTRAN MTP3 User Adaptation Layer (M3UA) draft protocol defines:

• the conversion of MTP3 messages to M3UA messages, without affecting MTP3 user signaling, such as ISUP and SCCP, and vice versa.

• the procedures for transporting M3UA messages between two end points using Streams Control Transmission Protocol (SCTP). SCTP is a reliable transport protocol operating on top of a connectionless packet network, such as the IP network.

The SIGTRAN protocol uses the concepts and terms listed in the following subsections.

2.1.2.1 Signaling Gateway (SG)

An SG is a signaling agent that receives and sends SCN native signaling between the SS7 and IP networks. It appears to the SS7 network as an SS7 Signaling Point that is addressed with a unique Point Code (PC). It is also represents a set of nodes in the IP domain to the SS7 network for routing purposes between the two networks.

• An SG has one or more Signaling Gateway Processes (SGP), and usually one or more is actively processing traffic.

• An SG is capable of routing SS7 user traffic destined for one or more logical entities, which are referred to as Application Servers (AS) in the IP domain.

2.1.2.2 Signaling Gateway Process (SGP)

An SGP is a process instance of the SG. It uses M3UA and SCTP to communicate with the Application Server Process (ASP) in the IP domain.

2.1.2.3 Application Server (AS)

An AS is a logical entity serving a unique route key. A route key consists of a set of SS7 parameters that define a range of traffic handled by the AS. An example of an AS is a virtual switch element handling all call processing for a unique range of PSTN trunks identified by an SS7 route key combination of SIO/DPC/OPC/CIC range. An AS consists of one or more Application Server Processes (ASP), of which one or more is normally actively processing traffic.

2.1.2.4 Application Server Process (ASP)

An ASP is a process instance of an AS. It uses M3UA and SCTP to communicate with the SGP. An ASP can serve as an active or standby process in an AS. It can also serve more than one AS (i.e., route key).

2.1.2.5 IP Server Process (IPSP)

A process instance of an IP-based application. An IPSP is essentially the same as an ASP, except that it uses M3UA in a point-to-point fashion. Conceptually, an IPSP does not use the services of a Signalling Gateway node.

2.1.2.6 IP Application Server (IPAS)

An IP-based application server. An IPAS is essentially the same as an AS, except that it uses IPSPs instead of ASPs.

2.1.2.7 SCTP Association

The SGP and ASP are viewed as signaling process end points between which an SCTP association is established. The association provides the transport for the delivery of M3UA messages between SGP and ASP.

2.1.2.8 Signaling Point Management Cluster (SPMC)

An SPMC is a complete set of ASs represented to the SS7 network under one specific SS7 Point Code of one specific Network Appearance. The Network Appearance uniquely identifies an SS7 network. See RFC 3332 for more information on the IETF SIGTRAN.

2.1.3 NewNet Communication Technologies' Signaling Gateway Suite

Signaling Gateway Client is part of the NewNet Communication Technologies' Signaling Gateway Suite that includes the Signaling Gateway and the Signaling Gateway Client. Together, the two products provide a complete SS7 over IP interworking solution. Both the Signaling Gateway and Signaling Gateway Client are ANSI/ITU SS7 and IETF SIGTRAN compliant.

- Signaling Gateway is a signaling gateway between the SS7 and IP networks and it extends MTP service to remote SS7 User Parts using M3UA over SCTP. Please see the *Signaling Gateway^{TM} User Manual* for more information.
- Signaling Gateway Client provides SS7 User Part services, such as ISUP, SCCP and TCAP, to IP-based applications such as the Media Gateway Controller (MGC) or an IP-based Home Location Register (HLR). The Signaling Gateway Client solution offers user applications in the IP domain the ability to use SS7 services without having to focus on protocol conversion.

Although part of the Suite, Signaling Gateway Client can also be deployed without Signaling Gateway, it is designed to integrate with an SS7 network that is ANSI/ITU compliant, and with any signaling gateways or signaling gateway clients that are IETF SIGTRAN compliant.

The following illustrates the Signaling Gateway Suite's internetworking solution for communicating between the SS7 and IP networks:

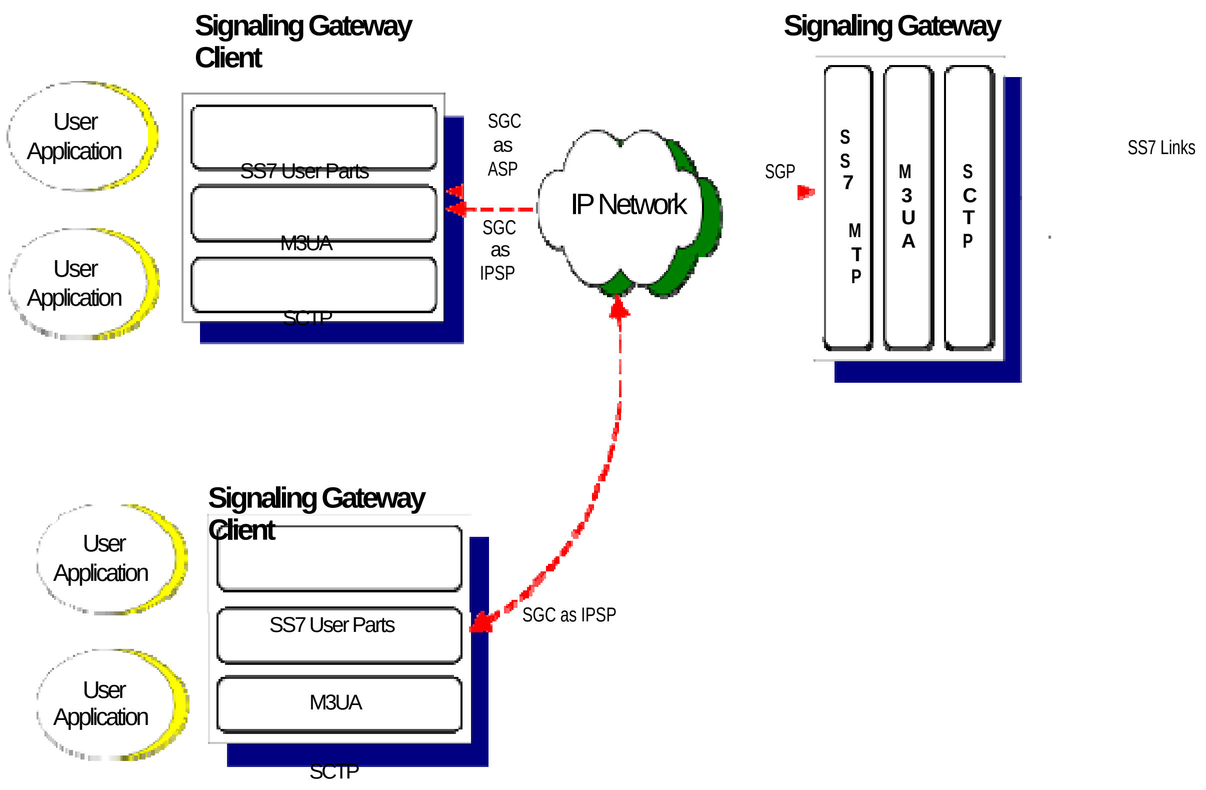


Figure 2-4: NewNet Communication Technologies' Signaling Gateway Suite

2.2 Signaling Gateway Client Features and Capabilities

2.2.1 SS7-IP Interworking

The following figure illustrates the SS7-IP interworking solution.

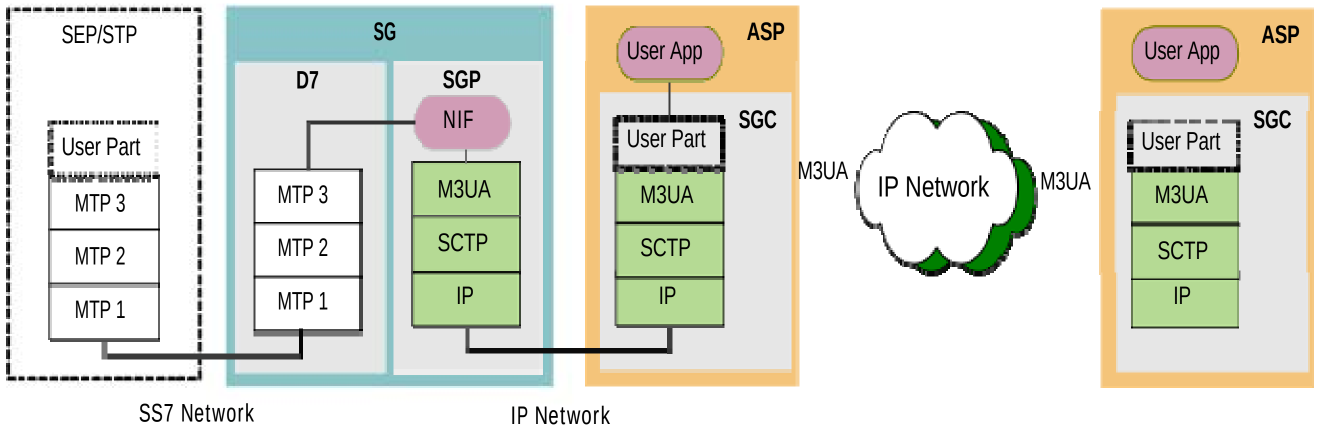


Figure 2-5: SS7-IP Interworking

2.2.2 Message Distribution

2.2.2.1 Address Translation and Mapping

The SGC maintains an internal routing table consisting of routing keys with references to the ASs. A routing key is unique for an Application Server within the SGC's context. The operator provisions the routing keys through the administrative interface as explained in the following sections. The routing key defined for an AS can be one of the following combinations:

- DPC
- DPC - SIO
- DPC - SIO - CIC
- DPC - OPC
- DPC - OPC - SIO
- DPC - OPC - ISUP CIC

The operator can define one or more ranges for ISUP CIC, and OPC fields in a single route key.

2.2.2.2 Traffic Distribution

IP to MTP3 User Part

Incoming messages received at the ASP are translated into MTP3 User Part primitives through M3UA. Based on the message content, the AS value is determined for the received message. The message is converted to the Distributed7 format and forwarded to the user part running on the SP which matches the calculated AS value. In the distributed mode any ASP is capable of receiving the traffic from IP direction. Each ASP follows the same procedure and forwards the incoming messages to the MTP3 user parts. Please note that an ASP can receive messages from either gateways (SGP) or other clients (IPSP).

MTP3 User Part to IP

Outgoing MTP3 User Part messages are dispatched to an ASP instance. The ASP then finds out the RK (Routing Key) value for the outgoing message. Every provisioned ASP RK either points to a SGP or an IPSP. If the message is destined for an SGP, it is sent to the SGP or forwarded to another ASP if the DPC of the message is not reachable from the ASP itself. In the case of an RK provisioned for an IPSP, the message is routed towards that IPSP.

ASP Selection

In the standalone mode there is always one active ASP instance so that the selection of the ASP is straightforward.

On the other hand, in the distributed mode there are multiple ASPs actively processing traffic. First, the selection of the ASP should ensure a loadsharing mechanism while preserving the message ordering when necessary. Also, at any given time, each ASP may have a different SG (IPSP) connectivity view. This may be caused by a failed connection from an ASP to a particular SGP (IPSP), or the connectivity of an ASP to a particular SGP (IPSP) may have been taken down for administration purposes. Whatever the reason, there is a possibility that each ASP may be processing different ranges of SS7 traffic at any given time. Hence, an ASP instance that is capable of handling the message should be selected.

Signaling Gateway Client is configured to direct the outgoing MTP3 User Part primitives to the closest ASP instance. This ensures the sequenced delivery of the outgoing messages. In other words, the messages from a particular user part process are always forwarded to the same ASP. This approach avoids the rerouting of the messages to the extent possible, guarantees sequenced delivery of the messages, and distributes the load evenly among the cluster members. Nevertheless, if an ASP is unable to process a certain range of traffic as described in the previous paragraph, the messages are forwarded to an alternative ASP, which can handle the messages.

ASP's internal routing function makes sure that the messages are always forwarded to the same ASP instance, as long as that instance is available. Also, the routing function maintains the reachability information for each SGP (IPSP) instance dynamically. If the routing function fails to locate an ASP instance for a particular message, that message is discarded and an alarm is generated.

SG Selection

Once the message is received at the ASP, it is forwarded to an appropriate SG according to the DPC reachability status.

An SG contains a set of SGPs processing traffic for the same set of SS7 DPCs. These SGPs may work in override or loadsharing mode. The mode of operation is configurable at the SGC. Depending on the mode of operation, configuration changes, fail-over mechanisms and availability status of the SGPs of an SG changes dynamically. The SGC monitors the status of all SGPs.

When SGPs are working in override mode, the SGC always directs the SS7 traffic to the active SGP. When SGPs are working in loadsharing mode, it distributes the traffic to the actively available SGPs. The distribution is done according to an algorithm based on the SLS value. Hence, the ordered delivery of the messages to the SGPs is guaranteed.

Also, there may be more than one SG provisioned for the same set of DPCs. In that case SGC is capable of selecting the SG depending on the configured selection method. That selection method may be loadshare, which tells the ASP to distribute the traffic to different SGs evenly, or it may be override method, which tells the ASP to send the traffic to the primary SG as long as it is alive.

2.2.3 Reliability

The two most important aspects of reliability are high availability and fault tolerance.

2.2.3.1 High Availability

Signaling Gateway Client offers high availability by distributing traffic and configuration data over multiple hosts:

- User Part distribution - User Part are distributed across all hosts. When User Part service is not available in one of the hosts, traffic is automatically rerouted to the other host. Necessary information is synchronized among the hosts to prevents the loss of existing calls and transactions.
- Multiple ASPs - IPSP and ASP are logical functions of an SGC. Each SGC can act as an ASP and/or as an IPSP, so when we are talking about one SGC instance on a particular host, it acts as an ASP when interfacing with an SG (SGP) but acts as an IPSP when interfacing with another SGC (IPSP).
  Each host in an Signaling Gateway Client cluster is seen as a separate ASP or IPSP entity as seen from the M3UA. Multiple ASPs or IPSPs serving the same AS or IPAS result in redundancy on the M3UA level.
- Configuration distribution - all configuration databases are replicated and synchronized across all hosts, and are accessible from any host in the cluster through the OAM interface.

2.2.3.2 Fault Tolerance

Signaling Gateway Client is fault tolerant in the following ways:

- Redundant LAN Cluster - Dual LAN interface is provided for cluster communication to increase redundancy. Communication is possible through the redundant LAN should any LAN fail.
- SCTP Multihoming - SCTP supports multihoming where multiple IP addresses can be defined for each SGP-ASP association. The redundant interface prevents single-point-of-failure should any IP interface fail.
- Multiple ASPs and internal message routing among ASPs - when an ASP loses its connection to the SGP, messages are rerouted to other ASPs. This avoids a single-point-of-failure should any ASP lose its SCTP connection.

2.2.4 Fault Management

Signaling Gateway Client has the following fault management functions:

- Application Process Management (APM) - the APM feature provides automatic coordination of platform startup and shutdown. APM also provides continuous process monitoring such that whenever a process fails, it is restarted automatically and instantaneously without human intervention.
- Automatic startup - automatic application startup can be enabled so that all application processes start automatically following a system reboot. This is useful when a power failure occurs.
- Alarm reporting - all components of Signaling Gateway Client have built-in fault detection. An alarm is generated and the faulty event is logged whenever a software, operating system or network failure is detected. Alarm reporting can be configured so that the alarm message is sent either to the console, log file, or as an SNMP trap to a management station.

2.2.5 Troubleshooting Tools

Signaling Gateway Client has alarm reporting, daily event log file and runtime tracing

utility
for troubleshooting.

2.2.6 Security

Signaling Gateway Client has the following security features:

- **User Access** - restricted system access is provided through the UNIX password-protected user account. Only the administrator or the *superuser* has the privileges to administer and maintain the system.
- **Transport Security** - the use of SCTP allows the support of transport related security features such as protection against Blind Denial of Service Attacks, flooding or blind masquerade.

When the network involves more than one party, it is recommended to use IPsec to ensure confidentiality of user payload.

2.3 Signaling Gateway Client Interfaces

Signaling Gateway Client runs on a distributed platform that supports up to eight hosts, which form a cluster. Figure 2-6 illustrates the four interfaces of Signaling Gateway Client:

- User application interface (A)
- SCTP network interface (B)
- Cluster LAN interface (C)
- OAM interface (D)

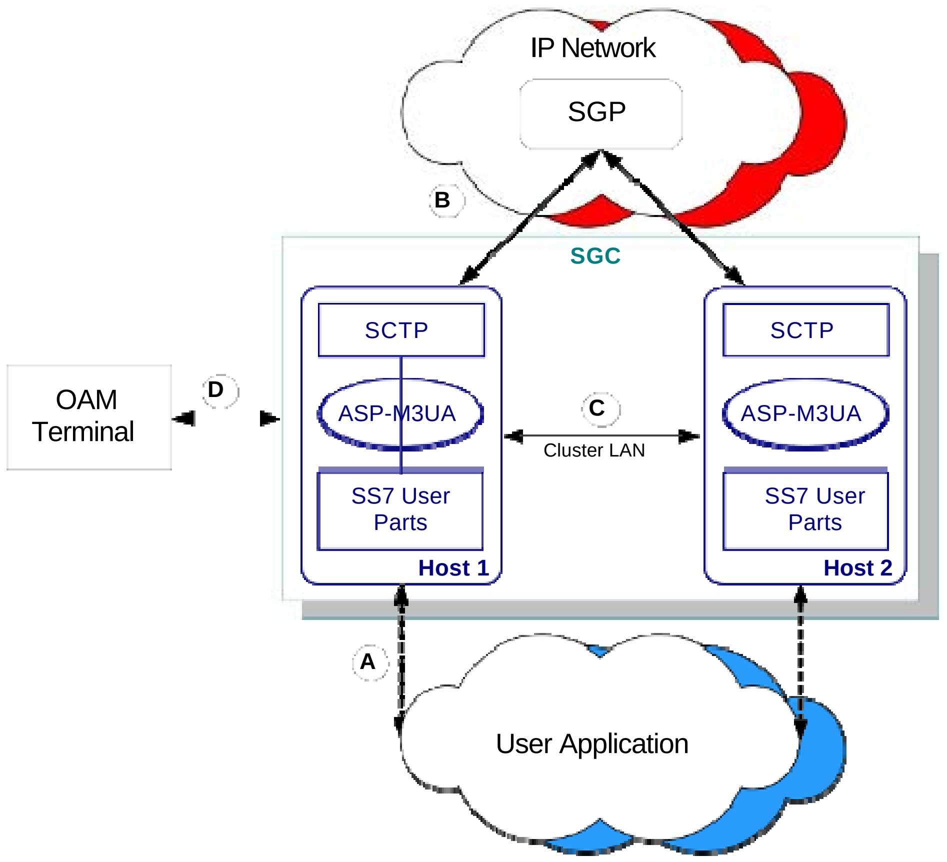


Figure 2-6: Signaling Gateway Client Interfaces

2.3.7 User Application Interface

User applications use the Distributed7 APIs to get the platform services. The services include the SS7 communications (MTP3, ISUP, TCAP, SCCP), alarms, logging, tracing, distributed service, enhanced IPC, provisioning, and other services as explained in [14]. SGC does not impose any changes in the existing Distributed7 APIs. Hence existing Distributed7 applications run on SGC without modification. From the operational perspective, the only change for a Distributed7 user is to replace the MTP provisioning with Signaling Gateway Client provisioning.

The following Distributed7 APIs are available to the user: MTP, ISUP, TCAP, SCCP, GSM MAP, DSM, DKM, DRA, OAM, SPM, and Alarm.

2.3.8 SCTP Interface

Signaling Gateway Client communicates with the SGPs through SCTP. SCTP provides reliable transport of packets over IP. IPsec, which is supported by Solaris OS, may be used for enhanced security.

With the multihoming support provided by SCTP, an SGP and an ASP can communicate over an SCTP association using more than one IP address. Any failure of an IP address causes a fail-over to alternative addresses. Each host can have multiple IP addresses specified for this purpose.

2.3.9 OAM Interface

Signaling Gateway Client provides OAM interfaces for system configuration and administration. Protocol parameters and application attributes are modeled as Managed Objects (MO) and can be managed through the following interfaces:

- Man Machine Interface (MMI) - a text-based interface based on the standard ITU Z-315 MML syntax. OAM functions can be performed from a system console, such as a dumb terminal, connected to the serial port of the host, or from a remote virtual terminal connection such as a telnet or rlogin session.
- Simple Network Management Protocol (SNMP) versions 1 and 2 - OAM functions can be performed from an SNMP management console.
- AccessMOB - A GUI tool that enables configuration of the MOs.

In addition, there are several utilities and scripts provided for system startup, shutdown, software upgrade, tracing, etc.

2.3.10 Cluster LAN Interface

The distributed hosts in the cluster are connected to each other through a private LAN based on TCP/IP. The distributed hosts exchange data related to service distribution, database synchronization and management messages through the cluster interface.

For added redundancy, the cluster interface is made up of a dual LAN network, in which each host has access to both LANs through separate Ethernet interfaces.

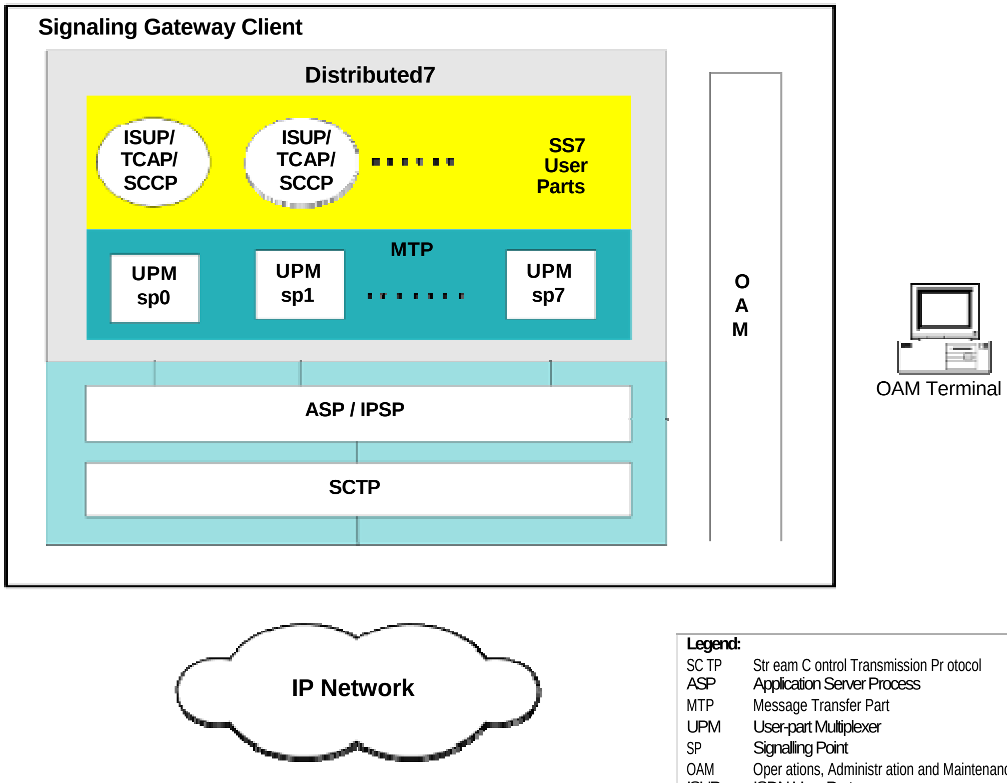
It is recommended that the cluster LAN be separated from the SCTP interface or any other network to be sure that the performance and integrity of the distributed platform is not affected.

2.4 SoftwareComponents

The Signaling Gateway Client system has the following software components:

- Distributed7 - a distributed, open-architecture, high-performance, real-time and scalable SS7 application development platform.
- Signaling Gateway Client Core - an IETF SIGTRAN-based signaling gateway client that provides M3UA and SCTP services for transporting SS7 user traffic to and from M3UA-based SGs.

Signaling Gateway Client runs on the Sun Solaris 10 operating system. The following illustrates the high-level software architecture:



Legend:

SCTP	Stream Control Transmission Protocol
ASP	Application Server Process
MTP	Message Transfer Part
UPM	User-part Multiplexer
SP	Signalling Point
OAM	Operations, Administration and Maintenance
ISUP	ISDN User Part
SCCP	Signalling Connection Control Part
TCAP	Transaction Capabilities Applications Part
IPSP	Internet Protocol Security Protocol

Figure 2-7: Signaling Gateway Client Software Architecture

2.5 Capacity

Signaling Gateway Client supports a distributed, load sharing platform over a maximum of
eight hosts. The following table lists the various capacity factors:.

Table 2-1: Capacity

	Description	ANSI/ITU
MTP	destination point codes (SS7 + capability)	2048
	destinations behind link sets	2048
	links	511
	link sets	64
	routes per destination	16
SCCP	destination point codes	8192
	subsystems per destination	256
	concerned point codes per subsystem	8192
	global title types	16
	translation types per global title type	256
	simultaneous open SCCP connections	16384
	simultaneous reassembly processes per system	16
ISUP	destination point codes	2048
	circuit groups per destination	3040
	total trunk groups for all destinations	8192
	circuits per circuit group	32
TCAP	simultaneously open dialogues	262144
	local instances of the same subsystem	31
INE	signalling points	8
IPSP	Maximum number of IPSPs	256
	Maximum number of IPASs	2048
	Maximum number of IPSPs per IPAS	5
SYSTEM	Maximum number of cluster hosts	8
	Maximum number of IP addresses per host for SCTP interface	2
	Maximum number of AS (only 8 DPCs can be used during AS configuration)	2048
	Maximum number of SGP peers	16

Chapter 3: Distributed7 Overview

3.1 ChapterOverview

This chapter provides an overview of the Distributed7 software platform.

3.2 GeneralDescription

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 3-1).

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.

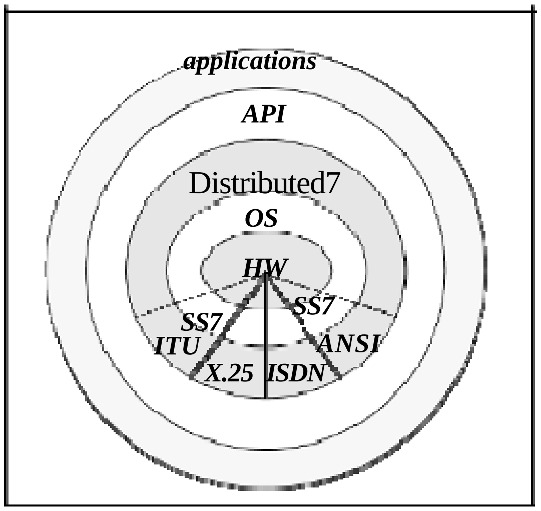


Figure 3-1: Distributed7 Layered Architecture

3.3 Features

3.3.1 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.

3.3.2 Common Channel Signaling System No. 7

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

3.3.2.1 Message Transfer Part

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

3.3.2.2 Signaling 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.

3.3.2.3 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.

3.3.2.4 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 signaling. The Distributed7/ISUP supports all signaling 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.

3.3.2.5 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.

3.3.2.6 Distributed SS7 Stack Operations

The Distributed7 software provides the functionality for distributed systems operations in the MTP, SCCP, TCAP, and ISUP Layers.

3.3.3 Platform Services

3.3.3.1 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

Related Information

- Chapter 2: SPM API Programming Guide in the *Signaling Gateway Client Client Development Manual*

Registration

In the Distributed7 environment, applications establish a service end point through a selected module and bind an address to that service end point—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.

Related Information

- Chapter 2: SPM API Programming Guide in the *Signaling Gateway Client Client 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 signaling 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 signaling 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.

3.3.3.2 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.

3.3.3.3 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.

3.3.3.4 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:

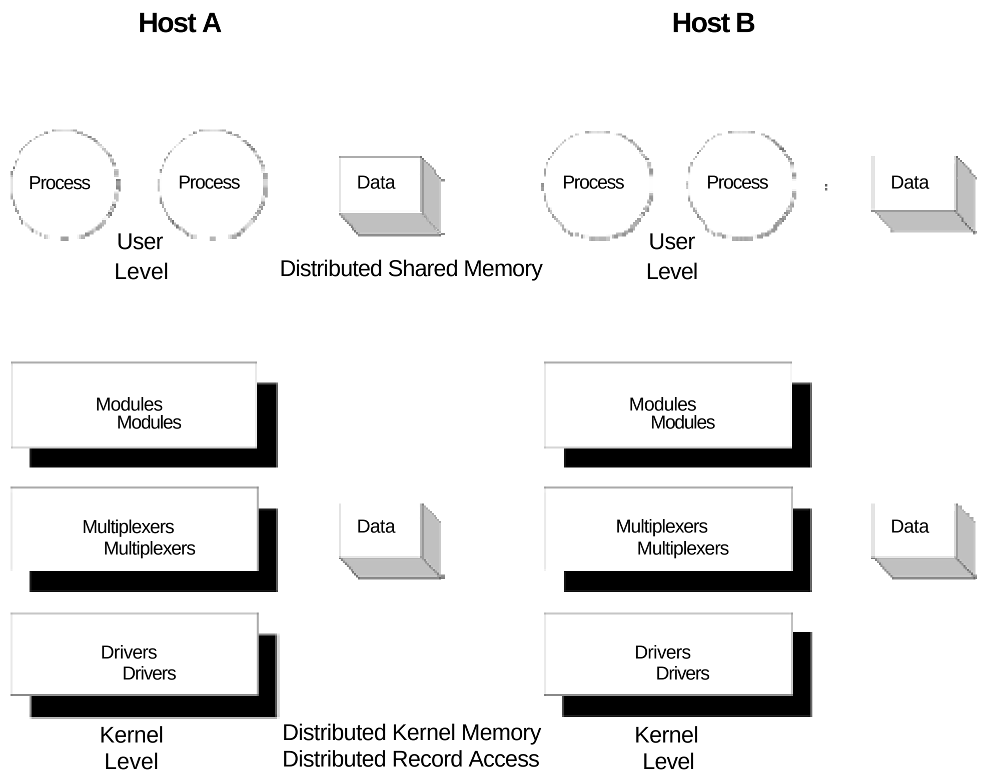


Figure 3-2: Distributed Memory Management

- 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.

3.3.3.5 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.

3.3.3.6 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.

3.3.3.7 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.*

3.3.3.8 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.

3.3.3.9 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

3.3.3.10 Backward Compatibility

Distributed7 API Libraries are largely backward compatible with earlier releases of the Distributed7 product.

3.3.4 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.

3.4 Architecture

Figure 3-3 illustrates the Distributed7 high-level platform architecture.

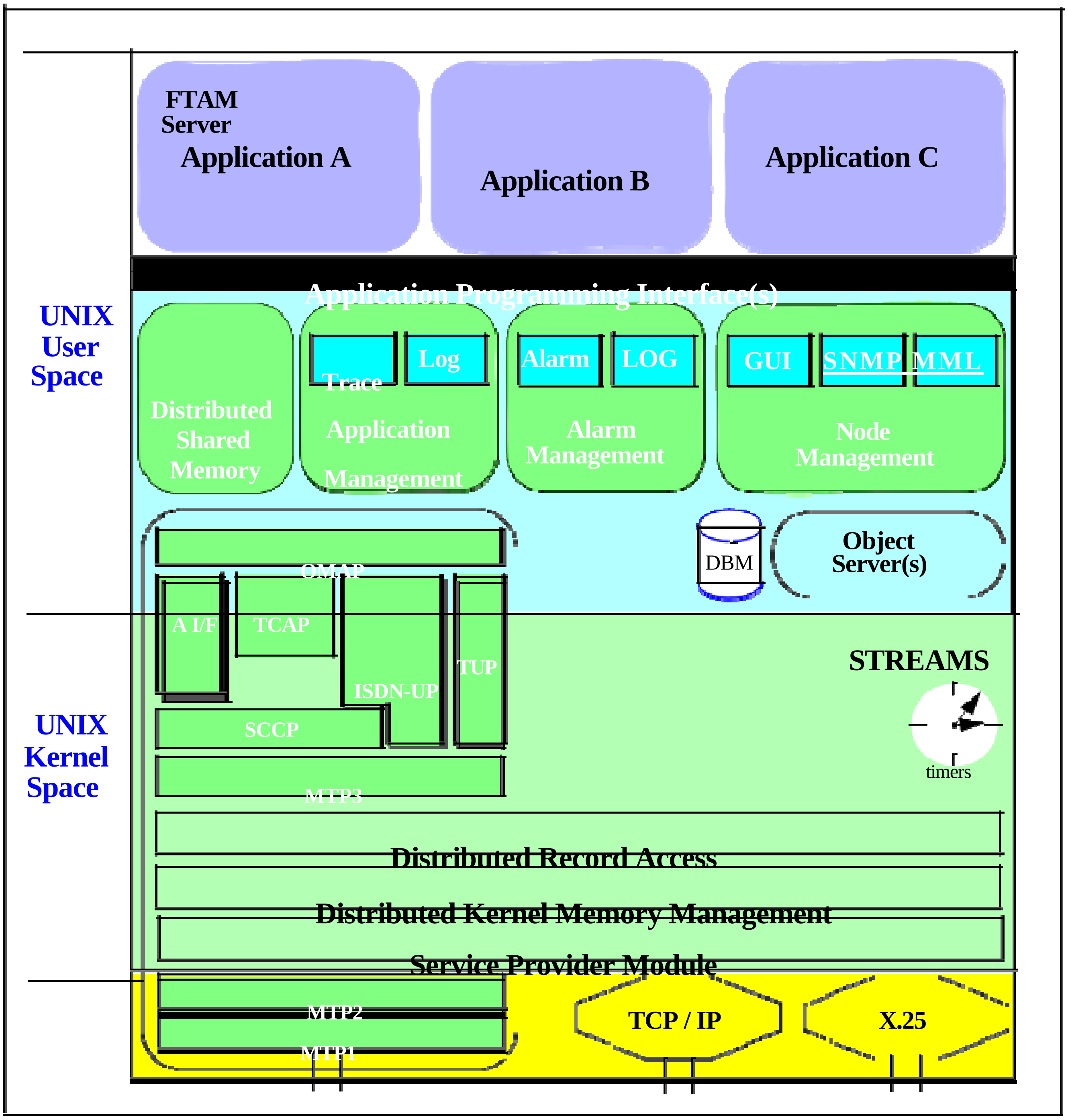


Figure 3-3: Distributed7 Software Architecture

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 Management

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 signaling data link software.

MTP-L3

The MTP Level 3 is implemented as a STREAMS driver that performs signaling message handling and signaling 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 signaling 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, signaling 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.

3.4.1 SPM API Library

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

3.4.2 APM API Library

The APM API Library provides the application process management capabilities.

3.4.3 DSM API Library

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

3.4.4 OA&M 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 signaling point operation.

3.4.5 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.

3.4.6 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.

3.4.7 SCCP API Library

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

3.4.8 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.

3.4.9 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.

3.4.10 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.

3.4.11 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.

3.4.12 Gateway API Library

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

3.4.13 IS41-D API Library

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

3.4.14 GSM MAP API Library

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

3.4.15 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).

3.5 Distributed7SystemApplications

The Distributed7 software environment provides all the signaling 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 Signaling Interfaces to media servers providing voice, fax, and video services
- Protocol converters and gateways

3.5.1 Media Server Network Signaling 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.

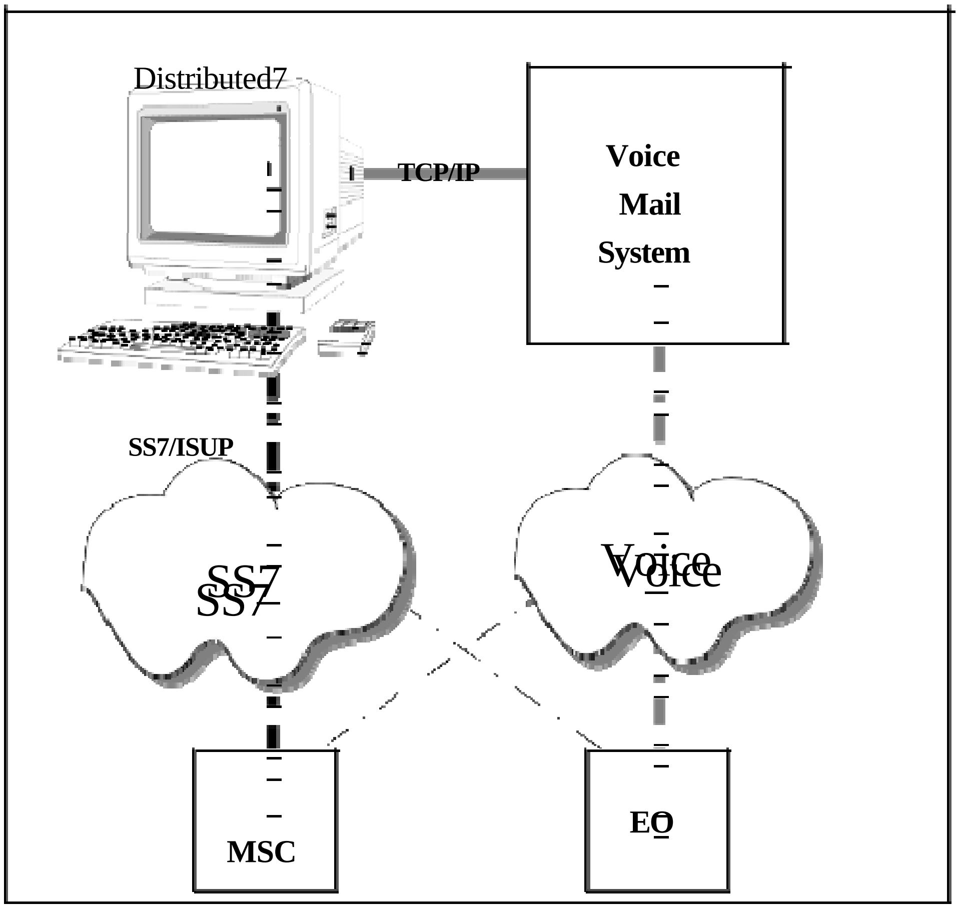


Figure 3-4: Media Server Network Interface

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 3-4.

3.5.2 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.

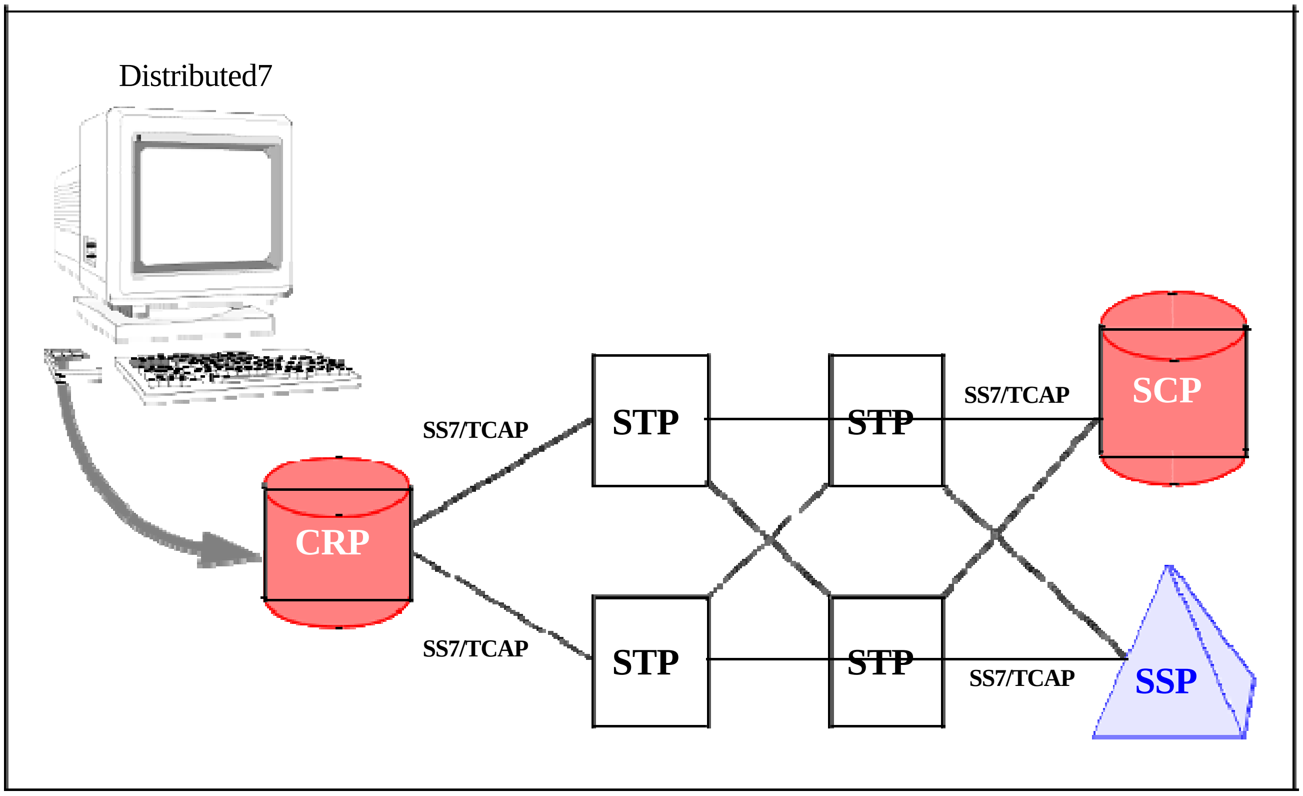


Figure 3-5: Distributed7 as a Customer Routing Point

Customers can use the Distributed7 platform to build a CRP, as shown in Figure 3-5. 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.

3.5.3 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 3-6 depicts the high-level software architecture of SMServer.

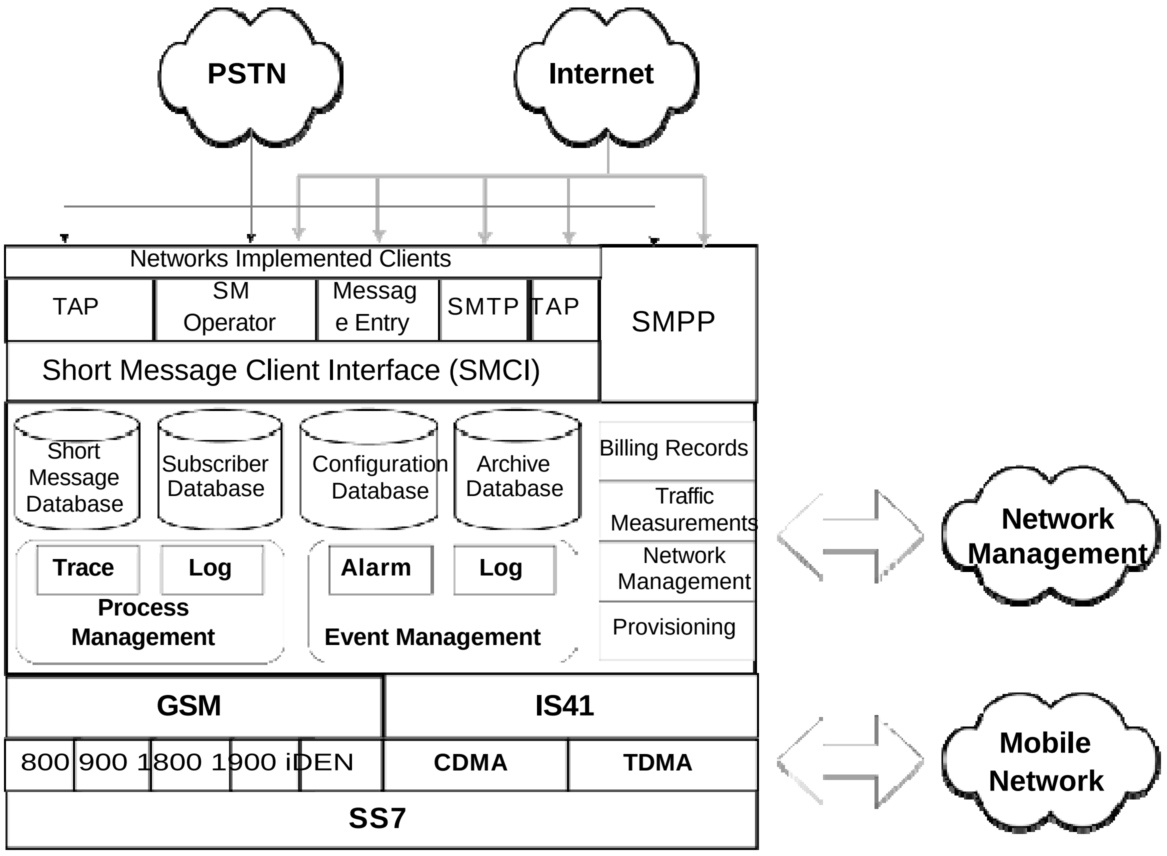


Figure 3-6: SMServer Software Architecture

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.

3.5.4 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.

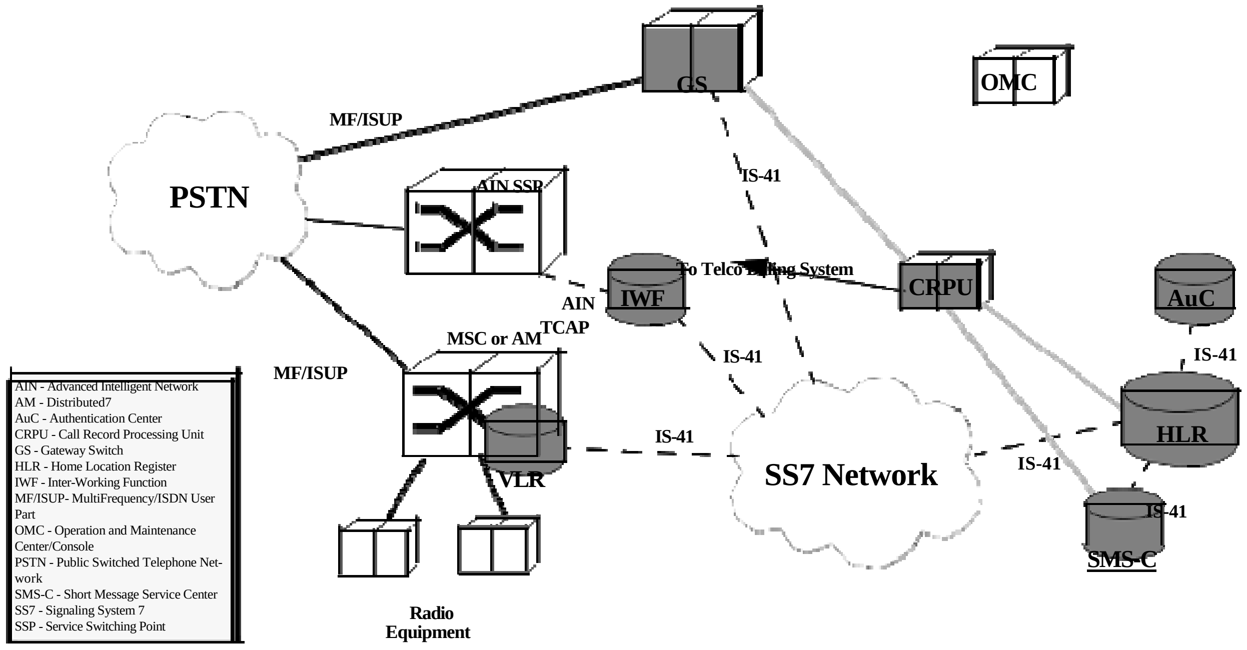

GS
OMC
MF/ISUP
PSTN
AIN SSP
IS-41
To Telco Billing System
CRPU
AuC
IWF
AIN
TCAP
MSC or AM
MF/ISUP
IS-41
IS-41
HLR
VLR
IS-41
SS7 Network
IS-41
IS-41
SMS-C
Radio Equipment
AIN - Advanced Intelligent Network
AM - Distributed7
AuC - Authentication Center
CRPU - Call Record Processing Unit
GS - Gateway Switch
HLR - Home Location Register
IWF - Inter-Working Function
MF/ISUP- MultiFrequency/ISDN User Part
OMC - Operation and Maintenance Center/Console
PSTN - Public Switched Telephone Network
SMS-C - Short Message Service Center
SS7 - Signaling System 7
SSP - Service Switching Point

Figure 3-7: 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

3.5.5 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.

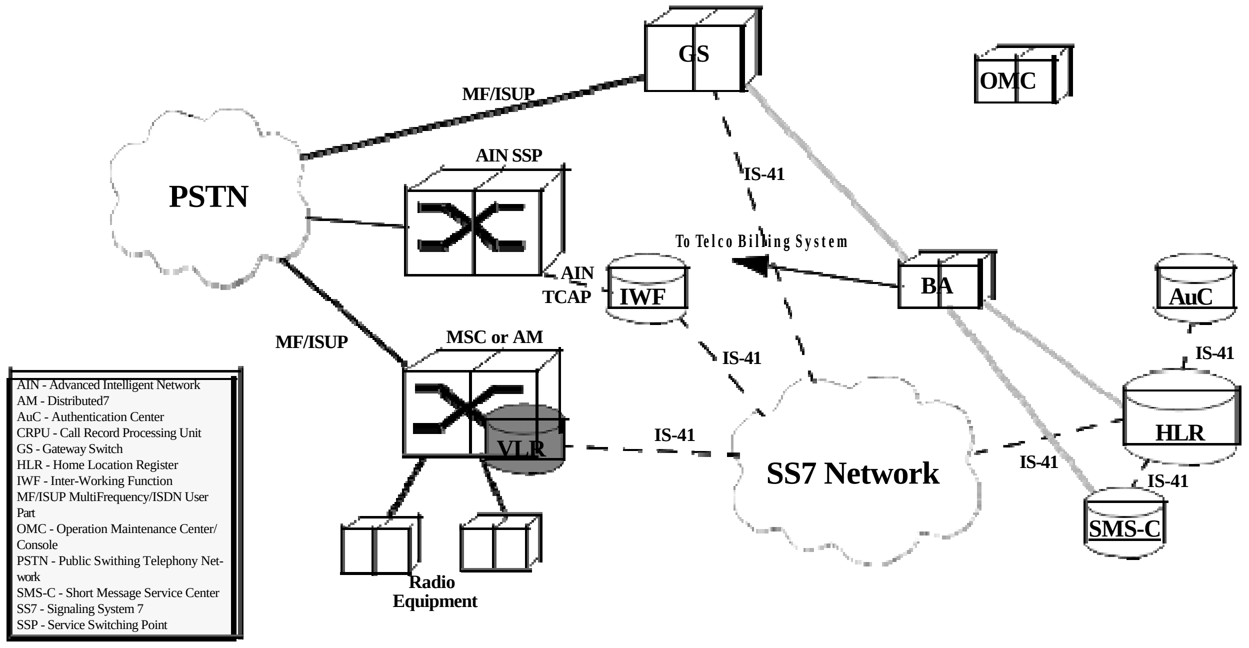

GS
OMC
MF/ISUP
AIN SSP
IS-41
PSTN
To Telco Billing System
AIN TCAP
IWF
BA
AuC
IS-41
MF/ISUP
MSC or AM
IS-41
HLR
IS-41
VLR
SS7 Network
IS-41
IS-41
SMS-C
Radio Equipment
AIN - Advanced Intelligent Network
AM - Distributed7
AuC - Authentication Center
CRPU - Call Record Processing Unit
GS - Gateway Switch
HLR - Home Location Register
IWF - Inter-Working Function
MF/ISUP MultiFrequency/ISDN User Part
OMC - Operation Maintenance Center/ Console
PSTN - Public Swithing Telephony Network
SMS-C - Short Message Service Center
SS7 - Signaling System 7
SSP - Service Switching Point

Figure 3-8: 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.

3.6 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 3-1: 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
Interface	IS-41-D	GSM MAP	GSM A		
Document Number	TIA/EIA-41-D	ETSI 09.02 version 4.11	ETSI GSM 04.01 - 04.08		
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 3-2: Additional 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 3-3: Interfaces

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

3.7 Capacity and Configuration Options

3.7.1 SS7 Database Capacity

Table 3-4 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 3-4: 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	16384	16384
	local instances of the same subsystem	31	31
INE	signaling points	8	8

3.7.2 Host Platform Options

For host platform options, refer to the SGC and SG 1.5.0 Release Notes.

3.7.3 SS7 Controller Options

Table 3-5: 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]
Sbus	RS-449	Up to 4		8
	V.35	Up to 4		
	T1	Up to 4		
	E1	Up to 4		
PCIbus	RS-449	Up to 4		
	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]		

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.*

3.7.4 External Dependencies

All the executable modules listed in Table 3-6 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 3-6: Executable External Dependencies

Module	Shared Libraries	Environment Variables
AccessAlarm	C++ library (provided byNewNet Communication Technologies, Inc. from Sun-Soft) (libC)	
AccessISUP	C++ library (provided by NewNet Communication Technologies, Inc. from Sun-Soft) (libC)	
AccessMOB	C++ library (provided by NewNet Communication Technologies, Inc. from Sun-Soft) (libC) Motif library (libXm) (not provided by NewNet Communication Technologies, Inc.), X Windowing System Libraries (libX11, libXt, libXext) (provided by o/s vendor)	MOTIFHOME OPENWINHOME DISPLAY
AccessOMAP	C++ library (provided by NewNet Communication Technologies, Inc. from Sun-Soft) (libC)	
AccessSNMP	C++ library (provided by NewNet Communication Technologies, Inc. from Sun-Soft) (libC)	
AccessStatus	C++ library (provided by NewNet Communication Technologies, Inc. from Sun-Soft) (libC) Motif library (libXm) (not provided by NewNet Communication Technologies, Inc.), 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, Inc. from Sun-Soft) (libC)	
scmd	C++ library (provided by NewNet Communication Technologies, Inc. from Sun-Soft) (libC)	

Table 3-6: Executable External Dependencies (Continued)

Module	Shared Libraries	Environment Variables
MML	C++ library (provided by NewNet Communication Technologies, Inc. from SunSoft) (libC)	

The libraries listed in Table 3-7 require the associated shared library. These libraries are used with the OA&M and/or Object Server APIs.

Table 3-7: Library External Dependencies

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

Chapter 4: Concepts

4.1 ChapterOverview

This chapter provides a general overview of SS7, the Signaling Gateway Client managed object concepts, and the Product Specifications.

4.2 Managed Objects and the Object Server

The resources of the Signaling Gateway Client 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 belonging 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.

4.2.1 Object Server

The Object Server is a module containing all the Signaling Gateway Client 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 4.2.1.1 on page 4-53*.

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 4-1.

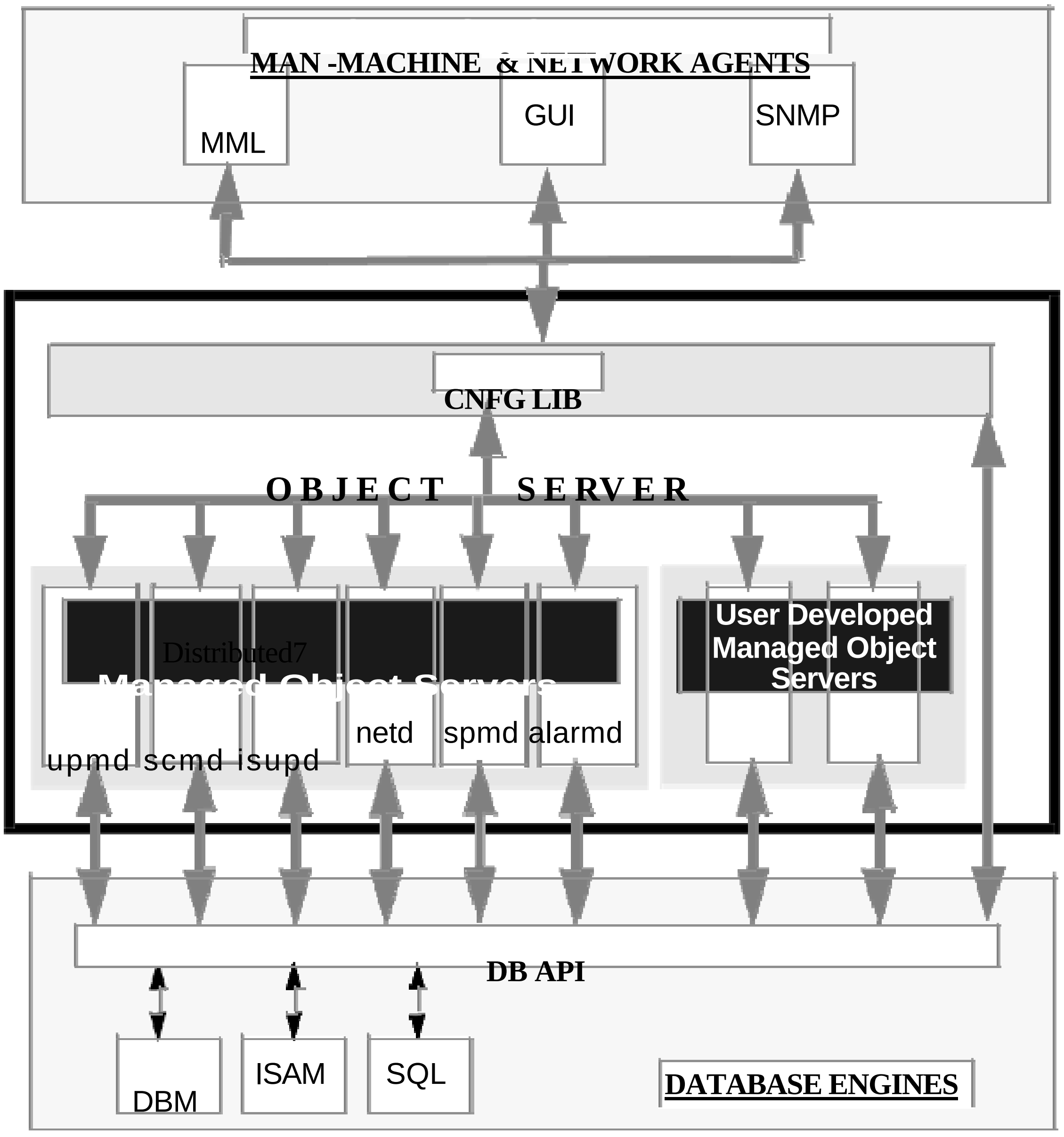


Figure 4-1: Object Server Internal Architecture

4.2.1.1 MO Groupings

Figure 4-2 and Figure 4-3 show the MO *parent-child* hierarchy in the Signaling Gateway Client 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.

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.

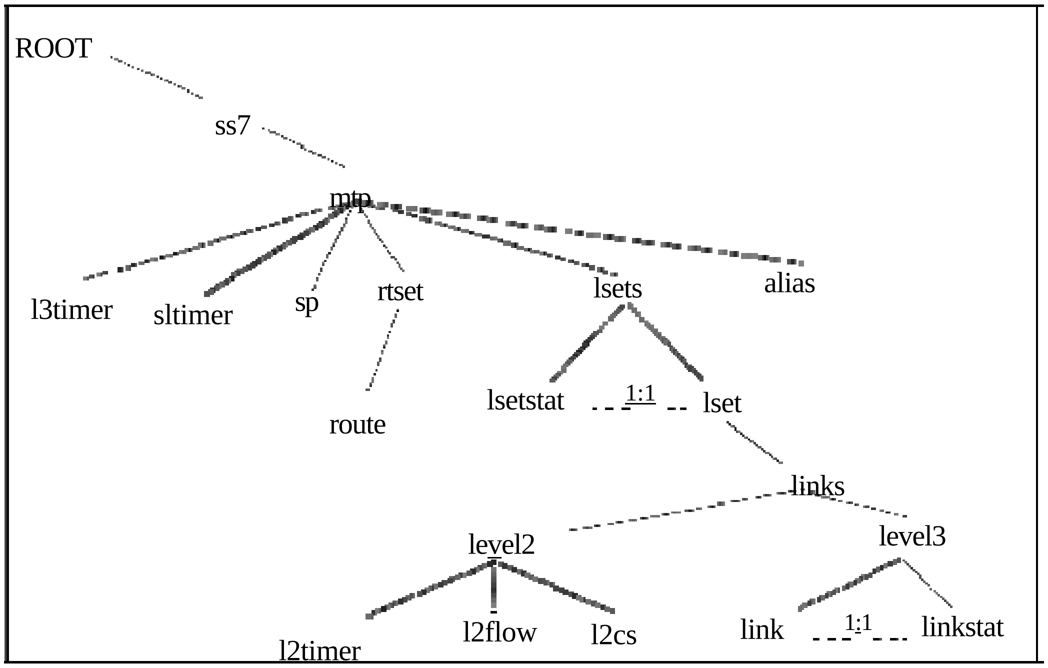


Figure 4-2: MTP Managed Object Containment Structure

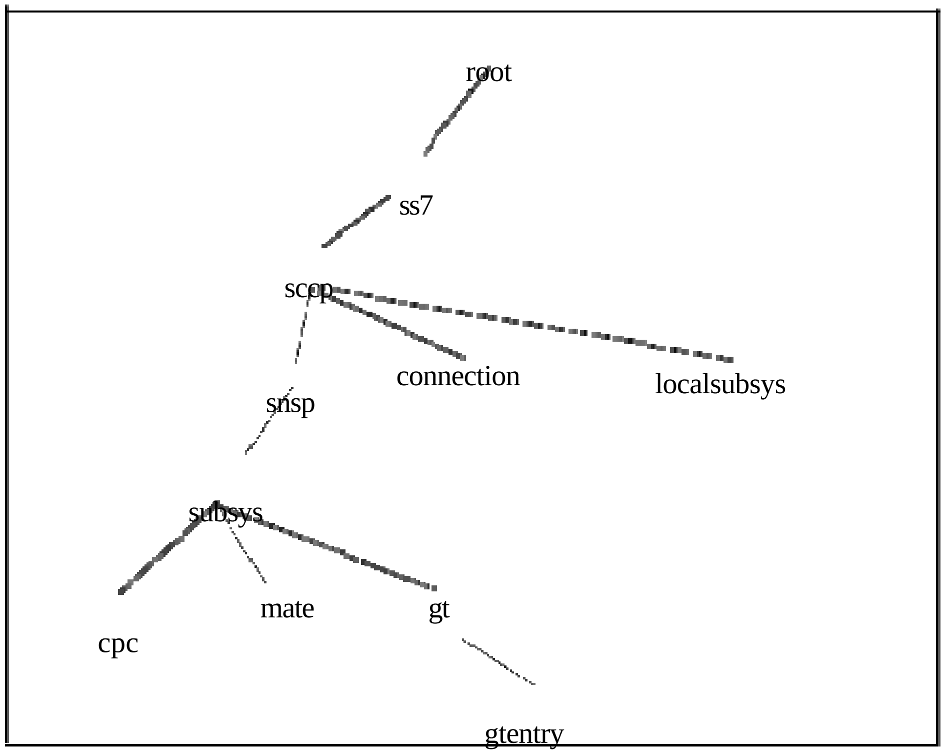


Figure 4-3: SCCP Managed Object Containment Structure

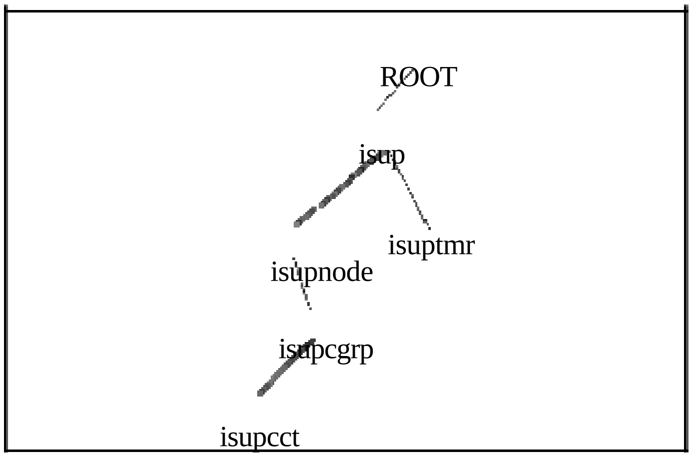


Figure 4-4: ISUP Managed Object Containment Structure

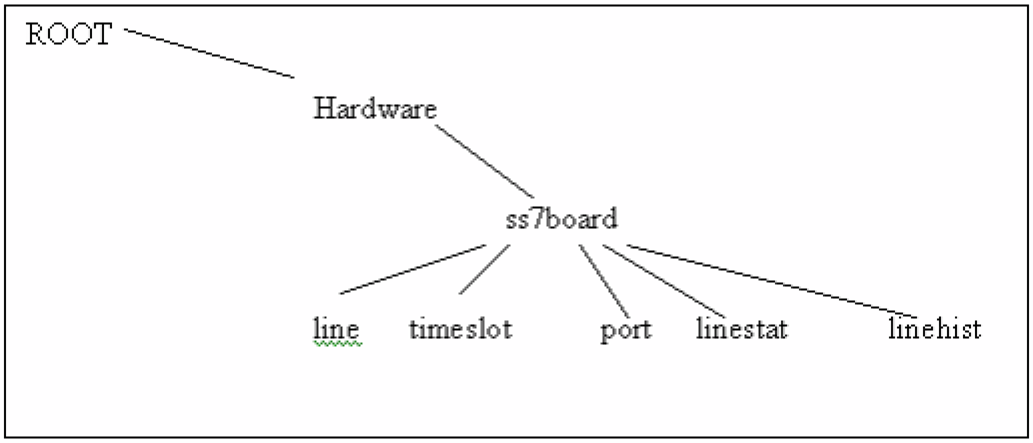


Figure 4-5: SPM Managed Object Containment Structure

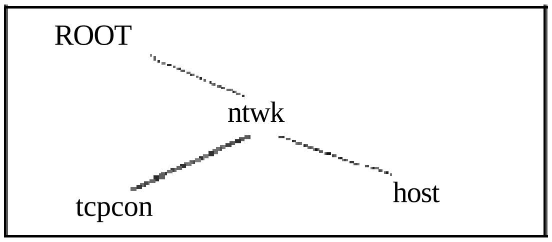


Figure 4-6: Network Managed Object Containment Structure

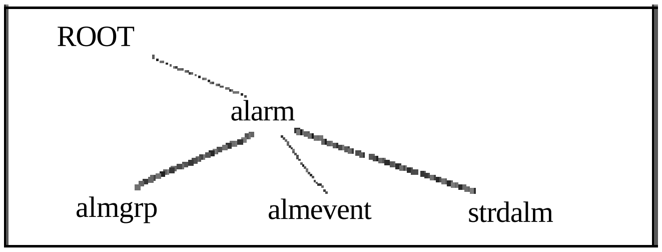


Figure 4-7: Alarm Managed Object Containment Structure

4.2.2 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
- Modfy
- 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 Signaling Gateway Client 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.

4.2.2.1 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 4-4 and Table 4-5 list the parameters and the operations for the Signaling Gateway Client 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 Ø, 1, 2, and 3.

The **Access** column in Table 4-1 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 4-1: SPM Branch Managed Object Descriptions

Managed Object	Parameters	Types	Access	Set Values	Operations
hardware	N/A	N/A	N/A	N/A	N/A
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	
	inst	Integer	read-create	-	
	conf	Set	read-write	ON/OFF	
	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	
	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	
	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 4-1: 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	
	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	
	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	
	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	
	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 4-1: 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	
	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	-	DISPLAYM ODIFY
	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 4-1: 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 4-2: 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	-	
	relilience	Integer	read-write	-	
host	hostname	String	read-create	-	ADD DELETE DISPLAY MODIFY
	rmthost	String	read-create	-	
	alias	String	read-create	-	
	rmthosttyp	Set	read-write	AMGR/OTHER	

Table 4-2: 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	-	
	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 4-3: 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-write	-	
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 4-3: 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_occur	String	read-only	-	
	last_occur	String	read-only	-	
	num_of_occur	Integer	read-only	-	
	text	String	read-only	-	

Table 4-4: MTP Managed Object Descriptions

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

Table 4-4: MTP Managed Object Descriptions (Continued)

Managed Object	Parameter	Types	Access	Set Values	Operations
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
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	-	

Table 4-4: MTP Managed Object Descriptions (Continued)

Managed Object	Parameter	Types	Access	Set Values	Operations
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	-	
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	
	inst	Integer	read-create	-	
	port	Integer	read-create	-	

Table 4-4: 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	
	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 4-5: 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 4-5: 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 4-6: 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 4-6: 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	

4.3 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 4-69
- Reliability on page 4-70
- Scalability on page 4-73

- Transparency on page 4-80
- Performance Considerations on page 4-86

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

4.3.1 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.

4.3.2 Reliability

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

4.3.2.1 High Availability

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

- Deign
  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 reliability 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 reliability, discussed below, is fault tolerance.

4.3.2.2 Fault Tolerance

In general terms, fault tolerant systems are designed to mask failures so that they are hidden 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.

4.3.3 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.

4.3.4 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.

4.3.5 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 Distributed7 are:

- 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.

4.4 Core Product Specifications

4.4.1 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 10+ OS release.

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.

4.4.2 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.

4.4.2.1 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 drasticallyimproves 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.

4.4.2.2 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.

4.4.2.3 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.

4.4.2.4 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.

4.4.3 Scalability

Distributed7 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 Distributed7 environment in the “global” sense is limited to 1024. Depending on the registered layer, each application running under Distributed7 is can have a maximum of 16 or 32 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.

4.4.4 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.

4.4.4.1 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.

4.4.4.2 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.

4.4.4.3 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.

4.4.4.4 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.

4.4.4.5 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 4-85*.

4.4.4.6 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™ on *page 2-48* of the *Installation and Maintence Manual*.

4.5 ProductSpecifications

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)

4.5.1 MTP Layer Product Specifications

4.5.1.1 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.

4.5.1.2 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.

4.5.1.3 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.

4.5.1.4 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.

4.5.1.5 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.

4.5.2 SCCP Layer Product Specifications

4.5.2.1 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

4.5.2.2 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 4-83. 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 4-89.

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 4-83.

4.5.2.3 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.

4.5.2.4 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.

4.5.3 TCAP Product Specifications

4.5.3.1 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.

4.5.3.2 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 4-92.

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 4-88. 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.*

4.5.3.3 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.*

4.5.3.4 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.

4.5.4 ISUP Product Specifications

4.5.4.1 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

4.5.4.2 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.

Note: *Fault-tolerant system architectures require a minimum of two instances of the ISUP software per signaling point (in the loadshare mode) on at least two different host machines. System configurations that have 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.

4.5.4.3 Scalability

Distributed7™ supports a maximum of 2Ø48 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, Distributed7 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.

4.5.4.4 Transparency

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

Access Transparency

Distributed7 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 Distributed7 management interfaces, e.g., MML and AccessMOB.

Location Transparency

Under Distributed7, 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 Distributed7, users can start-up/stop the ISUP 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 ISUP software on a specified host or set of hosts.

Concurrence Transparency

Distributed7 allows multiple instantiations of the ISUP layer software, for a specified signaling point, to execute concurrently on multiple hosts within a distributed Distributed7 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, Distributed7 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.

This page is intentionally blank.

Chapter 5: Installation

5.1 Overview

This chapter describes installation of the Signaling Gateway Client software on a Sun platform. Signaling Gateway Client uses Distributed7 for its SS7 MTP, ISUP, SCCP and TCAP functions, and requires that Distributed7 packages be installed first.

Important: *You need a license file to run the NewNet Communication Technologies software package. This license file is already on hardware systems purchased and staged by NewNet Communication Technologies and delivered with the software already installed. You must have the following information to obtain the license file from the NewNet Communication Technologies' Technical Assistance Center (TAC):*

- *License number from the label on the CD and/or installation tape.*
- *Host ID of your machine. Enter the following in the Solaris command window to find out the host id:*

  ```
  /usr/sbin/sysdef -h
  ```
- *Call TAC at (877) 698-5583 US, (203) 647-0580 International, or email the information to support@newnet.com.*
- *The license file is then emailed to you or put up on the FTP site for you to download.*

The key file must be copied into the ***$EBSHOME/access/etc*** *directory with the file name of* ***license.dat*** *(see Step 10 on page 5-106).*

5.2 SoftwareInstallation

The Signaling Gateway Client software is installed from either a CD ROM or 1/4 inch, 4mm tape. The software contains object files, include files, sample programs, libraries, utility files, and executables.

The steps in this section have the instructions to install Signaling Gateway Client using a *packaged* format. The packages, or filesets, are dependent on other packages as shown in Table 5-1. Packages must be installed in order of their dependencies, and removed in reverse order. For instance, *D7core1* is the first package to be installed and the last one to be removed.

Multiple versions of the software can be installed on a system.

Correct operations of the software may require manipulation of one or more of the following sample configuration files on all involved hosts. These files are used to locate a variety of databases associated with host names and network services. Information about a specific entry may come from a number of sources, such as UNIX files, NIS, NIS+.

- **`/etc/nsswitch.files`**This file is a sample configuration file that instructs the network services library functions to rely on the information specified in appropriate UNIX files only.
- **`/etc/nsswitch.nis`**This file is a sample configuration file that instructs the network services library functions to rely on the information specified in appropriate UNIX files and the NIS database.
- **`/etc/nsswitch.nisplus`**This file is a sample configuration file that instructs the network services library function to rely on the information specified in appropriate UNIX files and the NIS+ database.

Important: *Users must be aware that the* ***netdbase*** *service type, which relies on the use of connection-oriented TCP/IP protocol, has been defined in the* ***/etc/services*** *file on the system and correct operations of the Signaling Gateway Client software require that the* ***netd***
daemon can invoke the ***netdir_getbyname()*** *function to retrieve information about this service type. Therefore, it is up to the user to manipulate the sample configuration files listed previously to be sure that it is possible to retrieve this information from the* ***/etc/ services*** *file. Failure to do so results in the* ***netd*** *daemon terminating its execution prematurely with the reason for termination listed in the master log file.*

Important: *By default, Distributed7 software expects the official host name of the machine to be used. Customers interested in using alias names are required to run the* ***ebs_config*** *script following initial software installation. This script prompts the customer for the host name, and the host name that the customer specifies there is saved in the* ***/etc/amgrhost*** *file. From then on, when Distributed7 system software is started, the host name listed in the* ***/ etc/amgrhost*** *file is used. This allows customers to configure the Distributed7 product to make use of alias names for their host machines, and therefore provides flexibility in product configuration, e.g., when a particular host machine is part of multiple public/ private networks.*

Example

Incorportate the following lines into the `/etc/nsswitch.conf` file that has ***files*** as the first entry for the hosts and services entries:

```
hosts: files dns nisplus [NOTFOUND=return] files

services: files nisplus [NOTFOUND=return] files
```

Important*: For more information about the format of the entries listed in the file, and how each entry is interpreted, please see the online manual pages for the* ***nsswitch.conf*** *configuration file.*

5.2.1 Disk Space Requirements

Table 5-1 lists the disk space requirements for each Signaling Gateway Client software package, and the total disk space required:

Table 5-1: Required Packages for Signaling Gateway Client

Description	Package Name	Disk Space (KB)	Prerequisite
Core	D7core1	26808	-
Application Process Management	D7core2	13408	D7core1
Distributed Memory Management	D7core3	10672	D7core1, 2
Alarm Management	D7core4	2384	D7core1, 2, 3
Graphical User Interface	D7core5	21456	D7core1-4
Measurements Collection (OMAP)	D7core6	1616	D7core1-5, D7ss7p1
MTP	D7ss7p1	28648	all D7core packages
ISUP	D7ss7p2	9936	all D7core packages, D7ss7p1
SCCP	D7ss7p3	6744	all D7core packages, D7ss7p1
TCAP	D7ss7p4	11320	all D7core packages, D7ss7p1, D7ss7p3
IS41-D	D7ss7p5	10960	all D7core packages, D7ss7p1, D7ss7p3, D7ss7p4
GSM MAP	D7ss7p6	25880	all D7core packages, D7ss7p1, D7ss7p3, D7ss7p4
GSM A	D7ss7p7	4584	all D7core packages, D7ss7p1, D7ss7p3
APIS for SC5.8	D7xlib0	29565	D7core1
APIS for SC5.0	D7xlib1	28020	D7core1
APIS for GNU3.1	D7xlib2	11991	D7core1
Signaling Gateway Client Core	SGCcore	14,ØØØ	All of the above
Total Disk Space Required		293592	

5.2.2 Installation Steps

This section has the steps to install the Signaling Gateway Client software. Installation process
has the following requirements:

- Superuser privileges are required to complete the installation.
- Multiple releases of Signaling Gateway Client can be installed on a single host. Use the **sgc_setrelease** command to switch between different releases when there is more than one release installed on a host.
- The installation procedure should be repeated on all the hosts of the SGC cluster. It is possible to install Signaling Gateway Client as standalone or distributed.
  - For multiple hosts "distributed" mode must be selected in the installation process.
  - For one host operation "standalone" must be selected.

***Note**: It is possible to change the Distributed/Standalone mode of operation after completing the installation. Installation directory names need not be same across all hosts in the cluster.*

- Go to Step 1 if you have the license file. The following information is required to obtain the license file from TAC, which must be copied to the host in Step 10:
  - The Serial number from the label on the installation CD or tape.
  - The Host ID of the machine.

***Note**: Enter* `hostid` *in the Solaris command window to get the Host ID.*

  - Give the information to the NewNet Communication Technologies Technical Assistance Center by calling (800) 416-1624 US, (408) 432-2600 International, or emailing the request to support@newnet.com. The license file is emailed to you or put up on the FTP site for you to download.

***Important**: Installation instructions are updated in the Release Notes and README file. Be sure to check these for the latest information.*

Caution *Be sure that the following Solaris 10 patch is installed on your system.*

108528-18 (cluster patch)

Enter the following command to find out the installed patches on your system:

/bin/showrev -a

Although it is better to install the patches before installing Signaling Gateway Client, it is possible to apply the patches any time AFTER installing the NewNet Communication Technologies software. The NewNet Communication Technologies software may have abnormal behavior or you may *experience fatal kernel panics/crashes if the patches are not installed!*

The patches and their installation instructions can be obtained from Sun Support (http://sunsolve.sun.com/pub-cgi/show.pl?target=home):

5.2.2.1 Installation Preparation

From a CD

1. Log in as root.
2. Create the directory where the Signaling Gateway Client software packages are to be installed. This is the *base directory* that is referenced in later installation steps.

   mkdir /newnet Ret

   - These steps use newnet as the *base directory* example, but you can create the base directory with any name.

Caution: *It is strongly recommended that the base directory be on the same host as the installation. Also there should be enough free disk space to install all the packages and to accommodate the log files which may increase substantially in size during normal operation.*

   - The startup script specifies your *base directory* entry as the $EBSHOME and $SGCHOME environment variables when logged in as **sgcadm**.

3. Put the CD into its drive.

Important*: You may install the software directly from the CD or copy the CD files into a directory (but* ***NOT*** *the tmp directory). This allows for a faster software installation from the hard disk, and the files are readily available to reinstall something without the CD. However, there must be enough disk space to copy all the files and install the software. Complete the following substeps to make a directory and copy the files from a CD. Go to the next step if you do not want to do this.*

   a. Enter the following to create a directory in the root of the hard disk (/SGC140pkg in this example):

   mkdir /SGC140pkg Ret

   cd /SGC140pkg Ret

   b. Enter the following to copy the files from the CD to the /SGC140pkg directory:

   cp -r *<cd_device_name>***/* /SGC140pkg** Ret

   where `<cd_device_name>` is the name of the CD drive being used—`/cdrom/cdrom0` for example.

4. Complete the following ***IF*** a previous version of Signaling Gateway Client is installed on the machine:

   a. Modify the **`/var/sadm/install/admin/default`** file contents so that the value of the **instance** field is set from ***overwrite*** to ***unique***.

This insures that an already installed software package, such as D7core1, is not overwritten when the new version is installed. It also allows different versions of a particular software package to co-exist in the same directory, such as D7core1, D7core1.2, and SGcore.2. This change does not prevent re-installation of a particular version of a software package in the same *base directory*.

***Note**: Use the* `pkginfo` *command to check all existing versions if there are multiple versions of a software package installed on one machine.*

5. Install the software packages either directly from the CD, or from the directory created in Step 3.a on page 5-101:

Install directly from a CD:

```
pkgadd -d devname
```

where `devname` is the device name of the CD drive being used.

OR

Install directly from source directory created in Step 3.a on page 5-101

```
pkgadd -d pathname
```

where `pathname` is the full UNIX path name of the directory to which the files were copied from the CD (/SGC140pkg in this example).

From a Tape

1. Log in as root.
2. Create the directory where the Signaling Gateway Client software packages are to be installed. This is the *base directory* that is referenced in later installation steps.

```
mkdir /newnet
```

- These steps use newnet as the *base directory* example, but you can create the base directory with any name.

Caution: *It is strongly recommended that the base directory be on the same host as the installation. Also there should be enough free disk space to install all the packages and to accommodate the log files which may increase substantially in size during normal operation.*

- The startup script specifies your *base directory* entry as the $EBSHOME and $SGCHOME environment variables when logged in as **sgcadm**.

3. Put the tape into its drive. Be sure to set a tape's write protection away from *safe*.

***Important**: You may install the software directly from the tape, or copy the files into a directory (but **NOT** the tmp directory). This allows for a faster software installation from the hard disk, and the files are readily available to reinstall something without the tape. However, there must be enough disk space to copy all the files and install the software. Complete the following substeps to make a directory and copy the files from a tape. Go to the next step if you do not want to do this.*

a. Enter the following to create a directory in the root of the hard disk (/SGC140pkg in this example):

mkdir /SGC140pkg Ret

cd /SGC140pkg Ret

b. Enter the following to copy the files from the tape to the /SGC140pkg directory:

tar xvf <*tape_device_name*> Ret

where `<tape_device_name>` is the name of the tape drive being used—*/dev/rmt/Ø* for example.

4. Complete the following ***IF*** a previous version of Signaling Gateway Client is installed on the machine:

a. Modify the **`/var/sadm/install/admin/default`** file contents so that the value of the **instance** field is set from ***overwrite*** to ***unique***.

This insures that an already installed software package, such as D7core1, is not overwritten when the new version is installed. It also allows different versions of a particular software package to co-exist in the same directory, such as D7core1, D7core1.2, and SGcore.2. This change does not prevent re-installation of a particular version of a software package in the same *base directory*.

***Note**: Use the **`pkginfo`** command to check all existing versions if there are multiple versions of a software package installed on one machine.*

5. Install the software packages either directly from the tape, or from the directory created in Step 3.a on page 5-103:

Install directly from a tape:

pkgadd -d devname Ret

where **`devname`** is the device name of the tape drive being used.

OR

Install directly from source directory created in Step 3.a on page 5-101

pkgadd -d pathname Ret

where **`pathname`** is the full UNIX path name of the directory to which the files were copied from the tape (/SGC140pkg in this example).

5.2.2.2 Installation

1. The system displays a menu of the Signaling Gateway Client packages that can be installed. Install all software packages in the order they are listed:

```
The following packages are available:
  1  D7core1     Distributed7 Core 1.4.0.6
                 (sparc) 1.4.0.6
  2  D7core2     Distributed7 Process Management 1.4.0.6
                 (sparc) 1.4.0.6
  3  D7core3     Distributed7 Memory Management 1.4.0.6
                 (sparc) 1.4.0.6
  4  D7core4     Distributed7 Alarm Management 1.4.0.6
                 (sparc) 1.4.0.6
  5  D7core5     Distributed7 GUI 1.4.0.6
                 (sparc) 1.4.0.6
  6  D7core6     Distributed7 OMAP 1.4.0.6
                 (sparc) 1.4.0.6
  7  D7ss7p1     Distributed7 MTP 1.4.0.6
                 (sparc) 1.4.0.6
  8  D7ss7p2     Distributed7 ISUP 1.4.0.6
                 (sparc) 1.4.0.6
  9  D7ss7p3     Distributed7 SCCP 1.4.0.6
                 (sparc) 1.4.0.6
 10  D7ss7p4     Distributed7 TCAP 1.4.0.6
                 (sparc) 1.4.0.6
 11  D7ss7p5     Distributed7 IS41D MAP 1.4.0.6
                 (sparc) 1.4.0.6
 12  D7ss7p6     Distributed7 GSM MAP 1.4.0.6
                 (sparc) 1.4.0.6
 13  D7ss7p7     Distributed7 GSMA 1.4.0.6
                 (sparc) 1.4.0.6
 14  SGCcore     NewNet Signaling Gateway Client Core 1.4.0.6
                 (sparc) 1.4.0.6

Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]:
```

- Press Return after this menu to install all the packages.
- Install just the SGCcore package if the same version of Distributed7 packages are already installed. Type **14** and press Return to do this.
- Use **pkginfo** command to see which versions of the packages are already installed on the system.

2. During installation, the following prompt asks you to select *distributed* or *simplex* configuration:

```
Distributed7 product can be configured to support:

+ distributed product configurations where two or more host
machines are connected to each other via a local area network.

+ simplex product configurations where the Distributed7
environment comprises isolated [stand-alone] host machines.

Which mode do you want [(1)distributed (2)simplex]?  [?]
```

If you plan to use SGC in standalone mode select *simplex*, otherwise select *distributed*.

- Enter **1** to select a ***distributed*** system.
- Enter **2** to select a ***simplex*** system.

Important: *Run ebs_config to change the Signaling Gateway Client configuration whenever you need to change the mode between* **distributed** *and* **simplex**. *Running the ebs_config command is part of the configuration in Chapter 4: Operations, Administration and Maintenance.*

3. The following prompt for the Distributed7 base directory appears to prepare for the installation of its packages:

```
Enter installation directory base:  [?,q] newnet Ret
```

This example shows ***NewNet*** as the entry for the Distributed7 base directory.

4. The following prompt for the Signaling Gateway Client Core base directory appears to

prepare for the installation of its package:

```
Enter Signaling Gateway Client Core base directory (default: /newnet)  [?,q]
```

Notice that the default base directory is what was entered in the prompt for the Distributed7 base directory. While a different directory can be entered, it is recommended that Signaling Gateway Client be installed in the same base directory as
Distributed7.

Important: *The startup script automatically sets the EBSHOME and SGCHOME environment variables to this base directory when logged in as* **sgcadm**. *It must be set manually if logged in using a different account.*

5. During installation of the individual packages, prompts appear for the `setuid/setgid` permissions for various executables. Enter `y` (YES) to all prompts. The following is an example of one of these prompts:

```
The following files are being installed with setuid and/or setgid
permissions:
  /newnet/access.1.4.0.6/bin/logd <setuid root>
  /newnet/access.1.4.0.6/bin/mml <setuid root>
  /newnet/access.1.4.0.6/bin/netd <setuid root>
  /newnet/access.1.4.0.6/bin/snmp_i <setuid root>
  /newnet/access.1.4.0.6/bin/snmp_p <setuid root>
  /newnet/access.1.4.0.6/bin/spmd <setuid root>
  /newnet/access.1.4.0.6/install/cron.restart <setuid root>
  /newnet/access.1.4.0.6/install/ebs_modunload <setuid root>

Do you want to install these as setuid/setgid files [y,n,?,q]
```

6. Enter the password for **sgcadm** when prompted to do so. Enter it a second time to confirm that it was entered correctly.

```
Please supply a password for sgcadm.
New password:
Re-enter new password:
passwd (SYSTEM): passwd successfully changed for sgcadm
```

7. The system returns to the `pkgadd` menu after all Signaling Gateway Client software packages are installed. Quit this menu.
8. Enter the following to switch to the **sgcadm** account:

   `su - sgcadm` Ret

9. Go to Step 10 if this is the first time that Signaling Gateway Client have been installed on the machine—or if there is no need to activate the release just installed. Otherwise complete the following to activate the releases:

   sgc_setrelease <*sgrlsnum*> Ret

   where *sgrlsnum* is the activated Signaling Gateway Client release number, e.g., 1.5.0.

   Run the **sgc_setrelease** command again to change the activated release(s) whenever needed.

10. Enter the following to save the license file to ***$EBSHOME/access/etc*** as ***license.dat***:

    cp <licensefile> $EBSHOME/access/etc/license.dat Ret

11. Enter the following to adjust the system parameters required to run the Signaling Gateway Client software:

    ebs_tune Ret

12. Enter the following to reboot the system:

    sync Ret

    reboot Ret

The Signaling Gateway Client software is installed and ready to configure. Please review the files listed in the following sections before continuing.

Distributed7

The following Distributed7 files are located in the ***$EBSHOME/access*** directory:

- **README** - contains an outline of the Distributed7 directories and the files in them
- **BUGS** - lists the known deficiencies in the release

Signaling Gateway Client Core

The following Signaling Gateway Client files are located in the ***$SGCHOME/sgc*** directory:

- **README** - contains an outline of the Signaling Gateway Client Core directories and the files in them
- **BUGS** - lists the known deficiencies in the release

5.2.3 Live Upgrade

This section describes user considerations and procedures when installing and/or activating a new release of the Signaling Gateway Client software on one or more hosts in a cluster while the rest of the hosts in the cluster continue to run on an earlier release. This live upgrade capability allows the user to validate the behavior of the new software before upgrading all the machines in the cluster. Upgrades include from patch level, that is 1.Ø.Ø_B to 1.Ø.Ø.1, to full version installs, such as 1.1.1. to 1.4.0.

***Important**: The Signaling Gateway Client upgrade scripts include **ALL** the necessary upgrades for Distributed7packages.*

5.2.3.1 User Considerations

The following list describes system operations and user choices at the time of installation of the new release of the Signaling Gateway Client software:

- `pkgadd` results in the full installation/activation of the Signaling Gateway Client software the
  very first time they are installed. No additional action is necessary.
- If a version of Distributed7 and/or Signaling Gateway Client is already installed on the target
  machine, `pkgadd` results in a passive installation of the new release. The release that is already installed on that machine is not removed or deactivated. In this case, users must run the `sgc_setrelease` command to activate the newly installed release. The `sgc_setrelease` command allows users to switch between different Signaling Gateway Client releases installed on the same machine.
- The following sections apply to multiple versions of either software package installed on one machine.

Distributed7 Packages

- When multiple versions of Distributed7 packages are installed on one machine, an *access* tree is created under the base product installation directory for each new version, such as `access.1.Ø.Ø_B`, `access.1.Ø.Ø.1`. To make it easy to switch between the different versions of Distributed7, the `$EBSHOME/access` entry is maintained as a symbolic link to the Distributed7 version that is currently in use.
- When multiple versions of a Distributed7 software package coexist:
  - The **pkginfo** command lists all such packages. Users should issue the **ebs_setrelease -i** command to find out which Distributed7 version is currently running.
  - It is possible to remove obsolete versions using the `pkgrm` command. Alternately, the **ebs_pkgrm** command removes all software packages associated with a particular Distributed7 version.

Caution ***SGCcore package must** be removed BEFORE **Distributed7** packages are removed!*

- When multiple versions of Signaling Gateway Client Core (SGCcore) are installed on one machine, an *sgc* tree is created under the base product installation directory for each new version, such as `sgc.1.0.0_B`, `sgc.1.0.0`. To make it easy to switch between the different versions of SGCcore, the `$SGCSHOME/sgc` entry is maintained as a symbolic link to the version that is currently in use.
- When multiple versions of a Signaling Gateway Client software package co-exist:
  - The **pkginfo** command lists all such packages. Users should run the **sgc_setrelease -i** command to find out which Distributed7 version is required by the active SGCcore software.
  - It is possible to remove obsolete versions using the `sgc_pkgrm` command. Alternately, the **sgc_pkgrm** command removes all software packages associated with a particular SGCcore version.

Caution: *SGCcore **must** be removed **BEFORE Distributed7** packages are removed!*

5.2.3.2 Live Upgrade Steps

The following sections describe the two possible scenarios for a live upgrade of the software—both Distributed7 and Signaling Gateway Client Core are upgraded at the same time, or only Signaling Gateway Client Core is upgraded.

Upgrading Distributed7 and Signaling Gateway Client at the Same Time

The following is an example of upgrading both Distributed7 and Signaling Gateway Client Core:

	Distributed7	Signaling Gateway Client Core
Existing Release	1.3.0	1.1.1
New Release	1.5.0	1.4.0

- All packages must be installed in this scenario, following the Installation Steps on page 5-100.
- Then run **sgc_setrelease** to upgrade Signaling Gateway Client Core.

Upgrading Only Signaling Gateway Client Core

The following is an example of upgrading just Signaling Gateway Client Core:

	Distributed7	Signaling Gateway Client Core
Existing Release	1.4.0	1.1.1
New Release	1.4.0	1.4.0

- Only the Signaling Gateway Client Core packages should be installed in this scenario.
- Use the `pkgadd` command to install **SGCcore**.
- Run **sgc_setrelease** to upgrade the Signaling Gateway Client files.

Caution *DO **NOT** run the **ebs_setrelease** command unless it is required to fix problems. Always use the **sgc_setrelease** command to upgrade Signaling Gateway Client.*

Related Information

- See Chapter 4: Operations, Administration and Maintenance for information about the startup and configuration of your system.
- Table 4-1 to see the directory structure that exists after successfully installing the software

Related Commands

- ebs_config on page 9-337
- ebs_setrelease on page 9-372
- sgc_setrelease on page 9-482

5.2.4 Software Removal

The software must be removed by reversing the order in which it was installed—Signaling Gateway Client Core must be removed BEFORE Distributed7 packages are removed.

Caution: Before software is removed, make sure the version to be removed is NOT running!

Complete the following steps to remove the Signaling Gateway Client Core software package:

1. Login as **sgcadm**.
2. Determine the software version to be removed. Run the following command if you are not sure of the version:

```
sgc_setrelease -i
```

3. Run the following command to remove the software:

```
sgc_pkgrm version
```

where *version* is the software version number.

Complete the following steps to remove the Distributed7 software packages:

1. Login as **sgcadm**.
2. Determine the software version to be removed. Run the following command if you are not sure of the version:

```
ebs_setrelease -i
```

3. Run the following command to remove the software:

```
ebs_pkgrm version
```

where *version* is the software version number.

The **ebs_pkgrm** command automatically removes ALL Distributed7 packages listed in Table 5-1 in proper order.

Chapter 6: Operations, Administration, and Maintenance

6.1 Overview

This chapter describes the operations, administration and maintenance of the Signaling Gateway Client system. The following areas are covered:

- operating environment of the platform
- general operations
- getting started with a new system
- configuration

6.2 Operating Environment

This section describes the software platform and environment in which the Signaling Gateway Client operates.

6.2.1 User Account

The **sgcadm** UNIX account is created for software administration purposes during installation. The **sgcadm** account has privileges to start, stop and provision the Signaling Gateway Client system.

A set of environment variables required for starting the software are set in the default `.cshrc` file of the **sgcadm** account. Always switch the user to **sgcadm**, using the following command, to be sure that you are working with the proper environment settings:

```
su - sgcadm
```

Note: This user account has root, or superuser privileges, and must be used carefully.

6.2.2 Directory Structure and Files

All Distributed7 software files are located under the ***$EBSHOME/access.X*** directory, and Signaling Gateway Client Core software files are located under the **$*SGCHOME/sgc.X*** directory, where **X** indicates the software version number.

The following table shows the directory structure of the Distributed7 and Signaling Gateway Client Core software:

Table 6-1: Directories and Files

Software Package	Software Directories	Contents
Distributed7	$EBSHOME/access/bin	Executable files
	$EBSHOME/access/demo	Demo applications
	$EBSHOME/access/drv	Device drivers
	$EBSHOME/access/etc	License and miscellaneous data files
	$EBSHOME/access/gui	Command File Navigator files
	$EBSHOME/access/help	Help text for Managed Object browser
	$EBSHOME/access/include	Header files for Distributed7 API
	$EBSHOME/access/install	Scripts related to installation
	$EBSHOME/access/lib	Libraries for Distributed7 API
	$EBSHOME/access/manpages	Man pages
	$EBSHOME/access/sample	Sample programs using Distributed7 API
	$EBSHOME/access/RUN/DBfiles	Non-SS7 Managed Object database files
	$EBSHOME/access/RUN/alarmlog	Alarm log files
	$EBSHOME/access/RUN/alog	Alternative log files
	$EBSHOME/access/RUN/mlog	Master log files
	$EBSHOME/access/RUN/config/ALARM	Alarm configuration files
	$EBSHOME/access/RUN/config/MML	MML configuration files
	$EBSHOME/access/RUN/config/PMGR	Application Process Management configuration files
	$EBSHOME/access/RUN/config/SNMP	SNMP MIBs and configuration templates
	$EBSHOME/access/RUNx/DBfiles	Managed Object databases for signaling point x
	$EBSHOME/access/RUNx/alog	Alternative log files for signaling point x
	$EBSHOME/access/RUNx/mlog	Master log files for signaling point x
	$EBSHOME/access/RUNx/config/SNMP	SNMP Agent configuration for signaling point x

Table 6-1: Directories and Files (Continued)

Software Package	Software Directories	Contents
ServiceControl-ler 1520 Core	$SGCHOME/sgc/bin	Executable files
	$SGCHOME/sgc/etc	Miscellaneous files
	$SGCHOME/sgc/install	Installation scripts
	$SGCHOME/sgc/manpages	Man pages
	$SGCHOME/sgc/RUN/config/ALARM	Alarm configuration template file
	$SGCHOME/sgc/RUN/config/MML	MML configuration template file
	$SGCHOME/sgc/RUN/config/PMGR	Application Process Management configuration template file
	$SGCHOME/sgc/RUN/config/SCTP	SCTP configuration file
	$SGCHOME/sgc/RUN/config/SNMP	SNMP configuration template files
	$SGCHOME/sgc/RUN/config/AUTOSTART	Autostart configuration file
	$SGCHOME/sgc/RUN/config/ASP	Default ASP configuration file

6.2.3 Environment Variables

The following table lists the environment variables required for Signaling Gateway Client software:

Table 6-2: Environment Variables

Environment Variable	Description
EBSHOME	Base installation directory for Distributed7
SGCHOME	Base installation directory for Signaling Gateway Client Core

6.2.4 Application Processes

The application software processes are categorized in three groups, as shown in the following table:

Table 6-3: Application Processes

Process Type	Process Name	Description
Basic Platform Processes	apmd	Application Process Manager daemon
	mlogd	Log daemon
	spmd	Service Provider Module daemon
	netd	Network daemon
	alarmd	Alarm daemon
	dsmd	Distributed Shared Memory daemon
	dkmd	Distributed Kernel Memory daemon
SS7 Node Processes	upmd	User Part Multiplexer daemon
	isupd	ISUP (ISDN User Part) daemon
	scmd	SCCP (Signaling Connection Control Part) daemon
	tcmd	TCAP (Transaction Capabilities Applications Part) daemon
Signaling Gateway Client Core Processes	aspd	Application Server Process daemon

- **Basic Platform Processes** - processes that facilitate the common platform services such as process management, logging, alarms, etc. These processes must be running at all times as long as any signaling point is active.
- **SS7 Node Processes** - processes that provide SS7 services:
  - *upmd* multiplexes user part traffic, it runs the MTP3 stack
  - *isupd* runs the ISUP stack
  - *scmd* runs the SCCP stack
  - *tcmd* runs the TCAP stack.
  - Each of these SS7 services requires that an instance of the corresponding process be running for each signaling point that is in service.
- **Signaling Gateway Client Core Processes** - processes that interwork between the IP applications and the SS7 networks.
  - The *aspd* process translates SS7 MTP messages that is packed in M3UA format into SS7 User Part messages, and vice versa. There is only one instance of ASP process per host in the Signaling Gateway Client system.

6.3 General Operations

This section describes the general operations for administering and maintaining the Signaling Gateway Client system. Only the **sgcadm** user is allowed to perform all administrative and maintenance operations.

6.3.1 Starting the Software

The user may choose to start application processes in different phases. For example, start all processes, including the basic platform, SS7 and Signaling Gateway Client Core processes, or
start only a subset of processes for maintenance. The processes must be started in the following order:

1. Basic Platform processes (*apmd, mlogd, spmd, netd, dsmd, dkmd*)
2. SS7 Node processes (*upmd, scmd, isupd, tcmd*)
3. Signaling Gateway Client Core process (*aspd*)
4. Misc processes (*omapd*)

Additionally, each process has certain dependencies on the existence of a set of processes. When starting a particular process, the processes on which that process depend are started automatically.

Process hierarchy is explained below:

- ISUP (*isupd*) and SCCP (*scmd*) processes depend on MTP3 (*upmd*) process for successful initialization. So whenever they are started *upmd* is also started if it is not already running.

Independent from the process hierarchy explained above, it is recommended to start a process before the ones which have higher group numbers (and not automatically started within thehierarchy rules explained above).

•Use the **sgc_start** command to start the application processes listed previously.

EXAMPLES

sgc_start d7 starts only the Basic Platform processes.

sgc_start sp0 starts the MTP3 process (upmd) for signaling point 0. It also starts the Basic Platform processes if they have not been started already.

sgc_start asp starts the Signaling Gateway Client Core processes, including ASP and SCTP. It also starts the Basic Platform processes if they have not been started already. However, the SS7 Node processes do not start automatically unless explicitly specified in the sgc_start command.

sgc_start sc0 sc1 tc aspstarts the Basic Platform processes if they have not been started already, then starts SS7 Node processes MTP3 and SCCP for signaling points 0 and 1, the TCAP process, the SCTP process and the ASP process.

Related Commands

• sgc_start on page 9-479

6.3.2 Stopping the Software

Use the **sgc_stop** command to stop the processes. Like the **sgc_start** command, there are arguments that stop one or more processes based on each one's place in the hierarchy. Stopping a process at a specific level in the hierarchy also stops the processes that are higher in the hierarchy.

EXAMPLES

sgc_stop asp stops the ASP process.
sgc_stop ip stops the SCTP process and the ASP process.
sgc_stop sp0 sp1stops the MTP3 Processes at signaling point 0 and 1 as well as SCCP and ISUP if running.
sgc_stop d7 stops the Basic Platform Processes and all other processes.

Related Information

- Starting the Software on page 6-117

Related Commands

- sgc_stop on page 9-481

6.3.3 Automatic Startup

The automatic startup of Signaling Gateway Client can be enabled so that the software starts automatically following a system reboot. Complete the following steps to enable automatic startup:

1. Enter the following to copy `$SGCHOME/sgc/install/initsgc` to `/etc./init.d:`

   cp $SGCHOME/sgc/install/initsgc /etc/init.d Ret

2. Enter the following to create a link for the auto start script:

   cd /etc/rc3.d Ret

   ln /etc/init.d/initsgc S99initsgc Ret

3. Create a link for the auto stop script:

   cd /etc/rc0.d Ret

   ln /etc/init.d/initsgc K00initsgc Ret

4. Make the following edits in the `$SGCHOME/sgc/RUN/config/AUTOSTART/sgcstart.conf` configuration file:

   a. Be sure that the SGC_AUTO_START value is ON.

   b. Specify the startup states that you want in the SGC_START_OPT parameter, using the sgc_start command arguments.

5. Save and Close the file.

6. The auto start and stop is now configured. Reboot the system to test it and to verify that Signaling Gateway Client starts the way you want it to after rebooting.

Note*: The S99 and K00 numbers specified in Steps 2 and 3 specify the boot sequence and may be changed to match the sequence you want.*

6.3.4 Using MML for Configuration and Query

Most software configurations are done using the Man Machine Interface (MMI) or Simple Network Management Protocol (SNMP). The Man Machine Interface (MMI) uses Managed Objects (MOs) to configure the software. A managed object defines an entity's set of parameters, and which operations can be performed on the managed object (add, delete, modify and/or display).

The ITU Z-315 Man Machine Language (MML) syntax is used with the MMI. An MML session can be started for each logical signaling point.

- Run the following command at the command line to start an MML session:

  mml sp

  where ***sp*** is the signaling point to be configured or queried.

- Type the following in the MML command prompt to display help information for a particular MML command:

  help:;

  Then, type the MML command for which help is needed in the MML help prompt, as shown in the following example:

  modify-asp:;

- Type the following in the MML command prompt to terminate an MML session:

  exit:;

Related Information

- Chapter 7: MML Commands

6.4 GettingStarted

Complete the following steps to configure the system after all the required software has been installed. These steps must be completed before the system can successfully operate and process traffic.

***Important**: This section only serves as a step-by-step guide line, the details of each step of configuration are provided in Software Configuration on page 6-123:*

1. Log in or switch the user to sgcadm:

   su - sgcadm Ret

2. Configure the TCP/IP interfaces on each host (TCP/IP Interface Configuration on page 6-124).
3. Configure the SCTP parameters on each host (SCTP Configuration on page 6-126).
4. Run **ebs_config** on each host to configure the distributed platform:
   - *a.* Type the following in the command line:

     ebs_config Ret

   - *b.* When prompted for the hostname, enter the hostname of the primary cluster LAN interface (this should have been assigned in Step 2.)
   - *c.* Enter **1** to configure a *distributed system*

     OR.

     Enter **2** to configure a *simplex system* when prompted for the operation mode
5. Enter the following on each host to start the Basic Platform Processes:

   sgc_start d7 Ret

   Only a subset of the Basic Platform Processes start if this is a distributed system and if the cluster LAN has not been configured. Execution is suspended here until the cluster LAN is configured, and a banner is displayed.
6. Run **ebs_ps** and verify that the *apmd*, *mlogd*, *spmd*, and *netd* processes are listed:

   ebs_ps Ret

7. Complete this step only if you selected 1 for a *distributed system* in Step 4 c.

   Configure the cluster LAN on each host (Section 6.5.1.4, Cluster LAN Configuration on page 6-133).

   When the cluster LAN configuration is complete, execution of the rest of the Basic Platform Processes resumes. Continue with the MTP and Signaling Gateway Client configurations using MML commands from any host that it is connected to the distributed network.

8. Configure the signaling points in MTP:

a. On each host, start the MTP3 process for the interested signaling points:

sgc_start sp0 sp1 ↵

where sp0, sp1, etc. are signaling point numbers ranging from 0-7.

b. Configure MTP from any distributed host. (Section 6.5.2.1, MTP Configuration on page 6-138)

9. Configure Signaling Gateway Client:

a. On each host, enter the following to start the ASP process:

sgc_start asp ↵

Note: This command also starts SCTP automatically.

b. Configure Signaling Gateway Client from any distributed host (Section 6.5.3, Signaling Gateway Client Configuration on page 6-138).

c. On each host, start ISUP/SCCP processes for the interested signaling points:

sgc_start is0 sc0 is1 sc1 ↵

where is0, sc0, is1, sc1, etc represent ISUP, SCCP processes for signaling points ranging from 0-7.

10. Configure SNMP on each host (Section 6.5.7, SNMP Configuration on page 6-149).

11. Configure the alarms on any distributed host (Section 6.5.6, Alarm Configuration on page 6-148).

6.5 SoftwareConfiguration

This section describes the software configuration of Signaling Gateway Client. It includes instructions and examples on configuring the LAN and SCTP interface, signaling point, ISUP, SCCP, TCAP, SIGTRAN, SNMP and alarm handling.

6.5.1 Network Configuration

The Signaling Gateway Client operates on two different networks:

- LAN for internal cluster communication - used for internal communication among the distributed hosts for data synchronization.
- LAN that connects to the WAN using SCTP over IP - used to communicate between Signaling Gateway and IP nodes.

The two networks should be isolated from each other so that the external traffic passing through the system does not interfere with the internal communication among the distributed hosts.

Figure 6-1, “LAN Connections,” on page 124 illustrates the recommended network layout for a 4-node distributed Signaling Gateway Client system. For redundancy purposes, both the cluster LAN and SCTP network have dual Ethernet interfaces. Therefore, a total of five ethernet interfaces/IP addresses are required per host, with two for each network, and one for the primary interface (comes as default with the system).

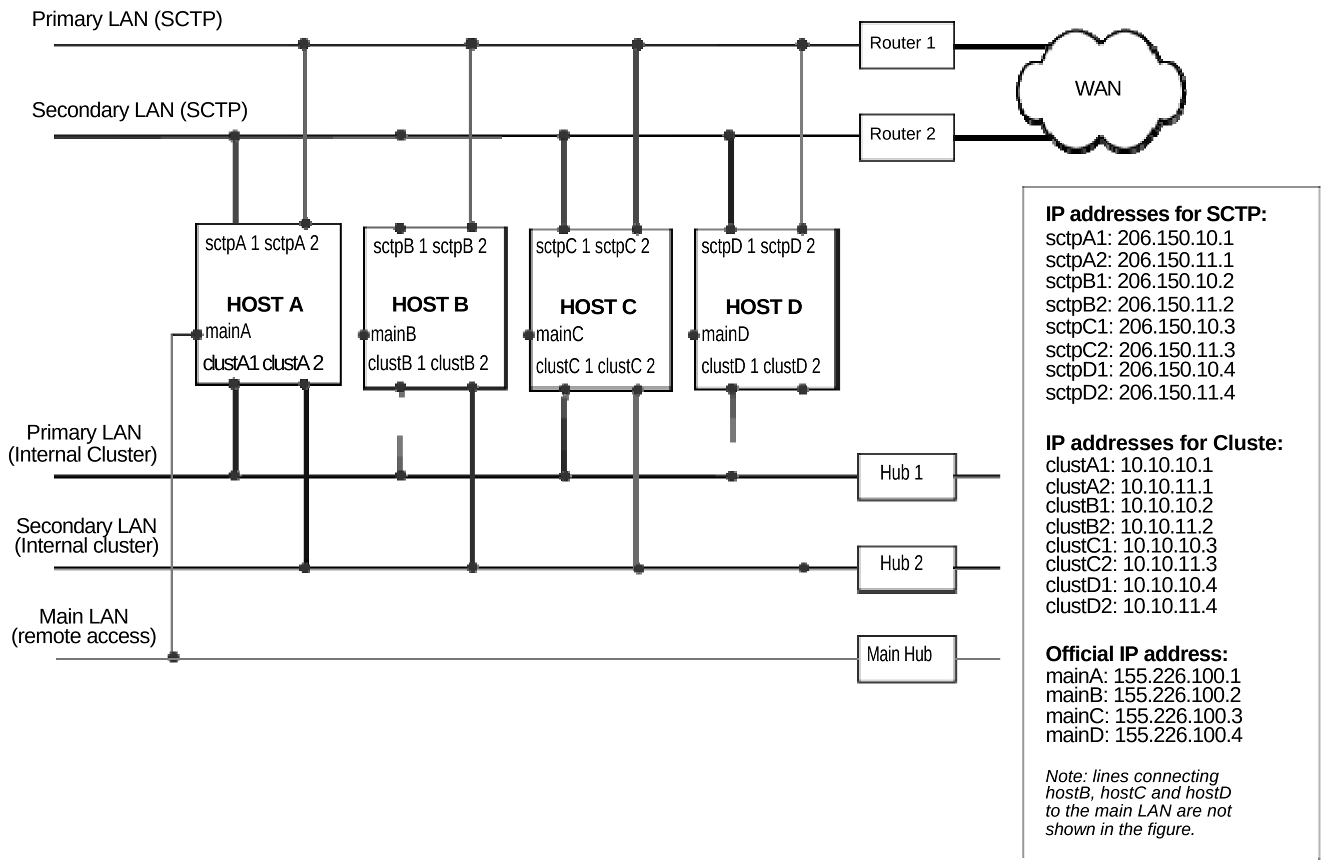


Figure 6-1: LAN Connections

6.5.1.1 TCP/IP Interface Configuration

1. Complete the following before proceeding with the configuration:

a. Design the network topology, similar to the example in Figure 6-1, “LAN Connections,” on page 124.

b. Assemble the required network hardware according to the topology designed, (installing Network Interface Cards (NIC), connecting Ethernet cables, hubs, routers, etc). Each machine must be equipped with at least five Ethernet ports:

•two for SCTP access

•two for internal cluster

•one for primary access on the main LAN

c. Assign IP addresses and host names to all network interfaces on each host involved in the network. At least five IP addresses are required for each host:

•two for SCTP

•two for the internal cluster

•one for the main LAN

2. Assign Ethernet interface names:

Each Ethernet interface on a machine is associated with an IP address or a host name. For each Ethernet interface installed on a host, create an `/etc/hostname.interface` file, where *interface* is the name associated with the Ethernet interface. Refer to the documentation that comes with the NICs for appropriate interface names.

The Solaris installation program automatically creates the `/etc/hostname.interface` file for the primary network interface that comes with the system. Additional interface files must be created for the interfaces that are installed after the initial Solaris installation.

Repeat the following steps for each of the SCTP and cluster LAN interfaces on each host:

a. With a text editor, create a file named `/etc/hostname.interface` for the network interface. Using Figure 6-2, "Sample Network Configuration," on page 137 as an example, if the network interface associated with sctpA1 is qfe0, the name of the file to create on hostA is `/etc/hostname.qfe0`.

b. Type the network interface's name or IP address in the file created in Step 2. a., then save the file. Using the previous example as before, type *sctpA1* as a one-line entry in the `/etc/hostname.qfe0` file.

3. Configure the IP addresses in the `/etc/hosts` file:

On each host, add all five IP addresses and names of each host involved in the network to the `/etc/hosts` file. Based on the example shown in Figure 6-1, "LAN Connections," on page 124, the `/etc/hosts` file may look like the following:

```
127.0.0.1       localhost


155.226.100.1   mainA        # hostA: official name
155.226.100.2   mainB        # hostB: official name
155.226.100.3   mainC        # hostC: official name
155.226.100.4   mainD        # hostD: official name


206.150.10.1    sctpA1       # hostA: SCTP primary
206.150.11.1    sctpA2       # hostA: SCTP secondary
206.150.10.2    sctpB1       # hostB: SCTP primary
206.150.11.2    sctpB2       # hostB: SCTP secondary
206.150.10.3    sctpC1       # hostC: SCTP primary
206.150.11.3    sctpC2       # hostC: SCTP secondary
206.150.10.4 sctpD1 # hostD: SCTP primary
206.150.11.4 sctpD2 # hostD: SCTP secondary


10.10.10.1      clustA1       # hostA: cluster primary
10.10.11.1      clustA2       # hostA: cluster secondary
10.10.10.2      clustB1       # hostB: cluster primary
```

```
10.10.11.2      clustB2      # hostB: cluster secondary
10.10.10.3      clustC1      # hostC: cluster primary
10.10.11.3      clustC2      # hostC: cluster secondary
10.10.10.4      clustD1      # hostD: cluster primary
10.10.11.4      clustD2      # hostD: cluster secondary
```

4. Configure `nsswitch.conf` to select the appropriate name services:

a. Open the `/etc/nsswitch.conf` file with a text editor.

b. Be sure that ***files*** is specified as the primary name service for the ***hosts*** and ***services*** network databases. ***Files*** must be listed as the first entry in the list of name services, as shown in the following example:

```
hosts:   files   nisplus [NOTFOUND=return] files
services:   files nisplus [NOTFOUND=return] files
```

These instructions cover basic network configuration. Consult the Solaris System Administration Guides for more details.

6.5.1.2 IP Security Configuration

IP security architecture (IPsec) protects the IP datagrams. It is possible to configure IPsec on the interfaces reserved for SCTP communication. This provides additional security for the SIGTRAN traffic at the IP level. Please consult the Solaris Administration Guide to learn how to configure IPsec.

6.5.1.3 SCTP Configuration

The default SCTP configuration file does not need to be modified for default operation. However, complete the following steps to open the file, customize the parameters, and implement the changes:

1. Open the SCTP configuration file (**`$SGCHOME/sgc/RUN/config/SCTP/sctpd.conf`**) with a text editor.

2. The ***sctpd.conf*** has comments explaining each configuration parameter in the file. The parameter values can be edited to meet the user needs.

3. Save the file, and quit the text editor.

4. On Solaris 10, if the aspd process is running, get the process id and send HUP signal to the process to implement the changes in the revised sctpd.conf file. The following is an example of doing this:

ps -eaf | grep aspd

(get the process id)

kill -HUP <pid>

Repeat this step on each of the distributed hosts that is running Signaling Gateway Client.

The
following is a printout of the default ***sctpd.conf*** file.

```
###############################################################################
# SCTP Configuration File
###############################################################################
#
# Parameters can be changed in run-time by sending a HUP signal to the
#   sctp daemon process on Solaris 8/9, or the asp/sgp daemon process on Solaris 10.
#
#   example:
#
#     On Solaris 8/9
#       prompt> ps -eaf | grep sctpd
#             (note the pid of sctpd in the output screen)
#       prompt> kill -HUP <pid of sctpd>
#
#     On Solaris 10
#       prompt> ps -eaf | grep aspd
#             (note the pid of aspd in the output screen)
#       prompt> kill -HUP <pid of aspd>
#
# To change a configuration item uncomment the example line and modify the
#   configuration value according to the acceptable values.
#
# Parameters are divided into two main parts. Changes in PART I parameters
#   are applicable only to the associations opened after the update.
#   Changes in PART II parameters are effective to all associations
#   immediately.
#
# Refer to RFC 2960 for further info on the following parameters.
#
###############################################################################
# PART I
###############################################################################
#
# max_init_timeo <msec>
#   <msec> : INIT Message Time-Out (RTO) in milliseconds. (default = 3000)
#
#   This value is used as the initial value of the Time-Out when sending INIT Message to a destination
# for setting up the SCTP association.
#
# example:
#   max_init_timeo 3000
#
#
# rto_init <msec>
#   <msec> : Initial Retransmission Time-Out (RTO) in milliseconds. (default = 1000)
#
#   This value is used as the initial value of the RTO before sending traffic to a destination after association
# setup or when a destination path becomes active.
#
# example:
#   rto_init 1000
#
```

```
#
# rto_min <msec>
#   <msec> : Minimum Retransmission Time-Out (RTO) in milliseconds. (default = 1000)
#
#   Defines the minimum allowed RTO value when SCTP stack adjusts RTO.
#
# example:
#   rto_min 1000
#
#
# rto_max <msec>
#   <msec> : Maximum Retransmission Time-Out (RTO) in milliseconds. (default = 1000)
#
#   Defines the maximum allowed RTO value when SCTP stack adjusts RTO.
#
# example:
#   rto_max 1000
#
#
# valid_cookie_life <msec>
#   <msec> : Valid cookie life in milliseconds. (default = 60000)
#
#   Defines the lifetime of cookie after which the cookie is considered invalid (i.e. stale).
#
# example:
#   valid_cookie_life 60000
#
#
# assoc_max_retrans <attempts>
#   <attempts> : Association max retransmissions. (default = 5)
#
#   The association is closed automatically when number of retransmissions exceeds assoc_max_retrans.
#
# example:
#   assoc_max_retrans 5
#
#
# path_max_retrans <attempts>
#   <attempts> : Path max retransmissions. (default = 2)
#
#   On Solaris 10, the parameter can't be changed in run-time, and it will take effect after aspd/sgpd restarts.
#
#   The path becomes INACTIVE after number of retransmissions to a path (i.e. destination) exceeds
# path_max_retrans.
#
# example:
#   path_max_retrans 2
#
```

```
# max_init_retrans <attempts>
#   <attempts> : Maximum init retransmissions. (default = 5)
#
#   Association setup fails when number of (INIT or COOKIE-ECHO) retransmissions exceeds
init_max_retrans.
#
# example:
#   max_init_retrans 5
#
#
# hb_interval <msecs>
#   <msecs> : Heartbeat interval in milliseconds. (default = 1000)
#
#   On Solaris 10, the parameter can't be changed in run-time, and it will take effect after aspd/sgpd restarts.
#
#   When a path is idle for hb_interval milliseconds a HEARTBEAT message is sent to the destination.
#
#
# example:
#   hb_interval 1000
#
#
# num_in_streams <value>
#   <value> : Default number of incoming streams per assoc. (default = 32)
#
# example:
#   num_in_streams 32
#
#
# num_out_streams <value>
#   <value> : Default number of outgoing streams per assoc. (default = 32)
#
# example:
#   num_out_streams 32
#
#
# arwnd_size <value>
#   <value> : Advertised receive window size in bytes. (default = 264000)
#
#   The parameter is valid for sctpd on Solaris 8/9.
#
# example:
#   arwnd_size 264000
#
#
# t_sack <msecs>
#   <msecs> : SACK delay timeout in milliseconds. (default = 200)
#
#   The parameter is valid for sctpd on Solaris 8/9.
#
#   Delayed acknowledgement timer. If set to 0 delayed acknowledgement is disabled. The timer value
# must not be greater than 500 msecs.
#
# example:
#   t_sack 200
#
```

```
#
# t_shutdown_guard <msecs>
#   <msecs> : Shutdown guard timer in milliseconds. (default = 5000)
#
#   The parameter is valid for sctpd on Solaris 8/9.
#
#   Graceful shutdown procedure is aborted when guard timer expires.
#
# example:
#   t_shutdown_guard 5000
#
#
# t_path_mtu <msecs>
#   <msecs> : Path mtu raise timeout in milliseconds. (default = 600000)
#
#   The parameter is valid for sctpd on Solaris 8/9.
#
#   In accordance with path discovery procedures the path maximum transmission unit (MTU) is increased
# when this timer expires.
#
# example:
#   t_path_mtu 600000
#
#
# report_unsent <value>
#   <value> : Enable/disable sending of unsent messages in the transmit buffer to the ULP when an
# association is closed. (default = 1)
#           0:disable 1:enable
#
#   The parameter is valid for sctpd on Solaris 8/9.
#
# example:
#   report_unsent 1
#
#
# init_tsn <value>
#   <value> : Initial TSN value for the associations. (default = -1)
#           -1:generated by SCTP randomly
#
#   The parameter is valid for sctpd on Solaris 8/9.
#
# example:
#   init_tsn -1
#
#
# tx_queue_length <value>
#   <value> : transmit queue length (default = 3000)
#
#   The parameter is valid for sctpd on Solaris 8/9.
#
#   Maximum number of data chunks that can be queued up before transmission.
#   overflowed chunks are discarded.
#
# example:
#   tx_queue_length 3000
#
```

```
#
# ecn <value>
#   <value> : enabled(1)/disabled(0) (default = 0)
#
#   The parameter is valid for sctpd on Solaris 8/9.
#
#   Explicit Congestion Notification support. (rfc3168, rfc2960)
#
# example:
#   ecn 0
#
###############################################################################
# PART II
###############################################################################
#
# chksum <value>
#   <value> : checksum algorithm (default = 1)
#             0:Adler32 1:CRC32
#
#   The parameter is valid for sctpd on Solaris 8/9.
#
# example:
#   chksum 1
#
#
# ip_tos <value>
#   <value> : IP Type Of Service value (default = 0)
#
#   ip_tos value is copied to the tos field of outgoing IP packets.
# Meaningful values:
#
#   IPTOS_PREC_NETCONTROL       224
#   IPTOS_PREC_INTERNETCONTROL  192
#   IPTOS_PREC_CRITIC_ECP       160
#   IPTOS_PREC_FLASHOVERRIDE    128
#   IPTOS_PREC_FLASH             96
#   IPTOS_PREC_IMMEDIATE         64
#   IPTOS_PREC_PRIORITY          32
#   IPTOS_LOWDELAY               16
#   IPTOS_THROUGHPUT              8
#   IPTOS_RELIABILITY             4
#   IPTOS_PREC_ROUTINE            0
#
# example:
#   ip_tos 16
#
#
# ip_ttl <value>
#   <value> : IP Time To Live value (default = 64)
#
#   ip_ttl value is copied to the ttl field of outgoing IP packets.
#
# example:
#   ip_ttl 255
#
```

6.5.1.4 Cluster LAN Configuration

The cluster LAN must be configured to connect all the distributed hosts into one network. This must be done on each new system immediately after starting the basic platform processes. For increased reliability, it is highly recommended to deploy the dual-LAN model as illustrated in Figure 6-1: LAN Connections. Complete the following steps on each host to configure the cluster LAN (based on the example in Section 6.5.1.4 on page 6-133):

The following subsections provide sample configurations for 2-node and 4-node clusters.

2-Node Cluster

The following configuration steps are based on a 2-host example, which is a subset taken from Figure 6-1, "LAN Connections," on page 124. They also assume that the network has hostA and hostB connected on a dual LAN.

1. Configure hostA:
   - *a.* Log in to hostA and start MML: **mml** 0
   - *b.* Define the primary and secondary cluster host names for hostA:

     modify-ntwk:hostname=clustA1,dualhost=clustA2;
   - *c.* Define the netmasks used for the cluster LAN on hostA:

     modify-ntwk:hostname=clustA1,dualhost=clustA2,netmask1=x, netmask2=y;
   - *d.* Define the remote primary and secondary cluster host names that are reachable from hostA:

     add-host:hostname=clustA1,rmthost=clustB1,alias=clustB2,conf=ON;
2. Configure hostB:
   - *a.* Log in to hostB and start MML.
   - *b.* Define the primary and secondary cluster host names for hostB:

     modify-ntwk:hostname=clustB1,dualhost=clustB2;
   - *c.* Define the netmasks used for the cluster LAN on hostB:

     modify-ntwk:hostname=clustB1,dualhost=clustB2,netmask1=x, netmask2=y;
   - *d.* Define the remote primary and secondary cluster host names that are reachable from hostB:

     add-host:hostname=clustA1,rmthost=clustB1,alias=clustB2,conf=ON;
3. After the hosts are configured, run the **ebs_showlink** command on each host to verify that it is connected to each host in the distributed network.

***Note**: For single LAN configuration, follow the same steps as above, except that you do not need to specify the secondary host name. For example:*

modify-ntwk:hostname=clustB1;

modify-ntwk:hostname=clustB1,netmask1=x;

add-host:hostname=clustA1,rmthost=clustB1,conf=ON;

4-Node Cluster

The following configuration steps are based on the details in Figure 6-1, “LAN Connections,” on page 124.

1. Configure hostA:
   - *a.* Log in to hostA and start MML: **mml 0**
   - *b.* Define the primary and secondary cluster host names for hostA:

     modify-ntwk:hostname=clustA1,dualhost=clustA2;
   - *c.* Define the netmasks used for the cluster LAN on hostA:

     modify-ntwk:hostname=clustA1,dualhost=clustA2,netmask1=x, netmask2=y;
   - *d.* Define all remote primary and secondary cluster host names that are reachable from hostA:

     add-host:hostname=clustA1,rmthost=clustB1,alias=clustB2,conf=ON;

     add-host:hostname=clustA1,rmthost=clustC1,alias=clustC2,conf=ON;

     add-host:hostname=clustA1,rmthost=clustD1,alias=clustD2,conf=ON;
2. Configure hostB:
   - *a.* Log in to hostB and start MML.
   - *b.* Define the primary and secondary cluster host names for hostB:

     modify-ntwk:hostname=clustB1,dualhost=clustB2;
   - *c.* Define the netmasks used for the cluster LAN on hostB:

     modify-ntwk:hostname=clustB1,dualhost=clustB2,netmask1=x, netmask2=y;
   - *d.* Define the remote primary and secondary cluster host names that are reachable from hostB:

     add-host:hostname=clustB1,rmthost=clustA1,alias=clustA2,conf=ON;

     add-host:hostname=clustB1,rmthost=clustC1,alias=clustC2,conf=ON;

     add-host:hostname=clustB1,rmthost=clustD1,alias=clustD2,conf=ON;
3. Configure hostC in similar fashion:

   modify-ntwk:hostname=clustC1,dualhost=clustC2;

   modify-ntwk:hostname=clustC1,dualhost=clustC2,netmask1=x, netmask2=y;

   add-host:hostname=clustC1,rmthost=clustA1,alias=clustA2,conf=ON;

   add-host:hostname=clustC1,rmthost=clustB1,alias=clustB2,conf=ON;

   add-host:hostname=clustC1,rmthost=clustD1,alias=clustD2,conf=ON;
4. Configure hostD in similar fashion:

```
modify-ntwk:hostname=clustD1,dualhost=clustD2;
modify-ntwk:hostname=clustD1,dualhost=clustD2,netmask1=x,
    netmask2=y;
add-host:hostname=clustD1,rmthost=clustA1,alias=clustA2,conf=ON;
add-host:hostname=clustD1,rmthost=clustB1,alias=clustB2,conf=ON;
add-host:hostname=clustD1,rmthost=clustC1,alias=clustC2,conf=ON;
```

5. After the hosts are configured, run the **ebs_showlink** command on each host to verify that it is connected to each host in the distributed network.

Note: *For single LAN configuration, follow the same steps as above, except that you do not need to specify the secondary host name. See Note after 2-Node Cluster on page 6-133.*

Related Information

- ebs_showlink on page 9-373

Related MML

- Host (HOST) on page 7-237
- Network (NTWK) on page 7-239

Changing the LAN Configuration

Stop the entire platform and complete the following steps if you need to switch from a single-LAN to dual-LAN configuration, or vice versa:

1. Enter the following to stop the entire platform:

```
sgc_stop d7
```

2. Enter the following to remove all local NTWK, HOST and TCPCO Managed Object databases:

```
ebs_config -u
```

3. Enter the following to restart Signaling Gateway Client software:

```
sgc_start d7
```

4. Enter the following to start MML:

```
mml 0
```

5. Repeat the steps at the beginning of Section 6.5.1.4 on page 6-133 to configure a single or dual-LAN.

Related MML

- ebs_config on page 9-337
- sgc_start on page 9-479

6.5.2 Signaling Network Configuration

This section describes configuring Signaling Gateway Client to provide both SIGTRAN M3UA and SS7 User Part services.

All explanations in the following subsections are based on the sample network in Figure 6-2 on page 137. In this example:

- Signaling Gateway Client operates on a 2-host distributed platform where one ASP runs on each host. Signaling Gateway Client has SP0, SP1 and SP2 in service, and all SPs are operating in STP mode.
- SP0
  - SP0's PC is 1-1-20, and has an SS7 destination with PC 1-1-10
  - SP0 is assigned NA 100 and serves an AS with RC 1
- SP1
  - SP1's PC is 2-2-20, and has an SS7 destination with PC 2-2-10
  - SP1 is assigned NA 200 and serves an AS named AS2
- SP2
  - SP2's PC is 2-2-21, and has an SS7 destination with PC 2-2-20
  - SP2 is also assigned 200 and serves an AS named AS3

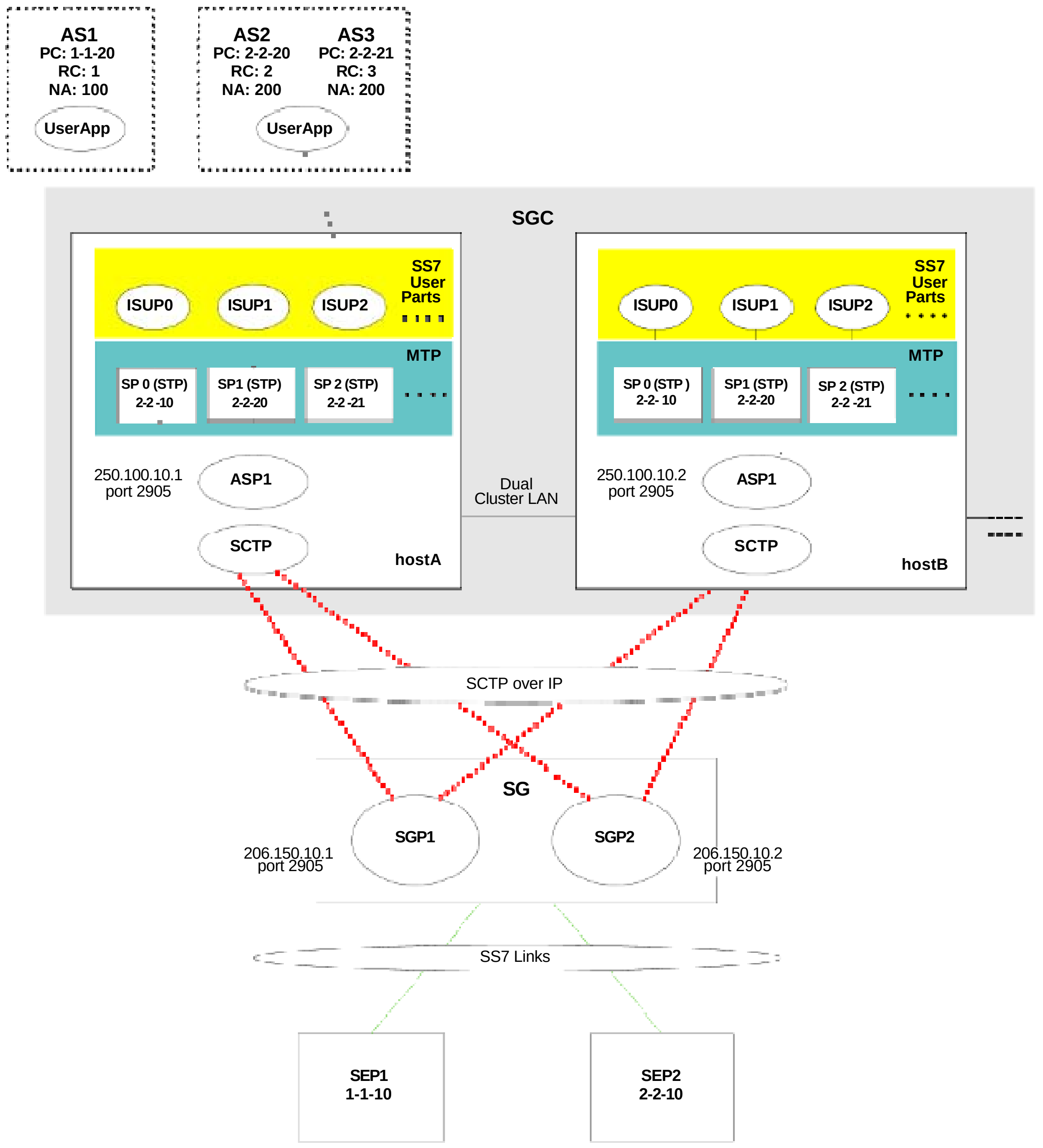


Figure 6-2: Sample Network Configuration

6.5.2.1 MTP Configuration

The Signaling Gateway Client system must be configured as an SS7 network node in order to
process SS7 signaling messages. This section provides a sample configuration steps for configuring SP0 of the SS7 MTP layer, as illustrated in the Figure 6-2: Sample Network Configuration.

1. Signaling Node

   Enter the following to configure Signaling Gateway Client as an SS7 node by defining
   the protocol variant and point code size used in MTP Ret to define the protocol information used by SP0:

   add-mtp:protocol=ANSI_96,pcsize=24_BIT;

   After the MTP instance is added, an instance of the SP is created automatically, and is modified in the next step.

2. Point Code

   Each addressable SS7 signaling point must be assigned a Ret unique SS7 point code. Assign a point code to SP0 and specify the network indicator and node type:

   modify-sp:spc=1-1-20,ni=NATIONAL,type=STP;

6.5.3 Signaling Gateway Client Configuration

The following serves as a step-by-step guideline for configuring the Signaling Gateway Client.
It is recommended that these guidelines be followed in the order provided:

1. Configure signaling point and network apearance mapping
2. Configure local ASP
3. Configure remote SG and SGP
4. Configure remote SS7 DPC
5. Configure route key and AS
6. Activate SCTP association and AS traffic control

6.5.3.1 Signaling Point and Network Appearance Mapping

Signaling Gateway Client can support up to eight signaling points (SP), each connecting to a different SS7 network. Every SP can serve one AS in a particular network appearance, and multiple SPs can serve the same network appearance.

The SGCSPNA managed object defines the Ret mapping between an SP and a network appearance ID. It must be created for each SP that will be in service, and must be defined before any route key and AS are added. The Ret following examples define SP0 and SP1, both serving the same network appearance:

add-sgcspna:spno=0,naid=100;

add-sgcspna:spno=1,naid=100;

The SP must be activated so that it can be in service after the mapping between NAID and SP is defined:

mod-sgcspna:spno=0,operst=ACT;

mod-sgcspna:spno=1,operst=ACT;

Related MML

- Signaling Point to Network Appearance Mapping (SGCSPNA) on page 7-258

6.5.3.2 Local ASP

The SGCASP managed object defines the local ASP process IP addresses and SCTP port number used for SCTP communication. Two IP addresses can be specified for SCTP multihoming purposes.

By default, the SGCASP managed object is created automatically for each ASP process, but has no IP addresses or port definitions initially.

Modify the SGCASP MO for each host to set the IP addresses and port number used by the ASP for SCTP communication:

```
modify-sgcasp:aspid=asp1,ip1=250.100.10.1,sctpport=2905,
   sgredmode=OVERRIDE,nwaspid=1000;
modify-sgcasp:aspid=asp2,ip1=250.100.10.2,sctpport=2905,
   sgredmode=OVERRIDE,nwaspid=1001;
```

***Important**: The ASP ID is pre-assigned the machine's official host name when the SGCASP is first created. Use the UNIX **uname -n** command to display a machine's official host name.*

Related MML

- Application Server Process (SGCASP) on page 7-243

6.5.3.3 Remote SG

The SGCSG managed object defines a remote Signaling Gateway that is connected to the Signaling Gateway Client. It also defines the traffic mode of the Signaling Gateway, that is, whether the SGPs are operating in loadsharing or override mode.

The following example defines two remote SGs that are connected to the Signaling Gateway Client:

```
add-sgcsg:sgid=SG1,mode=LOADSHARE;
add-sgcsg:sgid=SG2,mode=LOADSHARE;
```

Related MML

- Signaling Gateway (SGCSG) on page 7-253

6.5.3.4 Remote SGP

The SGCSGP managed object defines the SGP process's IP addresses and SCTP port number used for SCTP communication. An SGCSGP managed object must be added for each SGP instance that is to connect to the SGP.

Two IP addresses can be specified for SCTP multi-homing purposes. Each SGP must be assigned a unique SCTP port number for communication with SCTP if multiple SGPs are running on the same host. Be sure that the IP addresses and port numbers configured on the Signaling Gateway side are identical to those on the ASP side to ensure successful connections.

The following commands configure the remote SGPs shown in Figure 6-2, "Sample Network Configuration," on page 137:

```
add-sgcsgp:sgpid=SGP1,sgid=SG1,ip1=206.150.10.1,ip2=206.150.11.1,
    sctpport=2905;
add-sgcasp:sgpid=SGP2,sgid=SG1,ip1=206.150.10.2,ip2=206.150.11.2,
    sctpport=2905;
```

Related MML

- Signaling Gateway Process (SGCSGP) on page 7-255

6.5.3.5 SS7 Destination Point Code

The SGCDPC managed object defines the remote SS7 destination that is reachable by Signaling Gateway Client through a particular Signaling Gateway in a particular network appearance. It is possible for Signaling Gateway Client to reach a remote SS7 destination via more than one Signaling Gateway. The following examples define the reachability of an SS7 destination through two different SGs:

```
add-sgcdpc:pc=1-1-10,sgid=SG1,naid=100;
add-sgcdpc:pc=1-1-10,sgid=SG2,naid=100,priority=0;
```

Related MML

- Destination Point Code (SGCDPC) on page 7-247

6.5.3.6 Configuring a Route Key

The SGCRK managed object defines the route key associated with an AS. A route key includes a set of SS7 parameters that uniquely defines the range of signaling traffic to be handled by a particular AS. There is a one-to-one mapping between a route key and an AS. An SGCRK managed object must be added before it is associated with an AS.

The following commands add a route key for each AS shown in Figure 6-2, "Sample Network Configuration," on page 137:

```
add-sgcrk:rkid=r1,type=DPC,dpc=1-1-20;
add-sgcrk:rkid=r2,type=DPC,dpc=2-2-20;
add-sgcrk:rkid=r3,type=DPC,dpc=2-2-21
```

The following lists the different types of route keys:

DPC route by DPC only

DPC_OPC route by DPC and a ranges of OPCs

DPC_OPC_CIC route by DPC, and ranges of OPC-CIC pairs

DPC-SIO route by DPC and SIO

DPC_CIC_SIO route by DPC, SIO and CIC ranges

DPC_OPC_SIO route by DPC, SIO and OPC ranges

If traffic is to be routed by DPC only, there is no need to define an SGCRKRNG MO. However, if it is to be routed based on a range of parameters, the SGCRKRNG MO must be configured to specify the traffic range parameters, as shown in the following examples:

```
add-sgcrk:rkid=rk1,type=DPC_OPC_CIC,dpc=1-1-20;
add-sgcrkrng:rkid=rk1,opc=1-1-10,cicmin=1000,cicmax=2000;
add-sgcrkrng:rkid=rk1,opc=1-1-10,cicmin=2500,cicmax=3000;
```

OR

```
add-sgcrk:rkid=rk1,type=DPC_CIC_SIO;
add-sgcrkrng:rkid=rk1,si=ISUP,cicmin=5000,cicmax=6000;
add-sgcrkrng:rkid=rk1,si=ISUP,cicmin=7000,cicmax=7500;
```

Related MML

• Routing Key (SGCRK) on page 7-249

6.5.3.7 Configuring an AS

The SGCAS managed object defines the parameters of an AS, a logical entity serving a specific route key. Each SGCAS managed object is associated with an instance of SGCRK managed object which must be created in advance. In addition, each AS is also associated with a unique numeric value called *route context* (RC). There is a one-to-one mapping between an AS and a RC ID. The RC ID provisioned on the Signaling Gateway side must be identical to that of the ASP side to ensure successful message routing.

Each SP on Signaling Gateway Client can serve only one AS, or route key. When adding the SGCAS MO, the unique SPID is specified to associate the AS with the SP.

Using Figure 6-2, “Sample Network Configuration,” on page 137 as an example, SP0 serves AS1 in the network with NA 100; SP1 and SP2 serve AS2 and AS3 respectively, and both SPs are in the same network with NA 200. Thus, the ASs can be configured as follows:

```
add-sgcas:asid=AS1,spid=0,rkid=r1,rcid=1,mode=LOADSHARE;
add-sgcas:asid=AS2,spid=1,rkid=r2,rcid=2,mode=LOADSHARE;
add-sgcas:asid=AS3,spid=2,rkid=r3,rcid=3,mode=LOADSHARE;
```

Related MML

• Application Server (SGCAS) on page 7-241

6.5.3.8 Defining AS Traffic Control

An ASP may serve multiple ASs, and it has different traffic status for each of the ASs it serves. Moreover, an ASP's statuses may be different for different remote SGPs. For example, an ASP may be activated to process traffic for AS1 over an association between this ASP and a particular SGP, and it can be disabled so as not to process traffic for AS2 over the same association. The relation between the ASP, SGP, AS and the traffic statuses are represented in the SGCASTFC MO.

The following commands activate ASP1 to process traffic for AS1 over the associations with SGP1 and SGP2:

```
add-sgcastfc:aspid=ASP1,sgpid=SGP1,asid=AS1,operst=ACT;
add-sgcastfc:aspid=ASP1,sgpid=SGP2,asid=AS1,operst=ACT;
```

The following command deactivates traffic for AS2 over the association between ASP1 and SGP2:

```
mod-sgcastfc:aspid=ASP1,sgpid=SGP2,asid=AS1,operst=INACT;
```

Related MML

- AS-ASP Traffic Control (SGCASTFC) on page 7-245

6.5.3.9 IP Server Process

The SGCIPSP managed object defines the remote IPSPs IP addresses and SCTP port numbers used for SCTP communication. An SGCIPSP managed object must be added on the Signaling Gateway Client system for each IPSP instance that is to connect to the ASP. Two IP addresses can be specified for SCTP multi-homing purposes. Each IPSP must be assigned a unique SCTP port number for communication with SCTP if multiple IPSPs are running on the same host. Be sure that the IP addresses and port numbers configured on the Signaling Gateway Client side are identical to that of the IPSP side to ensure successful connection.

The following commands configure the remote ASPs as shown in Figure 4-2: on page 4 - 53:

```
add-sgcipsp: ipspid=jilin,nwaspid=75,ip1=10.20.2.75, sctpport=2905,mode=CLIENT;
add-sgcipsp:ipspid=shenyang,nwaspid=72,ip1=10.20.2.72,
sctpport=2905,mode=SERVER;
```

Related MML

- Remote IP Server Process on page 7-261.

6.5.3.10 IP Application Server

The SGCIPAS MO defines the attributes of a remote AS residing on other SGCs. Each SGCIPAS instance is associated with two instances of SGCRK, which must be created in advance. One of the SGCRK instances (OWN_RKID) defines the traffic originating from the local side towards the IPAS whereas the other (RMT_RKID) defines the traffic coming from the IPAS. In addition, each IPAS is also mapping between an IPAS and an RCID (Routing Context ID).

***Note**: Routing Key (RKID) defines the semantics of traffic flow (based on point code or protocol information). When an AS (SG/SGC) is defined, traffic flowing to/from the AS is defined per Routing Key. But when sending messages towards that AS, a unique protocol identifier needs to be provisioned the same on both the SG and the SGC. This identifier is called the Routing Context ID (RCID).*

The RCID provisioned on the Signaling Gateway Client side must be identical to that of the IPSP side to ensure successful message routing. An IPAS can be assigned a list of up to eight IPSP's, operating in either override or load sharing mode. These IPSPs must be created before they can be in the list of assigned IPSPs (IP Server Process on page 6-142). Signaling Gateway Client supports a total of 2048 IPASs, and each SP can be configured to serve more than one IPAS. When adding the SGCIPAS MO, the SPID is specified to associate the IPAS with the SP.

The remote ASs are configured as follows:

add-sgcipas: ipasid=IPAS, spid=6, OWN_RKID=rk_local, RMT_RKID=rk_remote,rcid=7572,ipsplist=jilin;

add-sgcipas: ipasid=IPAS, spid=7, OWN_RKID=rk_local, RMT_RKID=rk_remote, rcid=7572,ipsplist=shenyang;

Related MML

- IP Application Server (SGCIPAS) on page 7-263.

6.5.4 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 Signaling Gateway Client 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.

6.5.4.1 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:

add-snsp:spc=1-3-2;

Related MML

- SCCP Signaling Point (SNSP) on page 7-214

6.5.4.2 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 Figure 6-2, SP2 has two subsystems that will be accessed. These subsystems are added to the database with the following commands:

add-subsys:spc=1-3-2,ssn=253;

add-subsys:spc=1-3-2,ssn=254;

Related MML

- Subsystem (SUBSYS) on page 7-216

6.5.4.3 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 6-2, 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:

add-cpc:spc=1-2-3,ssn=20,cpc=1-3-5;

add-cpc:spc=1-2-3,ssn=21,cpc=1-3-5;

Related MML

- Concerned Point Code (CPC) on page 7-204

6.5.4.4 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 wild card 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*

Related MML

- Global Title (GT) on page 7-206

6.5.4.5 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;

Related MML

- Global Title Entry (GTENTRY) on page 7-208

6.5.4.6 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:

add-mate:spc=1-2-3,ssn=21,mspc=1-3-2,mssn=253;

Related MML

- Mate (MATE) on page 7-210

6.5.5 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.

6.5.5.1 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 6-2, 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.

Related MML

- ISUP Signaling Node (ISUPNODE) on page 7-227

6.5.5.2 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:

***T1:* add-isupcgrp:pcno=1,grpid=2,cctnum=24,trnkgrpid=3, scga=on ;**

***E1:* add-isupcgrp:pcno=1,grpid=4,cctnum=32,trnkgrpid=3;**

Related MML

- ISUP Circuit Group (ISUPCGRP) on page 7-224

6.5.5.3 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.

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.

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 (signaling 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.

The following commands add single circuits:

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 signaling link is in the group.

E1: **add-isupcct:pcno=1,grpid=4,cctnum=1,range=30;**

For an E1, if a signaling 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;

***Note**: 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.*

Related MML

• ISUP Circuits (ISUPCCT) on page 7-220

6.5.6 Alarm Configuration

6.5.6.1 Alarm Type and Severity

Alarms are generated to alert the operators of any system faults or major events that require attention. Alarms are categorized in the following groups:

- EVENT - informational alarms that are generated usually for event logging purposes
- SET_ALARM - alarms that are set in the system and remain until it is manually cleared by an operator
- CLR_ALARM - alarms that are generated to automatically clear an alarm of SET_ALARM type.

All alarms are logged and kept in the directory **`$EBSHOME/access/RUN/alarmlog`**. Alarms of SET_ALARM type can be queried through MML. Some alarms of SET_ALARM type are self-clearing—they are cleared automatically when the error condition is corrected. Non-self-clearing alarms must be cleared manually by the operator.

Alarms are rated according to the following four severity levels:

- INFO - informational alarm, reporting events such as system startup, shutdown, etc. These alarms do not affect the functional state of the system.
- MINOR - alarms that indicate minor error conditions that do not impair the continued operation of the system, such as configuration errors, invalid messages received, etc. These alarms should not be ignored as they may indicate an impending problem.
- MAJOR - alarms that indicate major error conditions that affect the normal operation of the system, such as traffic congestion, resource utilization reaching maximum threshold, etc. These alarms require immediate attention to prevent potential service interruption.
- CRITICAL - alarms that indicate critical error conditions that is interrupting the normal operation of the system, such as process failure, loss of network connection, hardware faults, etc.

6.5.6.2 Alarm Groups

The system alarms are categorized into various groups. Each group is responsible for different types of functions. The alarm groups are:

- TRMOD (translation module)
- SPM (signaling point management)
- UPM (user part multiplexer)
- NIMOD (connection management)
- DKM (distributed kernel memory)
- SG (signaling gateway application)

6.5.6.3 Alarm Display

By default, all alarms are displayed on the console and written to the alarm log file. Run the following command to toggle the display option:

modify-alarm:display=option; Ret

where *option* is either ON or OFF.

The alarms that are displayed can be filtered according to their severity levels. For example, to display alarms that have severity level that are equal to or higher than MINOR, Run the following command MML command:

modify-alarm:cons_thrs=MINOR; Ret

6.5.7 SNMP Configuration

Signaling Gateway Client supports Simple Network Management Protocol (SNMP) versions 1 and 2 for network management. In a network management system, there can be multiple network elements such as hosts, routers, terminal servers, etc., each with a processing agent called an *SNMP agent*. A centralized location called the *network management station* or *SNMP manager* is used to manage these individual network elements. The manager remotely controls or monitors the network elements by accessing their *management information*.

Management information is simply a collection of managed objects residing in a virtual information store, called a Management Information Base (MIB). MIBs are defined by the Structure of Management Information (SMI), which is a subset of OSI's Abstract Syntax Notation One (ASN.1).

Figure 6-3 illustrates the SNMP architecture involving an SNMP Agent and an SNMP Manager.

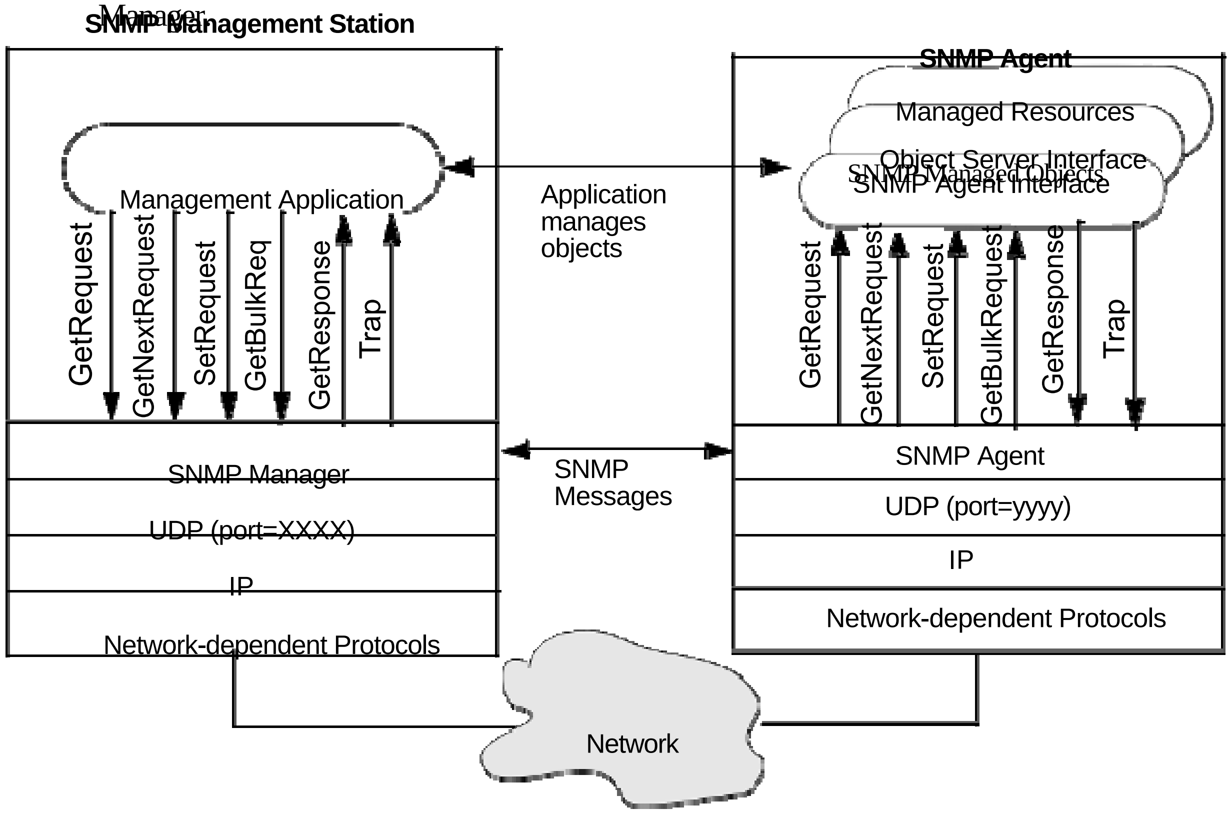


Figure 6-3: SNMP Architecture

The Signaling Gateway Client software package comes with an SNMP Agent that supports both SNMP v1 and v2. All Signaling Gateway Client MIBs are defined by the *ebs* branch under the .internet.private.enterprises subtree. The Distributed7 MIBs are defined by the *.ebs.accessMANAGER* branch, while Signaling Gateway Client MIBs are defined by the *.ebs.sgc* branch. The following figures illustrate the MIB hierarchy on Signaling Gateway Client system (the key fields are bolded).

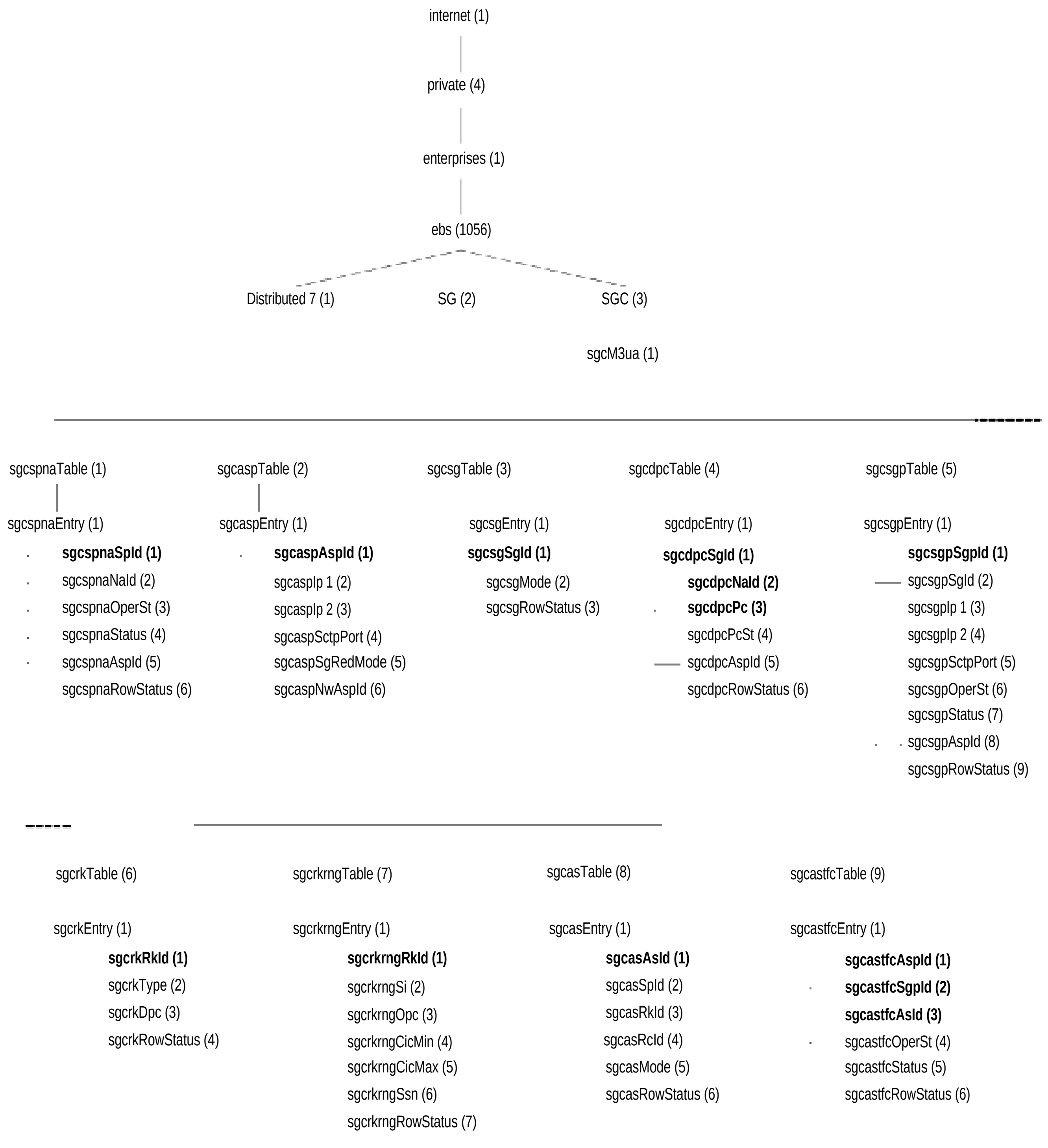


Figure 6-4: Internet MIB Tree

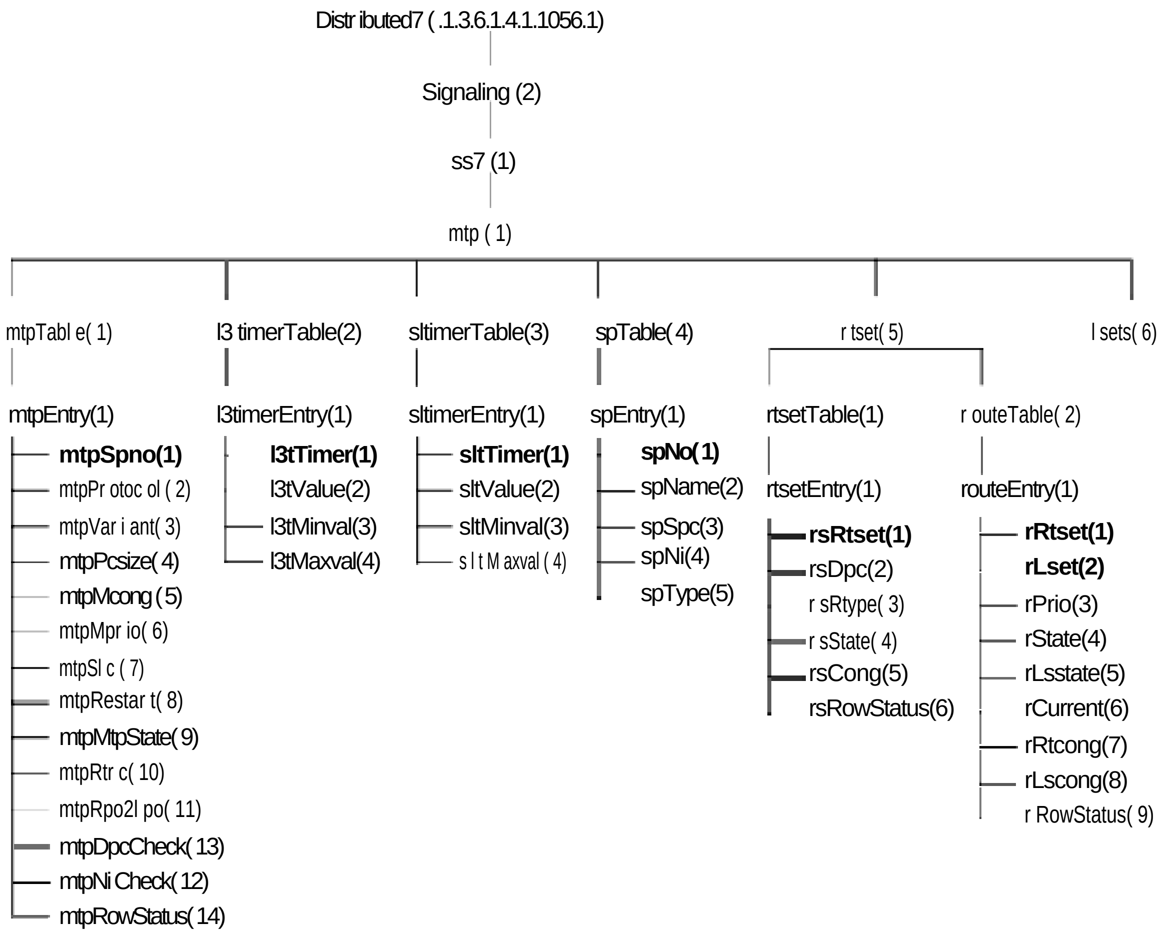


Figure 6-5: Distributed7 MIB - MTP Tree 1

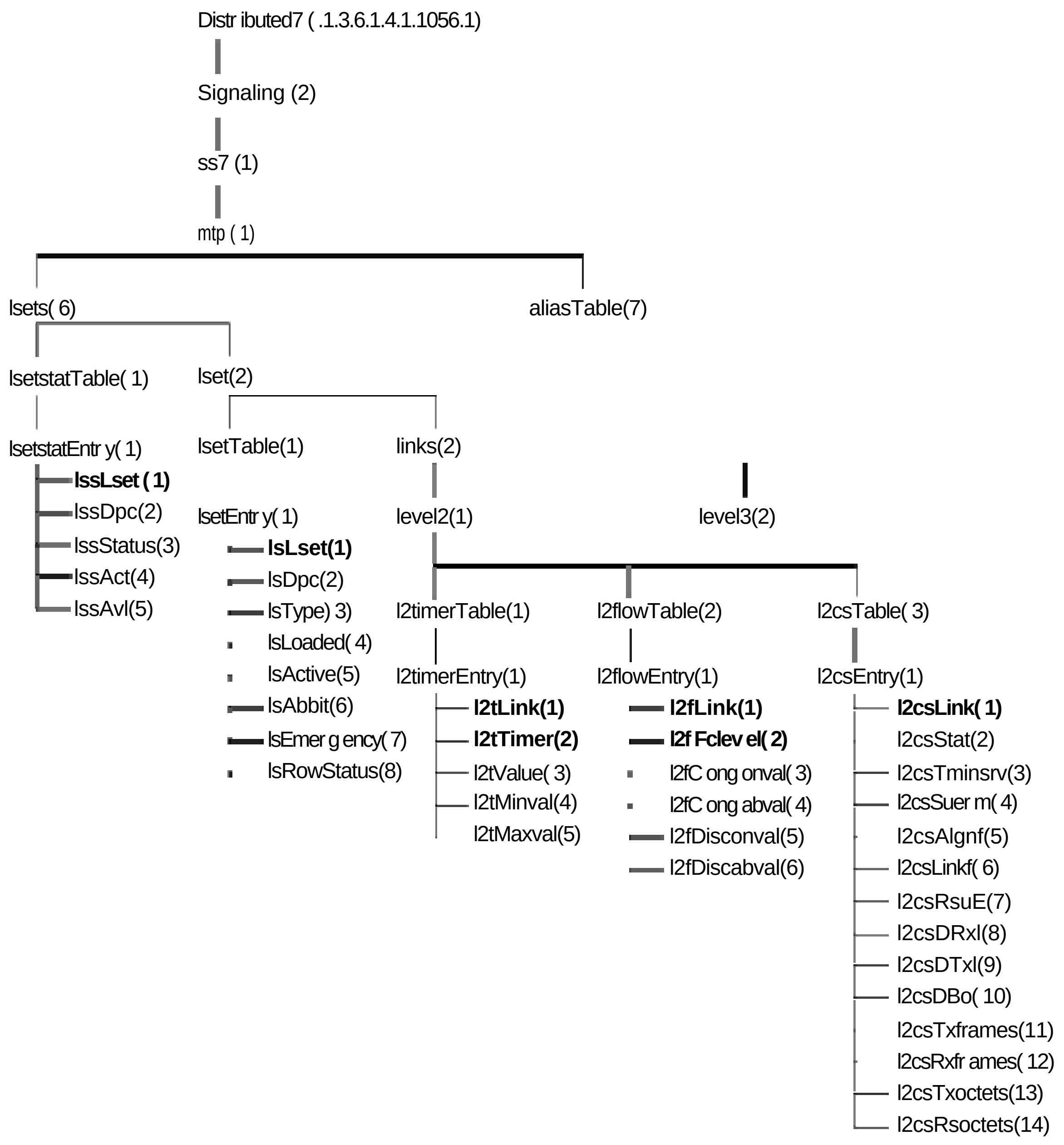


Figure 6-6: Distributed7 MIB - MTP Tree 2

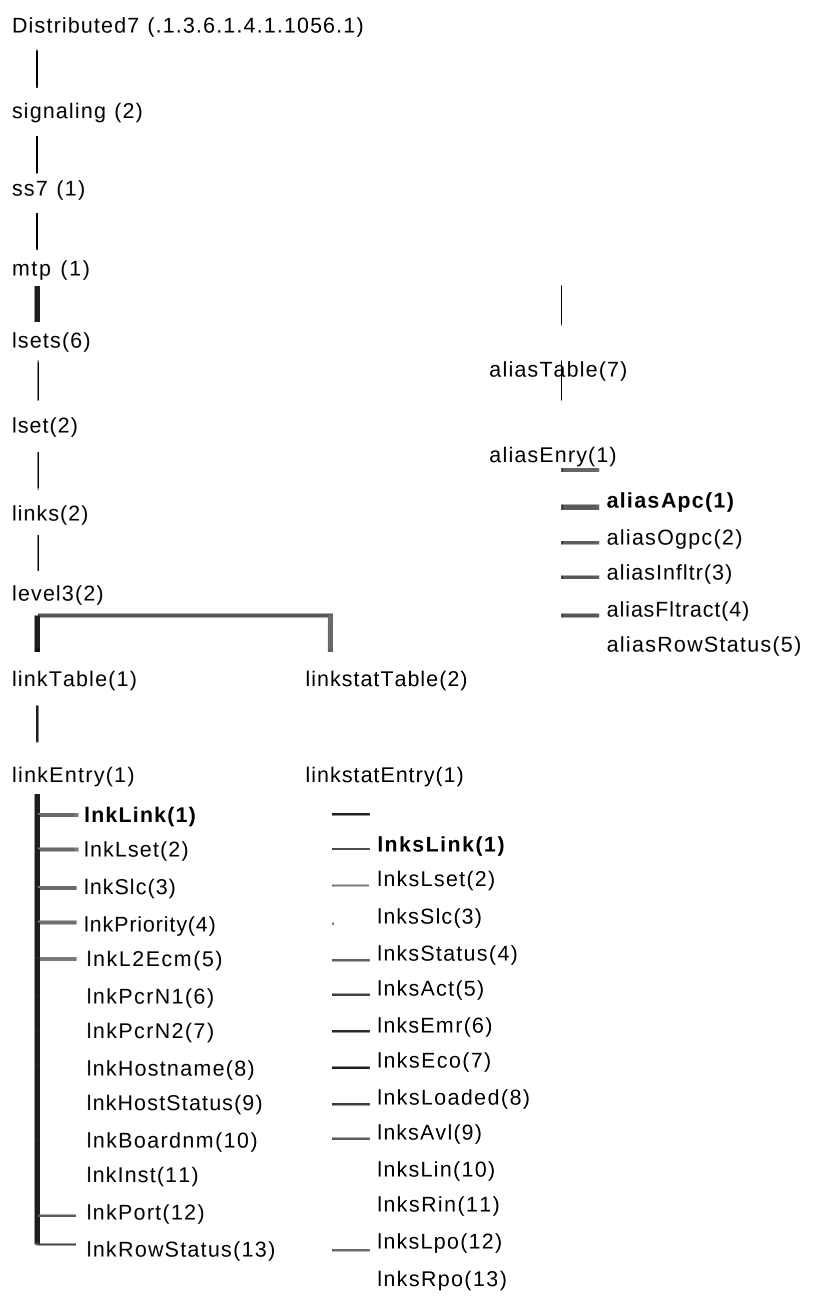


Figure 6-7: Distributed7 MIB - MTP Tree 3

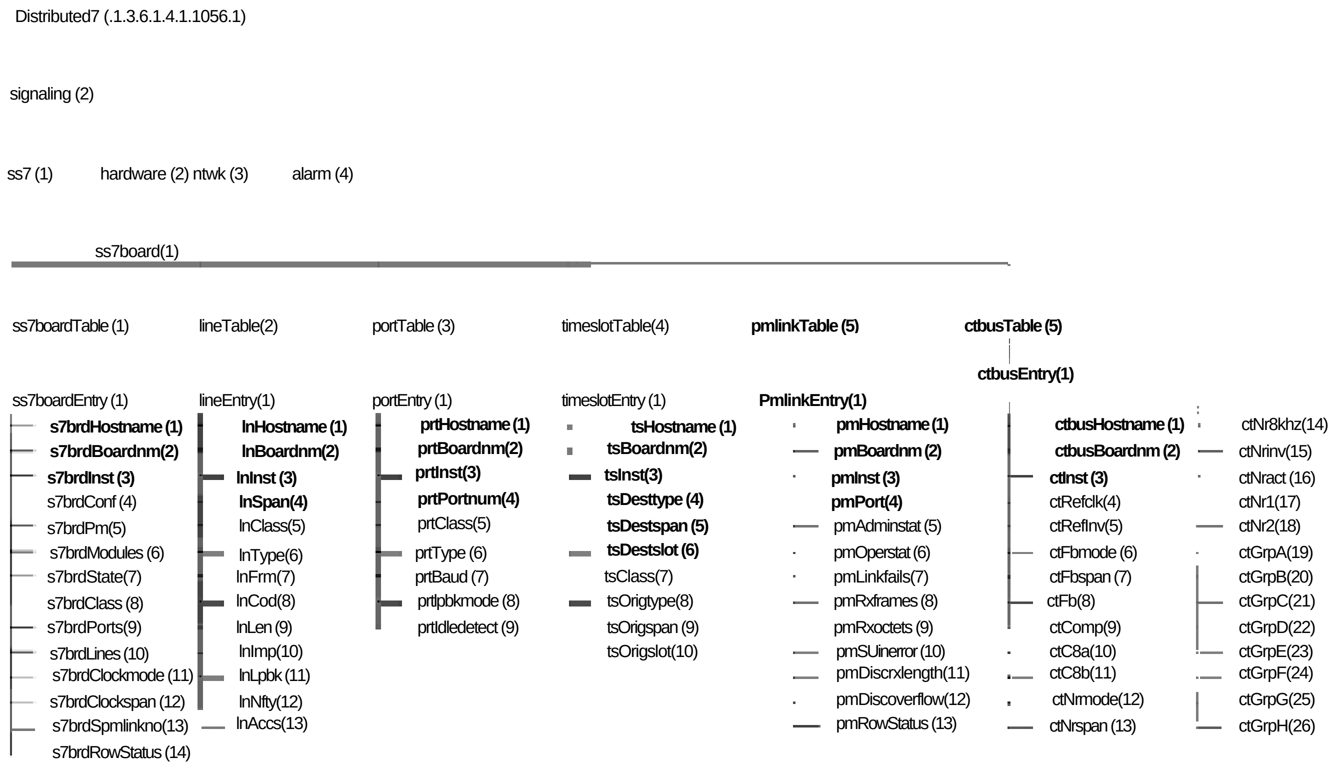


Figure 6-8: Distributed7 MIB - Hardware Tree

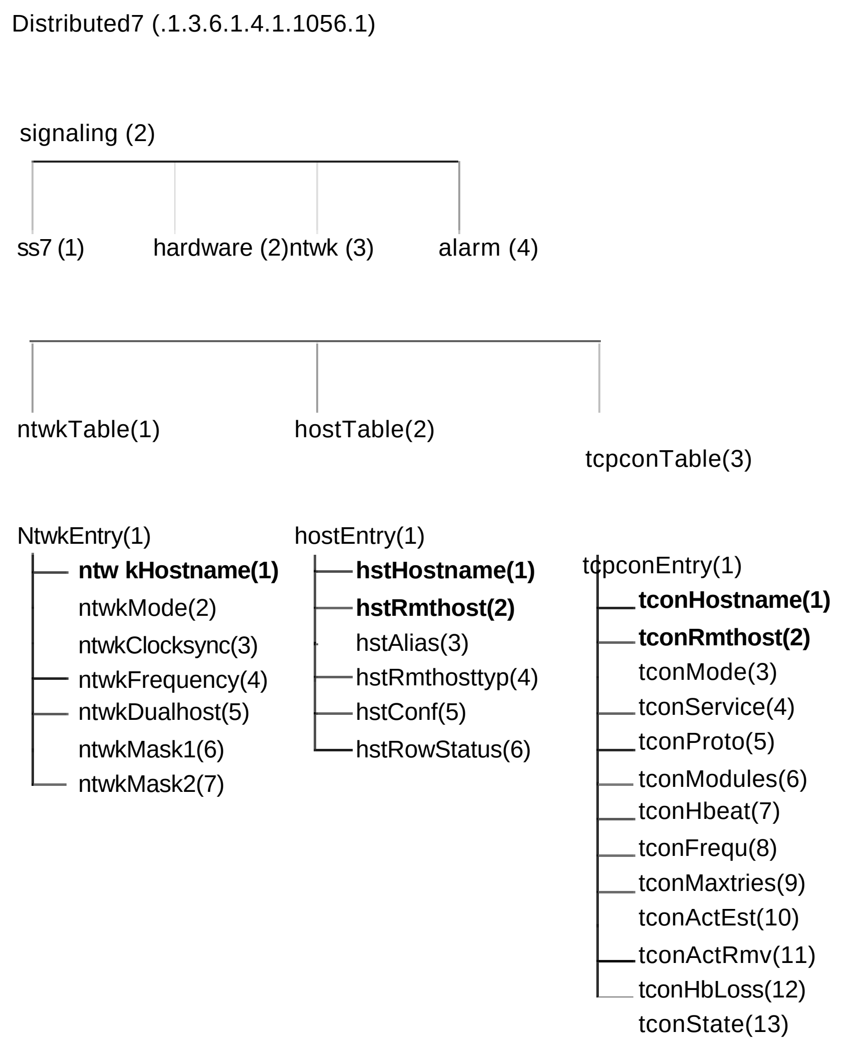


Figure 6-9: Distributed7 MIB - Network Tree

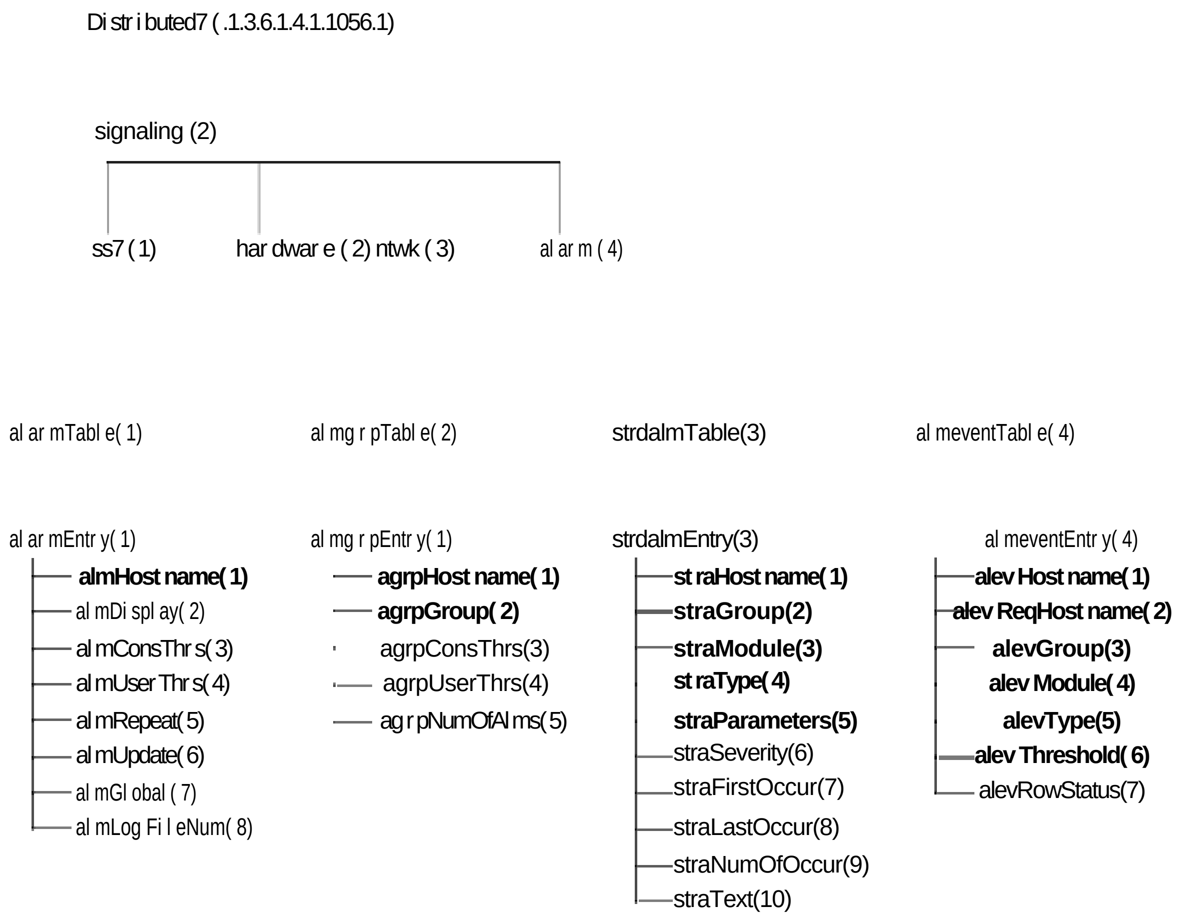


Figure 6-10: Distributed7 MIB - Alarm Tree

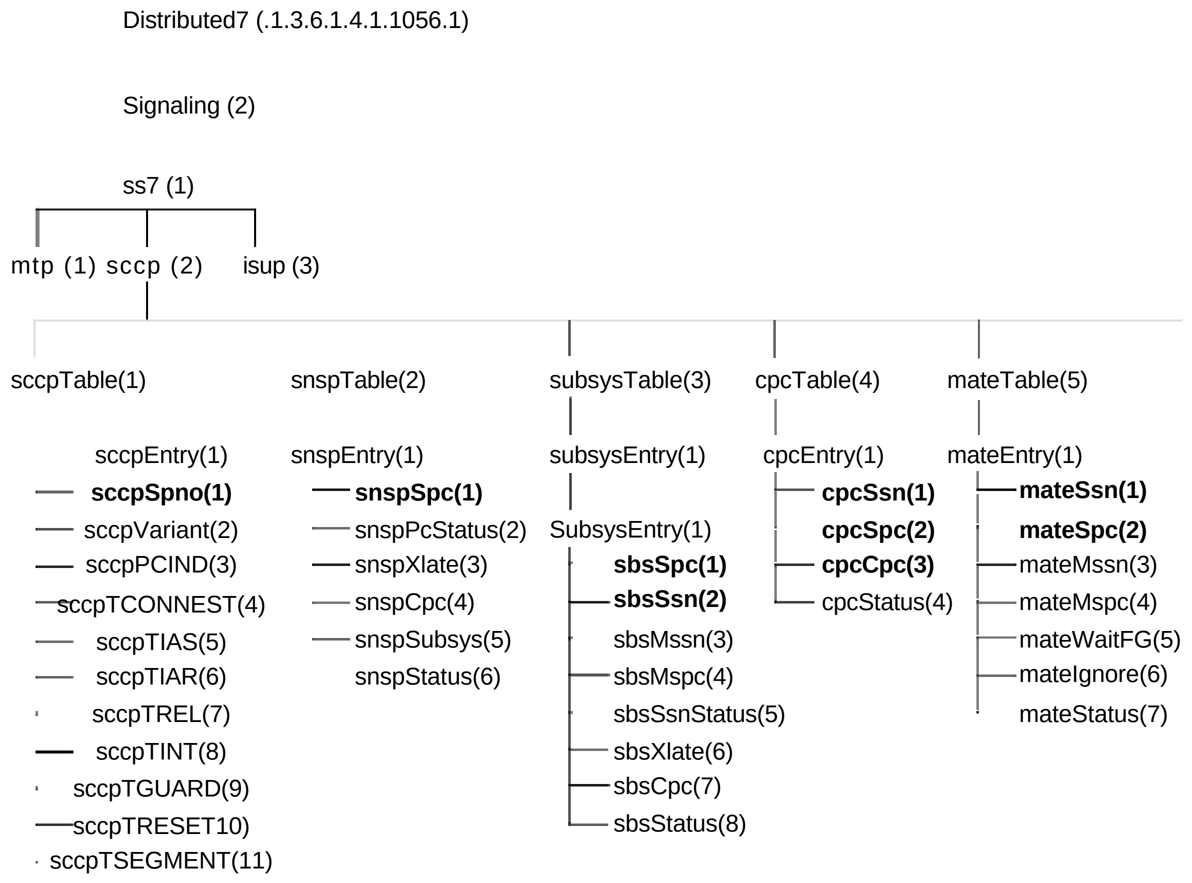


Figure 6-11: Distributed7 MIB (SCCP Layer - 1)

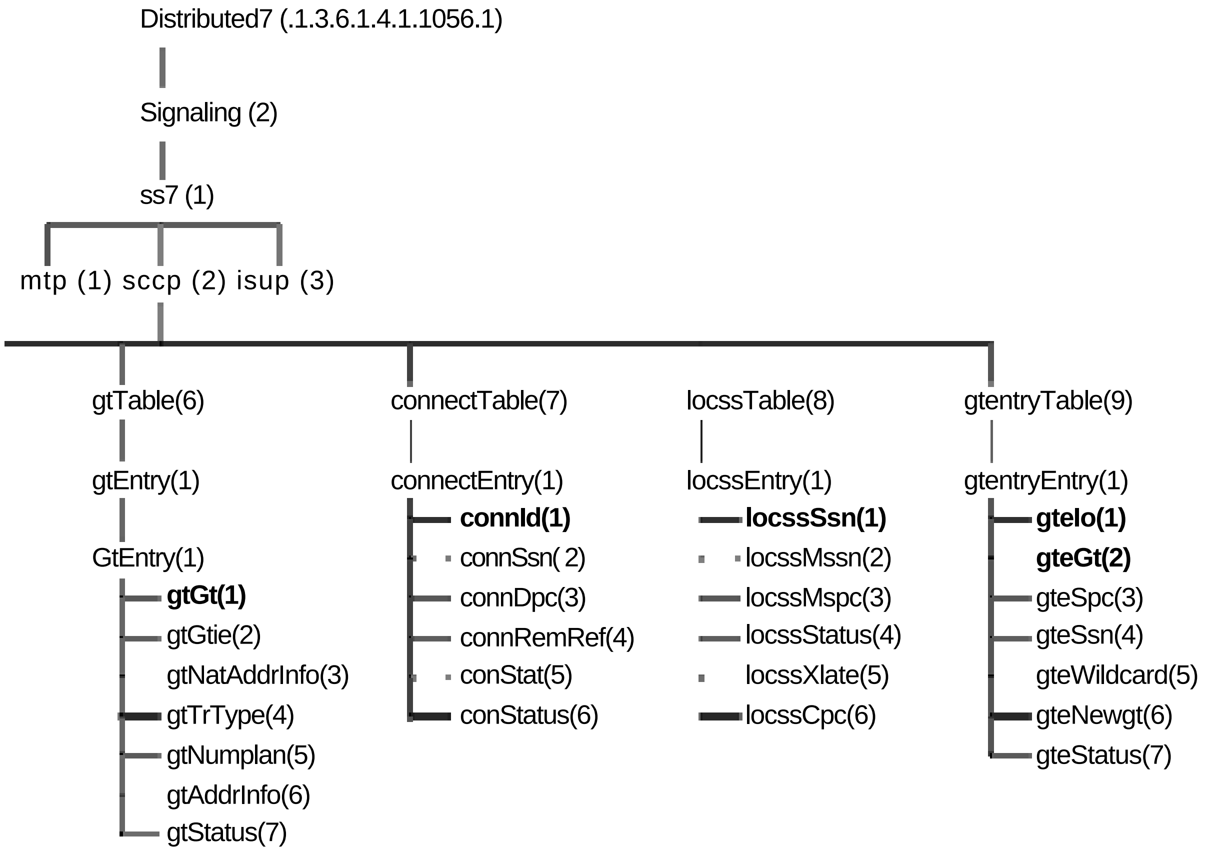


Figure 6-12: Distributed7 MIB (SCCP Layer - 2)

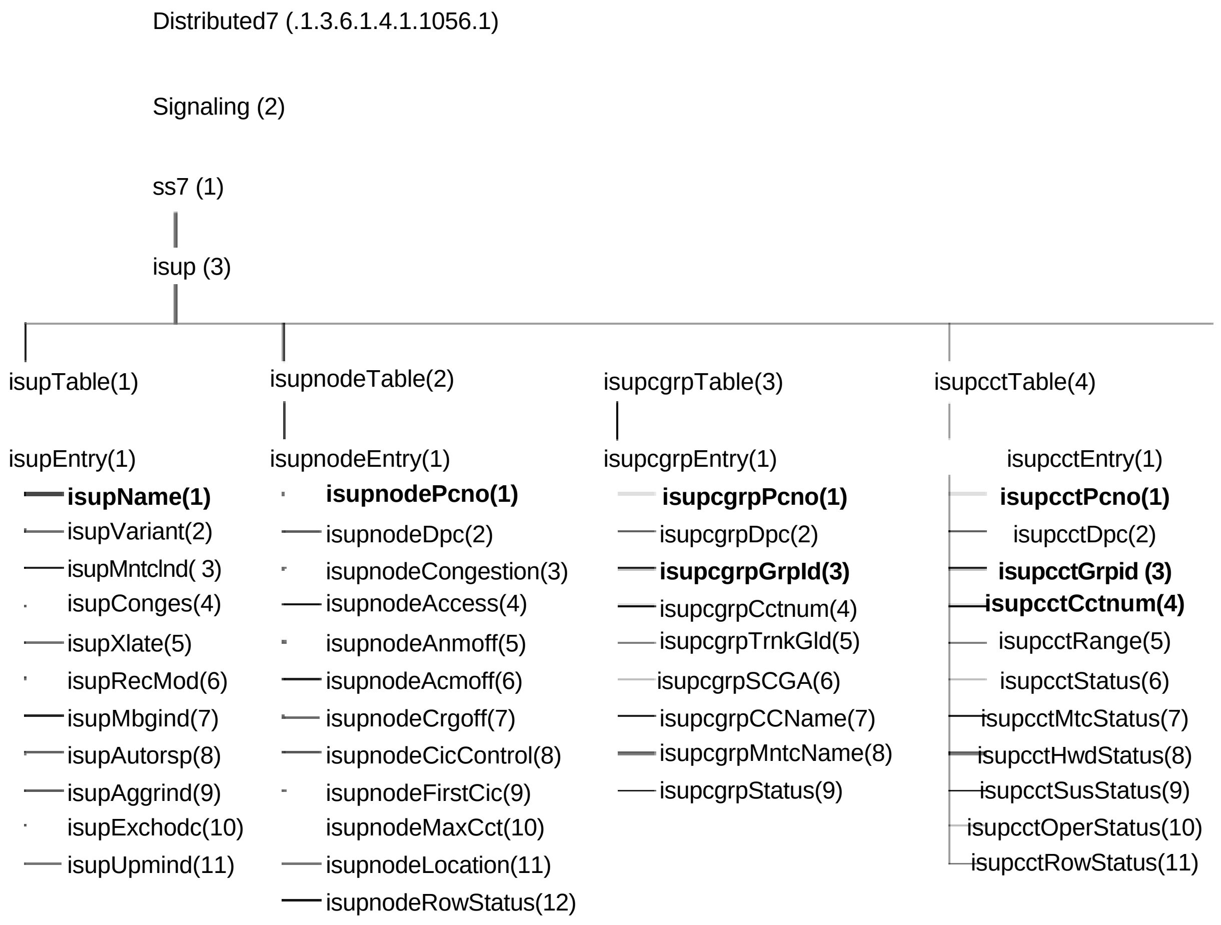


Figure 6-13: Distributed7 MIB (ISUP Layer - 1)

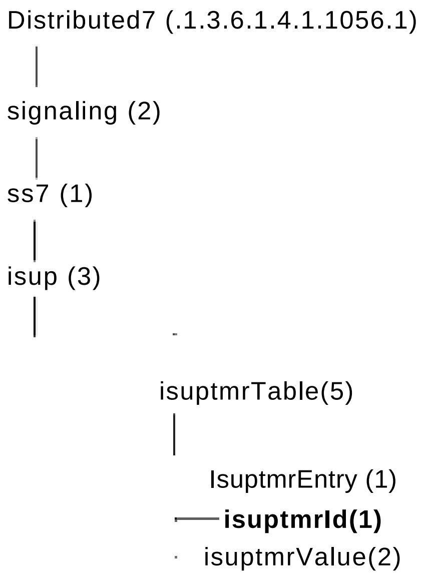


Figure 6-14: Distributed7 MIB (ISUP Layer - 2)

6.5.7.1 Configuring SNMP Agent

The MIBs and SNMP agent configuration files must be properly configured and placed in the appropriate directories to manage Signaling Gateway Client using SNMP. The following
table describes the location and meaning of all the configuration files involved.

Table 6-4: SNMP Configuration Directories and Files

Directory	Configuration File	Description
$EBSHOME/access/RUN/config/SNMP	mib_text.v1	MIB modules for SNMP v1. No modification is required in this file.
	mib_text.v2	MIB modules for SNMP v2. No modification is required in this file.
	snmp_cmnd.tbl	SNMP and MO mapping table. No modification is required in this file.
$EBSHOME/access/RUN/config/SNMP/etc	trap.ini (SNMP v1)	Template that defines where SNMP traps must be sent. Must be copied to $EBSHOME/access/RUNx/ config/SNMP/trap.conf.
	community.ini (SNMP v1)	Template that defines the community attributes. Must be copied to $EBSHOME/access/RUNx/config/ SNMP/community.conf.
	party.ini (SNMP v2)	Template that defines the party attributes. Must be copied to $EBSHOME/access/RUNx/config/SNMP/ party.conf.
	context.ini (SNMP v2)	Template that defines the context attributes. Must be copied to $EBSHOME/access/RUNx/config/SNMP/ context.conf.
	view.ini (SNMP v2)	Template that defines the supported MIB view. Must be copied to $EBSHOME/access/RUNx/config/ SNMP/view.conf.
	acl.ini (SNMP v2)	Template that defines the access policy. Must be cop-ied to $EBSHOME/access/RUNx/config/SNMP/ acl.conf.
$EBSHOME/access/RUN*x*/config/SNMP	trap.conf	Defines where traps are sent.
where *x* is the signaling point number.		Defines community attributes.
	community.conf	Defines party attributes.
	party.ini	Defines context attributes.
	context.ini	Defines MIB view.
	view.ini	Defines access policy.

acl.ini

- Determine which version of SNMP, v1 or v2, you are using before configuring anything.
- Then, based on Table 6-4, copy the necessary configuration templates named as
  $EBSHOME/access/RUN/config/SNMP/etc/*.ini
  to
  $EBSHOME/access/RUNx/config/SNMP/*.conf,
  where **x** is the signaling point number on which the SNMP Agent should run.

- If both SNMP v1 and v2 are used, then copy all the configuration templates. Otherwise, only copy the set of files needed for the version you are using. Based on the version, complete the following instructions to edit the configuration files.

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 identifying 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:

```
                        remote manager net address remote port number
#community name         yyy.yyy.yyy.yyy  xxx
public
```

SNMP v2 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 Signaling Gateway Client SNMP Agent supports the following subtrees:

- accessMANAGER (1.1.3.6.1.4.1.1056.1) - Distributed7-related MIBs
- sgc (1.1.3.6.1.4.1.1056.3) - Signaling Gateway Client application-related MIBs

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
```

6.5.7.2 Starting the SNMP Agent

The SNMP Agent must be started ***BEFORE*** the system can be managed remotely from a management station. Run the following command to start an SNMP Agent for a particular signaling point:

AccessSNMP -v version sp

where

- *version* is the SNMP version, either **1** or **2**
- *sp* is the signaling point number from 0 to 7.

6.6 UsingAccessMOB

6.6.1 Introduction

The Graphical User Interface (GUI) for Signaling Gateway Client is called the Managed Object Browser (MOB). This interface displays the hierarchical model of Signaling Gateway Client 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:

- Signaling Gateway Client 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 <u>must</u> 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-Signaling 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 Signaling Gateway Client 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 signaling point number (0, 1, 2, 3, 4, 5, 6, or 7) of a Signaling Gateway Client 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 Signaling Gateway Client environment. Multiple copies of the MOB can exist for the same signaling 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 <u>NOT</u> be run in the background (<u>do not</u> 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 <u>E</u>xit).

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 <u>not</u> 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 and the arrow keys Insert, Home, End ↑ ↓ → ← 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 sub-string 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 6-15) 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*. 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 6-16 on page 6-176. 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 6-16). 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 Wild Cards section.)
- Must have a character length within the specified range.

Wild Cards

The special character, *, represents a *wild card* value. A wild card 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 wild card usage exist:

- Existing Managed Objects only accept ONE wild card 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 wild card as a value.
- Wild cards can only be used in the View operation. The other operations require full specification of a unique instance.
- Wild cards 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 6-15 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.

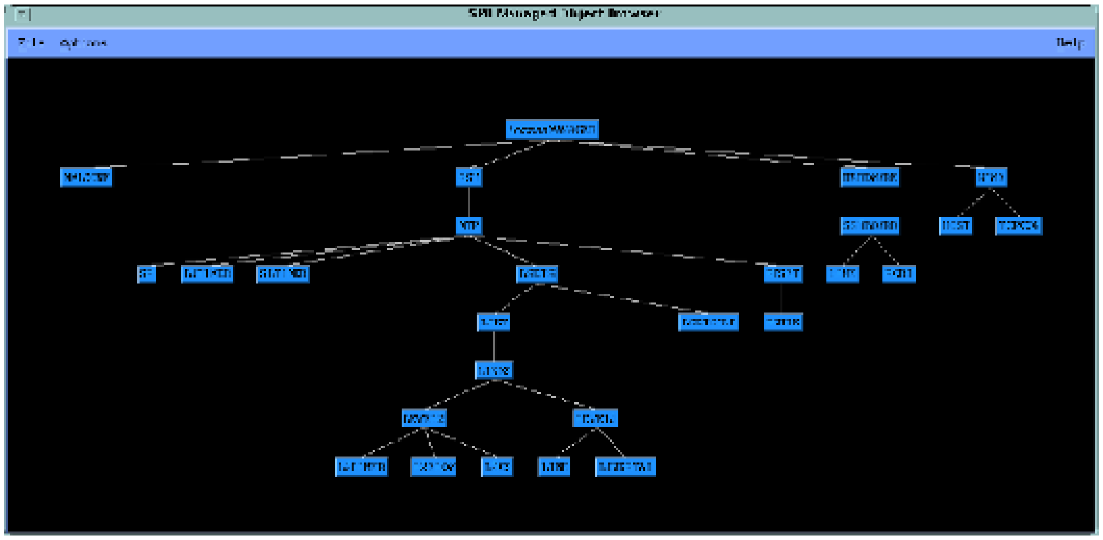


Figure 6-15: MOB Main Window

The main window of the MOB contains a tree representing the managed objects in your Signaling Gateway Client 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 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 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 W). 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 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 6-167*.

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 6-16 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 6-168.

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.

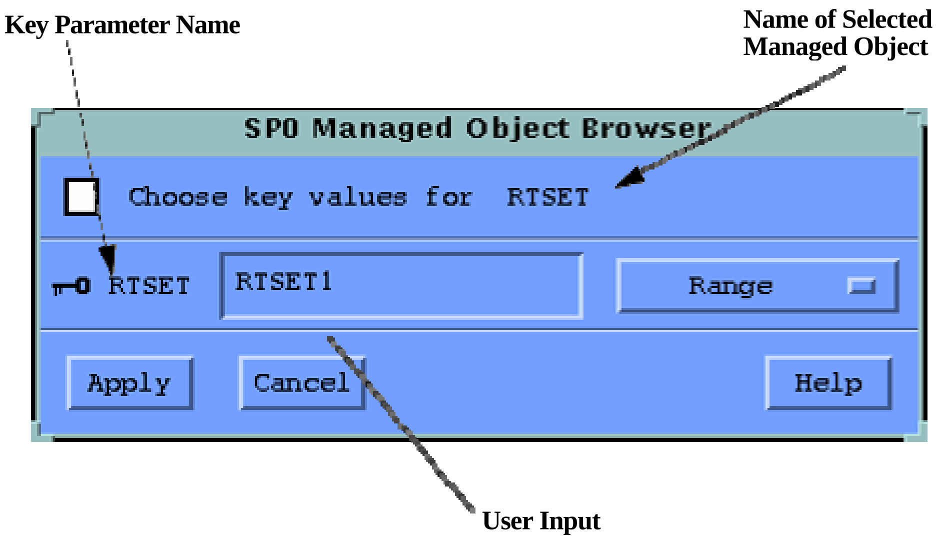

Key Parameter Name
Name of Selected Managed Object
SP0 Managed Object Browser
Choose key values for RTSET
RTSET
RTSET1
Range
Apply
Cancel
Help
User Input

Figure 6-16: 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 [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 6-17 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.

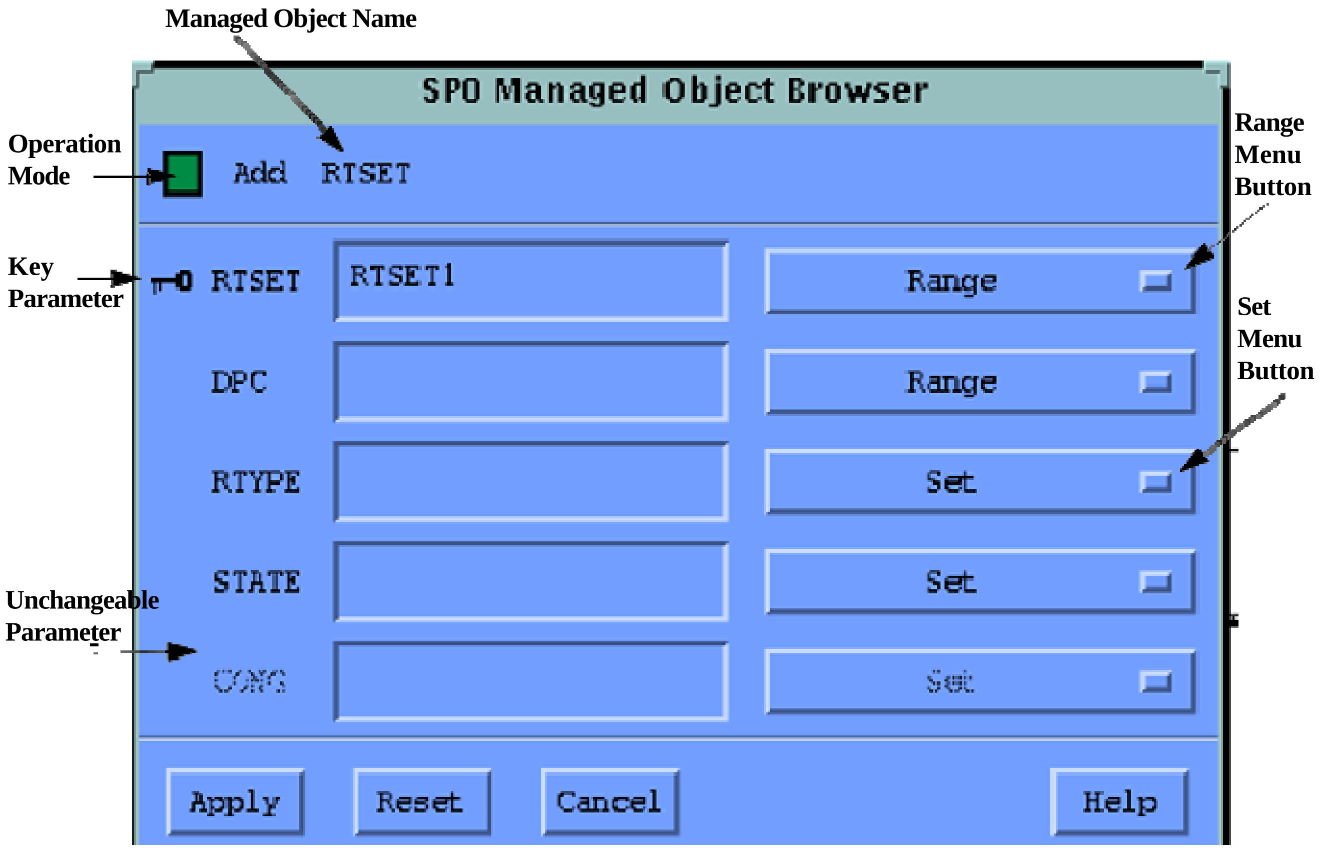


Figure 6-17: Add Dialog Box

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 then Tab edit the value. Any parameters shown in grey (e.g. CONG in Figure 6-17) 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.

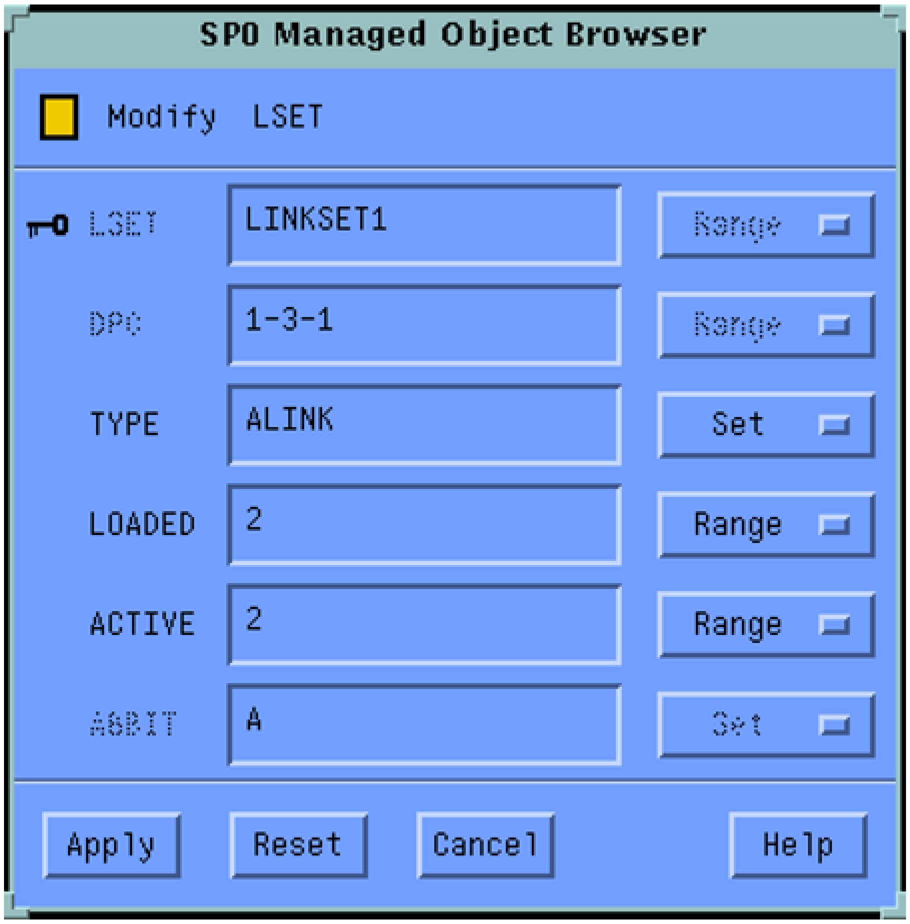
To add a managed object instance:

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 6-16 on page 6-176)
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 6-18 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.


SPO Managed Object Browser

Modify LSET

LSET	LINKSET1	Range
DPC	1-3-1	Range
TYPE	ALINK	Set
LOADED	2	Range
ACTIVE	2	Range
ABBIT	A	Set

Figure 6-18: 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 Esc 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 6-16 on page 6-176)
4. Enter the values to be modified in the Modify operation dialog box.
5. Click the ***Apply*** button.

Delete Dialog Box

Figure 6-19 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.

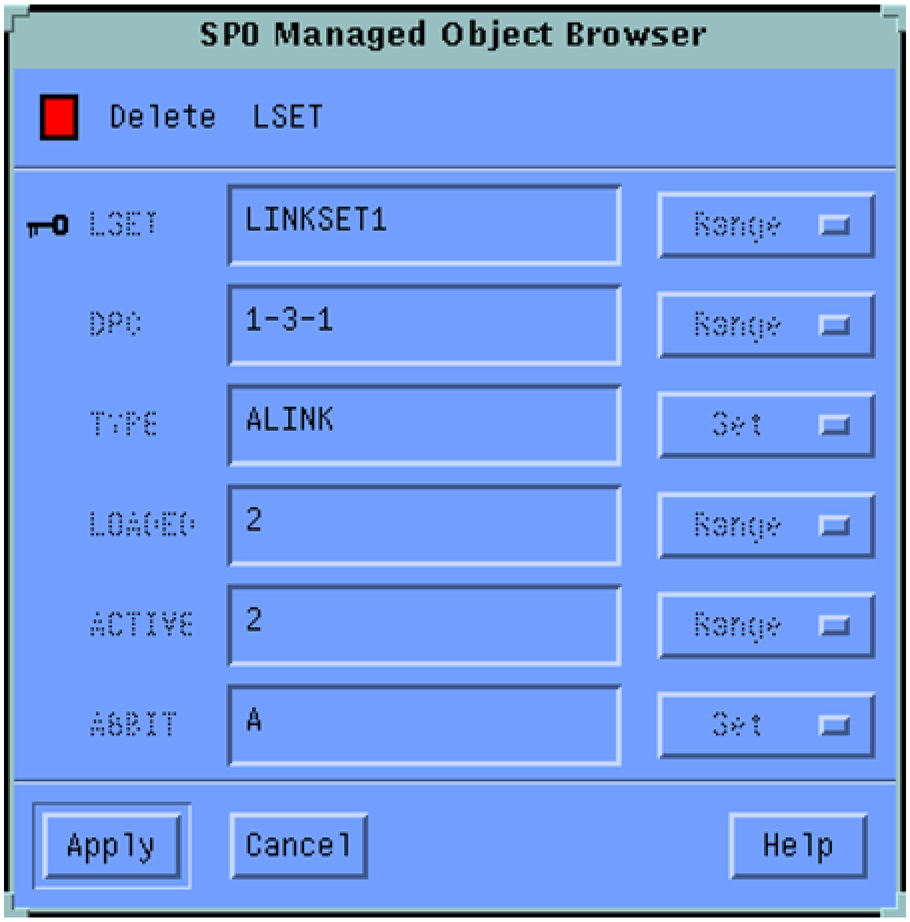


Figure 6-19: Delete Dialog Box

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 6-16 on page 6-176)
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 6-16 on page 6-176)

A window will appear showing the parameters of the single instance.

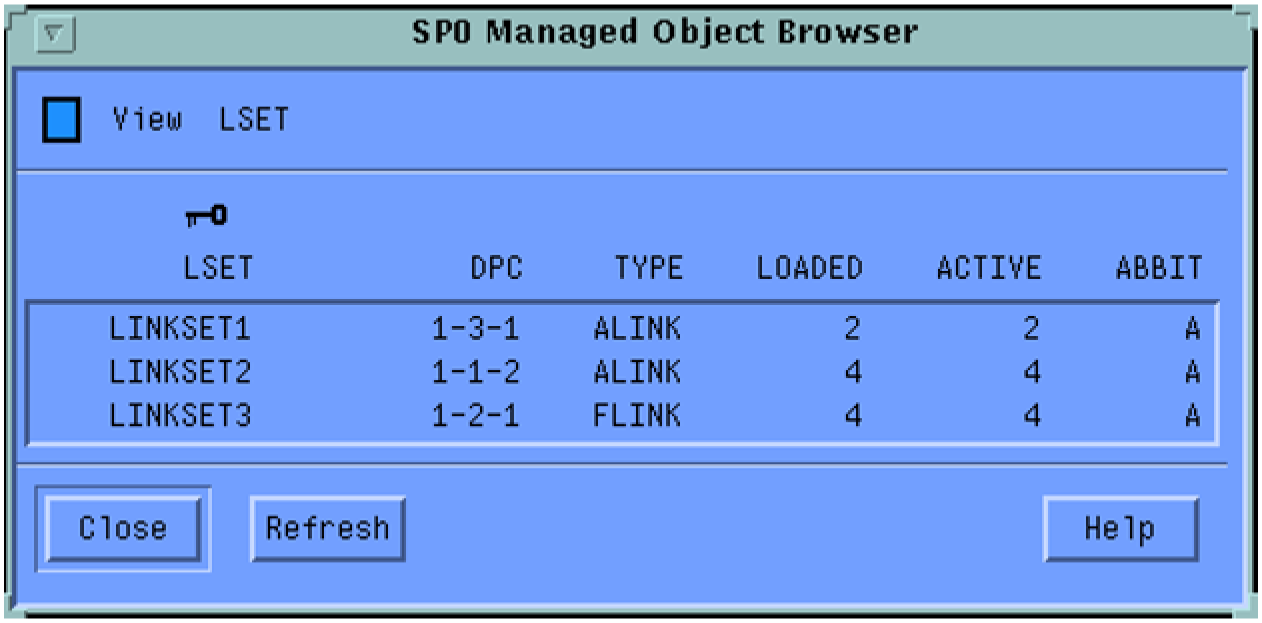
To view all instances:

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 6-20 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.



LSET	DPC	TYPE	LOADED	ACTIVE	ABBIT
LINKSET1	1-3-1	ALINK	2	2	A
LINKSET2	1-1-2	ALINK	4	4	A
LINKSET3	1-2-1	FLINK	4	4	A

Figure 6-20: 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 6-174). 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 any 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 signaling 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>

Related Information

- Section 9.7.15, AccessStatus on page 9-473

Chapter 7: MML Commands

7.1 ChapterOverview

This chapter has detailed information about using MML commands to configure Signaling Gateway Client as outlined in Chapter 6: Operations, Administration, and Maintenance.

There are two basic types of commands used: MML commands and shell commands. All MML commands are entered at the `MML_#>` prompt, where # is the signaling point number. The shell commands are entered at the UNIX command prompt. Some shell commands and common UNIX commands are presented in Chapter 9: User Commands.

7.1.1 MML Commands

Use the **mml** -a# (where # is the signaling point number) utility command to start the MML interface from which the operator can enter the commands to configure, administer and maintain the system.

The MML commands are grouped as the following managed objects:

• ***System*** Managed Objects configure and maintain the SS7 User Parts of the system.

- The operator uses these functions to configure SCCP, ISUP and cluster LANs.

- Table 7-2, Table 7-3 and Table 7-4 list all the SCCP, ISUP and system managed objects, parameter names, default Values, units, value ranges, and operation commands.

• ***Signaling Gateway Client*** Managed Objects configure, maintain and monitor the Signaling Gateway Client, including its network appearance for both SS7 user parts and M3UA.

- Table 7-5 lists all managed objects, parameter names, default values, units, value ranges, and operation commands.

Use the **HELP** command to view online information about specific MML commands.

7.1.2 Managed Objects

A managed object represents data with a set of parameters and command operations. The SS7 User Parts have their own managed objects.

7.1.3 MML Network Element Labels

The MML command labels for network elements are defined as follows:

- The value for SG, ASP and ASID is a number less than 11Ø.
- Hostname: This is a 15-character label that can include any number of alphanumeric characters, numbers and hyphens.
- Point Codes (DPC, OPC and SPC): Signaling point code are 11-character 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	Ø - 255	Ø - 255	Ø - 255

16-bit ITU:

Fields	**Zone**	**Area**	**SP**
Format	5 bits	4 bits	7 bits
Value Range	Ø - 31	Ø - 15	Ø - 127

14-bit ITU:

Fields	**Zone**	**Area**	**SP**
Format	3 bits	8 bits	3 bits
Value Range	Ø - 7	Ø - 255	Ø - 7

***Note**: Leading zeros (Ø) are not necessary.*

7.1.4 Syntax Rules for the Command Line

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 begin with ***H'***.
3. Octal values must begin 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.

7.1.5 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)

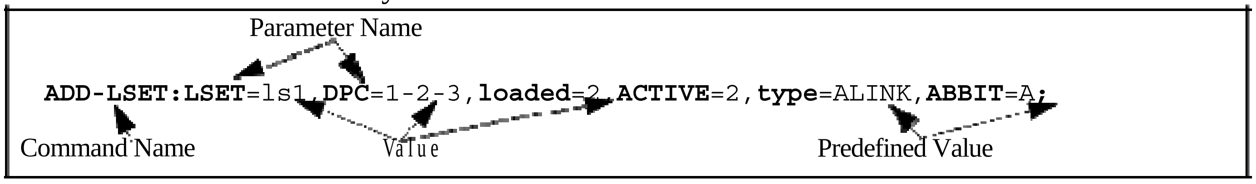
MODIFY-MTP:SLTC=O; (INVALID)

MODIFY-MTP:SLTC=OF; (VALID)

MODIFY-MTP:SLTC=ON; (VALID)

7.1.6 Case Sensitivity

MML is ***NOT*** case sensitive with respect to command and parameter names. However, the values entered for a parameter may be case-sensitive. The following figure illustrates the rules of case-sensitivity in MML commands.



```
ADD-LSET:LSET=ls1,DPC=1-2-3,loaded=2,ACTIVE=2,type=ALINK,ABBIT=A;
```

Figure 7-1: 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 an parameter, such as ALINK, must be typed in the same case as their definitions. For example, Signaling Gateway Client's SS7 user part-related, predefined values are lower case, and Signaling Gateway Client's M3UA, predefined values are upper case.

- All commands, except the add command, have both a short and long format. For example, you may enter either **`DEL-NA`** or **`DELETE-NA`**, **`DIS-NA`** or **`DISPLAY-NA`** and **`MOD-NA`** or **`MODIFY-NA`**.

7.1.7 Command Syntax

The following explains how the syntax for each command is documented in this manual:

- The **COMMANDS** section lists the commands that can be used with each managed object: ADD, MODIFY, DELETE and/or DISPLAY.
- Each command is followed by the command syntax:
  - The *command* and *parameters* are in ***UPPER CASE, BOLD ITALICS*** typeface.
  - The *values* for a parameter are the word as the parameter, but are lower case, regular typeface.
  - These lower case values are listed in the **PARAMETERS** section with a description of what the parameter does, and what can be entered as the value.
  - Optional parameters are in square brackets, ***[]***, and are listed after the required parameters.
  - The order in which the parameters are enter does not matter. However, the required parameters are shown before the optional parameters in the COMMAND section.

The following is an example of the command syntax for the ASP command:

MODIFY-ASP:ASPID=aspid***[,SGMODE***=sgmode***][,OPERSTATE***=operstate***];***

Any read-only parameters that appear in the output of the DISPLAY command are NOT listed in the **PARAMETERS** section, but are in a table after the **SAMPLE OUTPUT**. The following is the table that appears after the SCCP **SAMPLE OUTPUT:**

Table 7-1: SCCP DISPLAY VALUES

SPNP
Signaling Point number that is an integer from Ø to 7.

7.1.8 Output Messages

MML prints "`<SUCCESS>`" when a command runs successfully. It prints "`- mml error: error message`." if a command fails. 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:

1. - mml error: no such managed object

1. - mml error: no such operation for this MO

2. - mml error: inconsistent PDU size

3. - mml error: MO instance already provisioned

4. - mml error: MO instance not provisioned

5. - mml error: parameter out of range

6. - mml error: mandatory parameter missing

7. - mml error: no such parameter for this MO

8. - mml error: no space in database to add MO instance

9. - mml error: internal error

10. - mml error: licensed limit reached

Table 7-2: SCCP Configuration Managed Objects

Option	Parameter Name	Value	Unit	Range		Command Operation
CPC (Concerned Point Code Managed Object - see Section 7.2.1 on page 7-204)	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 7.2.2 on page 7-206)	GT	-	bits	8		ADD DELETE DISPLAY
	GTIE	-	-	1 to 15		
	TRTYPE	-	-	Ø 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: The default values are in italics.

Table 7-2: SCCP Configuration Managed Objects (Continued)

Option	Parameter Name	Value	Unit	Range		Command Operation
GTENTRY (Global Title Entry Managed Object - see Section 7.2.3 on page 7-208)	IO	INCOMING OUTGOING	-	-		ADD DELETE DISPLAY
	GT	-	-	1 to 131,Ø72		
	ENTRYTYPE	PRIMARY SECONDARY	-	-		
	XLATE_ID	-	Alpha Numeric characters	1 to 12 characters		
	SPC	-	-	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)	
	SSN	-	-	2 to 255		
	NEWGT	-	numeric	1 to 4		
	WILDCARD	YES *NO*	-	-		

Note: The default values are in italics.

Table 7-2: SCCP Configuration Managed Objects (Continued)

Option	Parameter Name	Value	Unit	Range		Command Operation
MATE (Mate Managed Object - see Section 7.2.4 on page 7-210)	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		
SCCP (SCCP Managed Object - see Section 7.2.5 on page 7-212)	PROTOCOL	*DEFAULT* ANSI_92 ANSI_96 ITU_93 ITU_97	-	-		DISPLAY MODIFY
	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	-		-		

Note: The default values are in italics.

Table 7-2: SCCP Configuration Managed Objects (Continued)

Option	Parameter Name	Value	Unit	Range		Command Operation
SNSP (SCCP Signaling Point Managed Object - see Section 7.2.6 on page 7-214)	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)	
SUBSYS (Subsystem (Managed Object - see Section 7.2.7 on page 7-216)	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 7.2.8 on page 7-218)	-	-	-	-		DISPLAY
CONNECTION (Connection Managed Object - see Section 7.2.9 on page 7-219)	ID	-	-	Ø to 16383 or * for all		DISPLAY

Note: The default values are in italics.

Table 7-3: ISUP Configuraton Managed Objects

Option	Parameter Name	Value	Unit	Range	Command Operation
ISUPCCT (ISUP Circuits Managed Object - see Section 7.3.1 on page 7-220)	PCNO	-	-	-	ADD DELETE DISPLAY MODIFY
	GRPID	-	integer	Ø to 3Ø39	
	CCTNUM	-	integer	Ø 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 7.3.2 on page 7-224)	PCNO	-	-	-	ADD DELETE DISPLAY MODIFY
	GRPID	-	integer	Ø to 3Ø39	
	CCTNUM	-	integer	Ø to max cct # defined for this node	
	TRNKGRPID	-	integer	Ø to 8191	
	SCGA	ON OFF	-	-	

Note: The default values are in italics.

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 7-3: ISUP Configuraton Managed Objects (Continued)

Option	Parameter Name	Value	Unit	Range	Command Operation
ISUPNODE (ISUP Signaling Node Managed Object - see Section 7.3.3 on page 7-227)	PCNO	-	Integer	Ø to 2Ø47	ADD DELETE DISPLAY MODIFY

	DPC[2]	-	-	for CCITT: Zone-Network-SPID
				for ANSI: Network-Cluster-Member	

	ANMOFF	ON OFF	-	-	
	ACMOFF	ON OFF	-	-	
	CRGOFF	ON OFF	-	-	
	CICCONTROL	ODD EVEN ALL NONE DEFAULT	-	-	
	LOCATION	For ITU: • LOCUSER • PUBNETLO-CUSER • PRVNETREM-USER • PRVNETLO-CUSER • TRANSNET PUBNETREM-USER • LOCINTER • INTERNATNET BEYINTWORK-PNT For Spain: • LOCUSER • PUBNETLO-CUSER • PRVNETREM-USER • PRVNETLO-CUSER • TRANSNET PUBNETREM-USER • LOCINTER • INTERNATNET BEYINTWORK-PNT • PCKHNDNAT For ANSI: • LOCUSER	Character	-	

Table 7-3: ISUP Configuraton Managed Objects (Continued)

Option	Parameter Name	Value	Unit	Range	Command Operation
ISUP (ISUP Configuration Managed Object - see Section 7.3.4 on page 7-230)	CFGNAME	CF<sp#>	-	-	MODIFY DISPLAY
	VARIANT	For ANSI: GENERIC, ANSI92, ANSI96, BELL, DSC, MCI For ITU: GENERIC, AUSTRA-LIA, BELGIUM, CHILE, CHI24, CZECH, ETSI97, FINLAND, FRANCE, GERMANY, HONG KONG, ITALY, ITU92, ITU97, MEX-ICO, NEW_ZEALAND, NORWAY, PHILIP-PINES, Q767, RUS-SIA, SINGAPORE, SPAIN, SWEDEN, SWEDENVI, SWIT-ZERLAND, THAI-LAND, TURKEY, UAE, UNIPAC	alphanumeric	-	
	MNTCIND	ON OFF GRPINDON	-	-	
	CONGES	ON OFF	-	-	
	RECMODE	RESCALL RELCALL		-	
	AUTORESP	*ON* OFF	character	-	
	EXCHODC	ON *OFF*	character	-	
	UPMIND (valid for ITU only)	*ON* OFF	character	-	
ISUPTMR (ISUP Timer Managed Object - see Section 7.3.5 on page 7-233)	TIMERID	-	milliseconds	1 to n	DISPLAY MODIFY
	VALUE	-	milliseconds	10 msec to 25 hours	

Note: The default values are in italics.

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 7-4: System Configuration Managed Objects

Name	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
HOST (Host Managed Object - see Section 7.4.1 on page 7-237)	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	-	
NTWK (Network Managed Object- see Section 7.4.2 on page 7-239)	HOSTNAME	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	MODIFY DISPLAY
	MODE	STNDLN *DSTRBTD*	characters	-	
	CLOCKSYNC	*ON - on* OFF - off	characters	-	
	FREQUENCY	*Ø* - stand alone *1ØØØ* - distributed	milliseconds	6Ø - 1ØØØØ (for distributed mode)	
	DUALHOST	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	
	NETMASK1	*7fØØØØØØ*Class A *3fffØØØØ*Class B *1fffffØØ*Class C	-	32-bit mask in hex format used to extract primary network ID	
	NETMASK2	*7fØØØØØØ*Class A *3fffØØØØ*Class B *1fffffØØ*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 that ARE 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 7-5: Signaling Gateway Client Configuration Managed Objects

Name	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
SGCAS (Application Server Managed Object - see Section 7.5.1 on page 7-241	ASID	-	alphanumeric	String of 1 - 15 alphanumeric characters	ADD MODIFY DELETE DISPLAY
	SPID	-	integer	Ø - 7	
	RKID	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	
	RCID	-	integer	Ø - 0x7fffffff	
	MODE	OVERRIDE LOADSHARE	alphanumeric characters	-	
SGCASP (Application Server Process Managed Object - see Section 7.5.2 on page 7-243	ASPID	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	MODIFY DISPLAY
	IP1	-	alphanumeric characters	String of 1 to 15 characters in dot notation format[3]	
	IP2	-	alphanumeric characters	String of 1 to 15 characters in dot notation format[3]	
	HOSTNAME	-	alphanumeric characters	String of 1 to 128 characters	
	SCTPPORT	29Ø5	integer	1ØØØ - 99999	
	SGREDMODE	OVERRIDE LOADSHARE	alphanumeric characters	-	
	NWASPID	0x7fffffff	integer	Ø - Øx7fffffff	
SGCASTFC (AS-ASP Traffic Control Managed Object - See Section 7.5.3 on page 7-245)	ASID	-	alphanumeric	String of 1 - 15 alphanumeric characters	ADD MODIFY DELETE DISPLAY
	SGPID	-	alphanumeric characters	String of 1 to 15 characters	
	ASPID	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	
	OPERST	ACT INACT	alphanumeric characters	-	

1 **Note**: When applicable, default values are shown in italics. Values that ARE 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'.

3 **Note**: An Internet address in dot notation has one to four numbers, and this must be in decimal format. It represents a 32 bit address, where each leading number is eight bits of the address (high byte first) and the last number is the rest. The following is an example of the decimal format: 146.169.22.42.

Table 7-5: Signaling Gateway Client Configuration Managed Objects (Continued)

Name	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
SGCDPC (Destination Point Code Managed Object - see Section 7.5.4 on page 7-247)	SGID	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	ADD MODIFY DELETE DISPLAY
	DPC	-	alphanumeric characters	String of 1 - 11 alphanumeric characters in ZoneID-AreaID-SPID format	
	NAID	-	integer	Ø - Øx7fffffff	
	ASPID	-	alphanumeric characters	String of 1 - 15 alphanumeric	
SGCRK (Routing Key Managed Object - see Section 7.5.5 on page 7-249)	RKID	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	ADD MODIFY DELETE DISPLAY
	TYPE	DPC DPC_OPC DPC_OPC_CIC DPC_SIO DPC_OPC_SIO DPC_CIC_SIO DPC_SSN	characters	-	
	DPC	-	alphanumeric characters	Format: x-y-z, based on the PCSIZE parameter of the NA MO. (see page 7-188 for more information about the point code format.)	

1 **Note**: When applicable, default values are shown in italics. Values that ARE 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'.

3 **Note**: An Internet address in dot notation has one to four numbers, and this must be in decimal format. It represents a 32 bit address, where each leading number is eight bits of the address (high byte first) and the last number is the rest. The following is an example of the decimal format: 146.169.22.42.

Table 7-5: Signaling Gateway Client Configuration Managed Objects (Continued)

Name	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
SGCRKRNG (Routing Key Range Managed Object - see Section 7.5.6 on page 7-251)	RKID	-	alphanumeric characters	String of 1 - 15 alphanumeric characters	ADD DELETE DISPLAY
	SI	SCCP ISUP TUP	characters	-	
	OPC	-	alphanumeric characters	Format: x-y-z, based on the PCSIZE parameter of the NA MO. (see page 7-188 for more information about the point code format.)	
	CICMIN	-	alphanumeric characters	0-65535	
	CICMAX	-	alphanumeric characters	0-65535	
	SSN	-	integer	2 - 255	
SGCSG (Signaling Gateway Managed Object - see Section 7.5.7 on page 7-253	SGID	-	alphanumeric characters	String of 1 to 15 characters	ADD MODIFY DELETE DISPLAY
	MODE	OVERRIDE LOADSHARE	characters	-	

1 **Note**: When applicable, default values are shown in italics. Values that ARE 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'.

3 **Note**: An Internet address in dot notation has one to four numbers, and this must be in decimal format. It represents a 32 bit address, where each leading number is eight bits of the address (high byte first) and the last number is the rest. The following is an example of the decimal format: 146.169.22.42.

Table 7-5: Signaling Gateway Client Configuration Managed Objects (Continued)

Name	Parameter Name	Value[1,2]	Unit	Range[2]	Command Operation
SGCSGP (Signaling Gateway Process Managed Object - see Section 7.5.8 on page 7-255	SGPID	-	alphanumeric characters	String of 1 to 15 characters	ADD MODIFY DELETE DISPLAY
	SGID	-	alphanumeric characters	String of 1 to 15 characters	
	IP1	-	alphanumeric characters	String of 1 to 15 characters in dot notation format[3]	
	IP2	-	alphanumeric characters	String of 1 to 15 characters in dot notation format[3]	
	HOSTNAME	-	alphanumeric characters	String of 1 to 128 characters	
	SCTPPORT	29Ø5	integer	Ø - 32768	
	OPERST	DISCONN CONN UP DOWN	characters	-	
	ASPID	-	alphanumeric characters	String of 1 to 15 characters	
SGCSPNA (NA to SP Mapping Managed Object - see Section 7.5.9 on page 7-258)	SPID	-	integer	Ø - 7	ADD MODIFY DELETE DISPLAY
	NAID	-	integer	Ø - Øx7fffffff	
	OPERST	ACT INACT	alphanumeric characters	-	

1 **Note**: When applicable, default values are shown in italics. Values that ARE 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'.

3 **Note**: An Internet address in dot notation has one to four numbers, and this must be in decimal format. It represents a 32 bit address, where each leading number is eight bits of the address (high byte first) and the last number is the rest. The following is an example of the decimal format: 146.169.22.42.

7.2 SCCPMMLCommands

7.2.1 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 Signaling 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	Signaling 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.

EXAMPLES

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.
```

7.2.2 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 •OF 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,**GT I E** =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,**GT I E** =4,***TRTYPE***=253, ***ADDRINFO***=8001234567;
DISPLAY-GT:IO=INCOMING,**GT I E** =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.
```

7.2.3 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 Signaling 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***=GT 1***,ENTRYTYPE***=SECONDARY***,***
XLATE_ID=xlate1***,SPC***=1-1-2***,SSN***=254***;***
DELETE-GTENTRY:IO=INCOMING***,GT***=GT 1***;***
MODIFY-GTENTRY:IO=INCOMING***,GT***=GT 1***,ENTRYTYPE***=SECONDARY***,***
XLATE_ID=xlate1***,SPC***=1-2-3***;***
DISPLAY-GTENTRY:IO=INCOMING***,GT***=GT 1***;***

7.2.4 Mate (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 Signaling Point Code (SPC) from the SCCP network database.

DISPLAY-MATE:SPC=spc**,*SSN***=ssn***;*

PARAMETERS

spc

Signaling 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 signaling 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.
```

7.2.5 SCCP (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

VARIANT Variant of the SCCP

- •NONE
- • TT
- •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
```

Table 7-6: SCCP Display Values

SPNO
Signaling Point number that is an integer from Ø to 7.

7.2.6 SCCP Signaling Point (SNSP)

NAME

SNSP

Adds, deletes, or displays a Signaling Point in the SCCP network.

COMMANDS

ADD

Adds a new signaling 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 Signaling 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 Signaling 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

Signaling 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.

EXAMPLES

	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 7-7: SNSP Display Values

SPC	STATUS	XLATE	CONCERNED	SUBSYSTEMS
See description in ~~Section 9.2.1 on page 9-2~~	Status of the signaling point: ACCESSIBLE INACCESSIBLE	PRIMARY SECONDARY	Whether the signaling point is a concerned point code.	Whether subsystems are provisioned.

RL	RSL	CLS
SCCP restriction level	SCCP restriction sub-level	SCCP congestion level

7.2.7 Subsystem (SUBSYS)

NAME

SUBSYS

Adds, deletes, or displays a subsystem or subsystem information for a Signaling 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

Signaling 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 7-8: 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 signaling point is a concerned point code.

7.2.8 Local Subsystem (LOCALSUBSYS)

NAME

LOCALSUBSYS Displays a local subsystem or local subsystem information for a Signaling 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 7-9: 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 signaling point is a concerned point code.

7.2.9 Connection (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.
```

Table 7-10: CONNECTION Display Values

STATUS
The status of the connection is 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

7.3 ISUPMMLCommands

7.3.1 ISUP Circuits (ISUPCCT)

NAME

ISUPCCT

Adds, modifies, deletes or displays 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***];***

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***;***

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***;***

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 point code 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 **Ø** to **3Ø39.**

cctnum

The circuit number. It is a number between Ø 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.

EXAMPLES

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 7-11: ISUP Circuit Display Report

PCNO	DPC	GRPID (group ID)	CCTNUM (circuit No.)	STATUS (circuit status)	MNTCSTATUS (maintenance)	HWDSTATUS (Hardware)	SUSSTATUS (Suspend)
1	5-100-5	1	2	NO-IND	UN-BLK	UN-BLK	NOT-SUS

Table 7-11: ISUP Circuit Display Report

PCNO	DPC	GRPID (group ID)	CCTNUM (circuit No.)	STATUS (circuit status)	MNTCSTATUS (maintenance)	HWDSTATUS (Hardware)	SUSSTATUS (Suspend)
1	5-100-5	1	5	IN-BUSY	L-BLK	L-BLK	ORG-SUS
1	5-100-5	1	10	IDLE	LR-BLK	LR-BLK	NOT-SUS

Important*: The HWDSTATUS column does not print for ANSI variants because it is not not meaningful for ANSI variants*

Table 7-12: ISUP Circuit Display Values

Circuit Status	Maintenance/Hardware Status	Suspend Status

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.

Maintenance status is related to a voluntary setting. Hardware status is related to an automatic setting due to hardware failure.		

7.3.2 ISUP Circuit Group (ISUPCGRP)

NAME

ISUPCGRP

Adds, modifies, deletes or displays 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***];***

MODIFY

Modifies a circuit group in the ISUP database.

MODIFY-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***;***

PARAMETERS

pcno

The unique point code 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 **Ø** to **3Ø49**.

cctnum

Number of circuits that will be in the ISUP circuit group. It is an unsigned integer between **Ø** 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 **Ø** 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.

EXAMPLES

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 7-13: ISUP Circuit Group Display Report– Pre-Call Ctrl & 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 only for ANSI variants.*

Table 7-14: ISUP Circuit Group Display Report – Post-Call Ctrl & 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.*

7.3.3 ISUP Signaling Node (ISUPNODE)

NAME

ISUPNODE

Adds, modifies, deletes, or displays 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***]***
[,CICCONTROL=ciccontrol***] [,LOCATION***=location***]***
[,MAXCCT=maxcct***][,FIRSTCIC***=firstcic***];***

MODIFY

Modifies a signalling point code in the ISUP database.

MODIFY-ISUPNODE:PCNO=pcno***[,DPC***=dpc***]***
[,ANMOFF=anoff***][,ACMOFF***=acmoff***][,CRGOFF***=crgoff***]***
[,CICCONTROL=ciccontrol***];***

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.

DISPLAY-ISUPNODE:PCNO=pcno***;***

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 Ø to 2Ø47.

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 (see Note)

acmoff

Address Complete Office (see Note)

crgoff

Charge Office (see Note)

Note: *The value of office information, i.e., anmoff, acmoff, crgoff, 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

Location 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 **Ø** to **65535**.

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.

EXAMPLES

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:;

PCNODPCCONG STATUSACCESS STATUSANMOFFACMOFFCRGOFFCICCONTROL

1 5-100-50ACCESSIBLEONOFFOFFEVEN

2 6-111-60INACCESSSIBLEOFFONOFFODD

3 2-200-21CONGESTEDONOFFONDEFAULT

<SUCCESS>:: 3 records found.
```

7.3.4 ISUP Configuration (ISUP)

NAME

ISUP

Modifies or displays the current ISUP configuration.

COMMANDS

MODIFY

Modifies the current ISUP configuration.

MODIFY-ISUP:CFGNAME=cfgname***[,VARIANT***=variant***]***
[,MNTCIND=mntcind***][,CONGES***=conges***]***
[,RECMODE=recmode***][,AUTORESP***=autoresp***]***
[,EXCHODC=exchodc***][,UPMIND***=upmind***];***

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. CFØ, CF1.

DISPLAY-ISUP:[CFGNAME=cfgname***];***

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: **GENERIC**, **ANSI92**, **ANSI96**, **BELL**, **DSC**, and **MCI**

Note: *Syntax for BELL MML requires the following:*

If the value is...	**Then,**
	type it directly
	precede the value with an O’
	precede the value with an H’

• For ITU: GENERIC, 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 running 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*)
- OF

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.

EXAMPLES

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

ANSI

```
MML_TH>dis-isup::
CFGNAMEVARIANTMNTCINDRECMODEMBGINDAUTORESP
CF0BELLONRESCALLOFFON
<SUCCESS>:: 1 record found
```

ITU

```
MML_TH>dis-isup::
CFGNAMEVARIANTMNTCINDCONGESTIONRECMODE
CF0TURKEYOFFONRELCALL
<SUCCESS>:: 1 record found
```

7.3.5 ISUP Timer (ISUPTMR)

NAME

ISUPTMR	Modifies or displays ISUP protocol timer values.

COMMANDS

MODIFY Modifies the protocol timer values in ISUP database.

MODIFY-ISUPTMR:TIMERID=tmrid***,VALUE***=value***;***

DISPLAY Displays the ISUP protocol timer values.

DISPLAY-ISUPTMR:TIMERID=tmrid***;***

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 are defined in Table 7-15 on page 7-234.
value	Timer value in milliseconds. Timer value can be between **1Ø** 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.

EXAMPLES

DISPLAY-ISUPTMR:TIMERID=2***;***
DISPLAY-ISUPTMR:TIMERID=****;***
MODIFY-ISUPTMR:TIMERID=2***,VALUE***=20000***;***

SAMPLE OUTPUT

```
TIMERIDVALUE
1  20000
2  15000
.
.
.
n  50000
```

Table 7-15: 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
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	-
WB: White Book				

Table 7-15: ISUP Timers (Continued)

Timer ID	Description	ANSI/ITU Usage	Default in ITU (msec)	Default in ANSI (msec)
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
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 9-32 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 7-16: ISUP Related Timers to Modify

First Timer 15 sec.	Second Timer 1 min.	Related Messages
T12	T13	BLO sent - BLA expected
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

7.4 System MML Commands

7.4.1 Host (HOST)

NAME

HOST	Adds, deletes, displays, or modifies a host instance.

COMMANDS

ADD Adds a new host instance.

ADD-HOST:HOSTNAME=hostname***,RMTHOST***=rmthost ***[,ALIAS***=alias***][,RMTHOSTTYP***=rmthosttyp***][,CONF***=conf***];***

MODIFY Modifies a host instance information.

MODIFY-HOST:HOSTNAME=hostname***,RMTHOST***=rmthost ***[,RMTHOSTTYP***=rmthosttyp***][,CONF***=conf***];***

DELETE Deletes a host instance.

DELETE-HOST:HOSTNAME=hostname***][,RMTHOST***=rmthost***];***

DISPLAY Displays a specific host instance or all instances.

DISPLAY-HOST:[HOSTNAME=hostname***][,RMTHOST***=rmthost***];***

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. It is a string of 1 to 15 alphanumeric characters maximum.
rmthost	Name of remote host. It is a string of 1 to 15 alphanumeric characters maximum.
alias	Alias name for the remote system, if it is a multi-homed host.
rmthosttyp	Remote system host type. It is a character string that accepts the following values: • AMGR: Distributed7-type system *(default)* • OTHER: other than Distributed7-type system

conf	Determines the configuration of the host. It is a character string that accepts the following values: • ON Establish this connection. • OFF Wait for more 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 parameter of the entry is ON

EXAMPLES

ADD-HOST:HOSTNAME=tweety-priv***;RMTHOST***=sylvester-priv***,CONF***=OFF***;***
MODIFY-HOST:HOSTNAME=tweety-priv,***CONF***=ON***;***
DELETE-HOST:HOSTNAME=tweety-priv***;***
DISPLAY-HOST:;

SAMPLE OUTPUT

```
MML_TH>DISPLAY-HOST:;
-------------------------------------------------------------------
         HOSTNAME          RMTHOST            ALIAS RMTHOSTTYP CONF
-------------------------------------------------------------------
      tweety-priv   sylvester-priv                -       AMGR   ON
   sylvester-priv      tweety-priv                -       AMGR   ON
<SUCCESS>:: 2 records found
```

7.4.2 Network (NTWK)

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 Signaling Gateway Client 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. It is a string of 1 to 15 alphanumeric characters maximum.

mode Specifies how the host operates in the network. It is a character string that accepts the following values:

- STNDLN stand alone mode
- DSTRBTDdistributed mode *(default)*

clocksync Specifies whether or not the ability to synchronize the network clock is available. It is a character string that accepts the following values:

- ON *(default)*
- OF

frequency Specifies, in milliseconds, how often to check the system clock on all hosts if CLOCKSYNC is **ON**. It is an integer from Ø to 1ØØØØ. The default value is Ø if running in the stand alone mode, or 1ØØØ in the distributed mode. THe range for the distributed mode is 6Ø to 1ØØØØ.

dualhost Specifies the alternate host name, if any, of the local host on a secondary host in a dual-LAN network. If dual-LAN is not in use, then this field must contain the local host name specified in the HOSTNAME parameter. It is a string of 1 to 15 alphanumeric characters maximum.

netmask1 Specifies the 32-bit mask, in hex format, that is used to extract the network ID on the primary network. The following default values are initialized on the basis of class type associated with the corresponding network:

- 7fØØØØØØ *(Class A default)*
- 3fffØØØØ *(Class B default)*

	• 1fffffØØ *(Class C default)*
netmask2	Specifies the 32-bit mask, in hex format, that is used to extract the network ID on the secondary network, if any. The following default values are initialized on the basis of class type associated with the corresponding network: • 7fØØØØØØ *(Class A default)* • 3fffØØØØ *(Class B default)* • 1fffffØØ *(Class C default)*

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

MODIFY-NTWK:HOSTANME=vortex***,MODE***=STNDLN***;***
DISPLAY-NTWK:HOSTNAME=vortex***;***
DISPLAY-NTWK:;

SAMPLE OUTPUT

```
MML_TH> display-ntwk:;
------------------------------------------------------------------------
HOSTNAMEMODECLOCKSYNCFREQUENCYDUALHOSTNETMASK1NETMASK2
------------------------------------------------------------------------
sylvester-pDSTRBTDON1000sylvester-p7f0000007f000000
tweety-pDSTRBTDON1000tweety-p7f0000007f000000
<SUCCESS>:: 2 records found
```

7.5 Signaling Gateway Client MML Commands

7.5.1 Application Server (SGCAS)

NAME

SGCAS	Configures each Application Server in the system, including the traffic mode, routing key index and routing context associated with the AS.

COMMANDS

ADD Adds an AS configuration for a particular Signalling Point (SP).

ADD-SGCAS:ASID=asid***,SPID***=spid***,RKID***=rkid,***RCID***=rcid***,***
[MODE=mode***];***

MODIFY Modifies the configuration of an existing AS.

MODIFY-SGCAS:ASID=asid***[,SPID***=spid***][,RKID***=rkid***][,RC***=rc***]***
[MODE=mode***];***

DELETE Removes an AS configuration, which can only be done when the AS is inactive.

DELETE-SGCAS:ASID=asid***;***

Caution: *All its associated SGCASTFC entries are also deleted when an AS is deleted!*

DISPLAY Displays the configuration of the AS.

DISPLAY-SGCAS:ASID=asid***;***

PARAMETERS

asid AS identification name. It is string that accepts up to 15 characters maximum.

spid Signaling Point ID of an AS. The SPID assigned to each AS must be unique because each SP serves only one AS. It is an unsigned integer from Ø to 7.

rkid The Route Key ID that this AS is serving, which must have been created before the AS is configured. It is a string of up to 15 characters maximum.

rcid The Route Context associated with the route key. It is a unique value assigned to represent the route key or traffic range served by this AS, and it must be identical to the one provisioned on the SG side. This is required to route messages correctly. It is an integer from Ø to 0x7fffffff.

mode The traffic mode of the ASPs in the specified AS. It is a character string that accepts the following value:

- **LOADSHARE** - The ASPs operate in the *loadsharing* mode, distributing the traffic between the configured ASPs.

ERRORS

Database operation failed
Configuration limit exceeded
Record already exists
Record does not exist
Invalid key
Missing key
Missing mandatory parameters
Error configuring M3UA stack
AS does not exist
SP is used by other AS
NA for the specified SP does not exist
Remote SGP does not exist
Route key is used by other AS

EXAMPLES

ADD-SGCAS:ASID=**1**,RKID=**r1,RCID=1;***
MOD-SGCAS:ASID=**1**,RKID=**r1**,RCID=**1ØØ**;
DELETE-SGCAS:ASID:ASID=**1**;
DISPLAY-SGCAS:ASID=**1**;
DISPLAY-SGCAS:ASID=**1**;
DISPLAY-SGCAS:;

SAMPLE OUTPUT

```
MML_TH>dis-sgcas:;
---------------------------------------------------------------
             ASID SPID             RKID        RCID       MODE
---------------------------------------------------------------
              as1    0               r1           1   LOADSHARE
<SUCCESS>:: 1 record found
```

7.5.2 Application Server Process (SGCASP)

NAME

SGCASP — Configures an ASP, which serves one or more ASs through one or more Stream Control Transmission Protocol (SCTP) associations.

COMMANDS

MODIFY — Changes the configuration of an ASP.

MODIFY-SGCASP:ASPID=aspid***[,IP1***=ip1***][,IP2***=ip2***][,HOSTNAME***=host name***] [,SCTPPORT***=sctpport***][,SGREDMODE***=sgredmode***] [,NWASPID***=nwaspid***];***

DISPLAY — Displays the configuration of an ASP.

DISPLAY-SGCASP:[ASPID=aspid***];***

PARAMETERS

aspid — The identification of the Application Server Process (ASP). It is a string of up to 15 characters or an * to display all ASPs. The preassigned value is the hostname where the ASP is running.

ip1 — Primary IP address used by the ASP to communicate with the SGPs. It is a string of up to 15 characters maximum entered in the dot notation, decimal format. A '-' can be entered to de-assign a previously assigned address.

ip2 — Secondary IP address used by the ASP to communicate with the SGPs. It is a string of up to 15 characters maximum entered in the dot notation, decimal format. A '-' can be entered to de-assign a previously assigned address.

hostname — Domain name of the system where the local ASP runs. This field can be used in place of the IP addresses if Internet Domain Name System (DNS) is available. Input is a string of 128 characters. A '-' can be entered to de-assign a previously assigned address.

sctpport — Local port number to which SCTP is bound. It is an integer from 1ØØØ to 99999. The default value is 29Ø5.

sgredmode — Redundancy mode between two or more SGs that are connected to the ASP. It is a character string that accepts the following values:

- **OVERRIDE** - This connection is used to send messages for as long as the connection to the primary SG is up. The primary SG has the smallest SG ID and an active connection to the ASP.
- **LOADSHARE** - The load of messages sent is shared by distributing them across all SGs with active connections. Loadsharing is based on the Signaling Link Selection (SLS) value.

Default value is LOADSHARE.

nwaspid The Network ASP ID. It a unique value that identifies an ASP in an AS. It accepts the following values Ø to Øx7fffffff *(default)*.

Note*: The IP address and hostname fields are mutually exclusive, i.e. only either IP addresses OR hostname can be specified at the same time, but not both.*

ERRORS

Database operation failed

Record does not exist

Invalid key

Missing key

Missing mandatory parameters

Invalid ASP ID

Invalid IP address

IP addresses cannot be identical

Error configuring M3UA stack

EXAMPLES

MOD-SGCASP:ASPID=chip,IP1=155.226.145.163,SCTPPORT=3000
DISPLAY-SGCASP:;

SAMPLE OUTPUT

```
MML_TH>dis-sgcasp:;
-----------------------------------------------------------------------------
ASPID: chip
HOSTNAME:
IP1: 155.226.145.163      IP2:      SCTPPORT: 2905   SGREDMODE: OVERRIDE
            NWASPID: 2147483647


<SUCCESS>:: 1 record found
```

7.5.3 AS-ASP Traffic Control (SGCASTFC)

NAME

SGCASTFC This managed object defines traffic control for an AS and ASP.

COMMANDS

ADD

Adds a traffic control definition for an AS and an ASP.

ADD-SGCASTFC:ASID=asid,DESTPID=destid[,ORIGPID=origpid][,TYPE=type];

MODIFY

Changes a traffic control definition for an AS and an ASP.

MODIFY-SGCASTFC:ASID=asid,DESTPID=destid,OPERST=operst[,ORIGPID=origpid];

DELETE

Deletes a traffic control definition for an AS and an ASP.

DELETE-SGCASTFC:ASID=asid,DESTPID=destid[,ORIGPID=origpid];

DISPLAY

Displays a traffic control definition for an AS and an ASP.

DISPLAY-SGCASTFC:[ASID=asid][,DESTPID=destid][,ORIGPID=origpid][,TYPE=type];

PARAMETERS

asid

AS identification name. It is string that accepts up to 15 characters maximum.

destpid

The remote process identification name. This should be a valid SGCSGP or SGCIPSP identifier. It is string that accepts up to 15 characters maximum.

origpid

The local process identification name. It is string that accepts up to 15 characters maximum.

type

Type of the peer target. Assumes one of the following:

• **SGP**	MO shows SGP traffic control information (default).
• **IPSP**	MO shows IPSP traffic control information.

ERRORS

Database operation failed

Configuration limit exceeded

Record already exists

Record does not exist

Invalid key

Missing key

Missing mandatory parameters

Error configuring M3UA stack

AS is active

AS traffic already defined

AS traffic not defined

EXAMPLES

ADD-SGCASTFC:ASID=as1***,DESTPID=sylvester,ORIGPID=vortex;***
MODIFY-SGCASTFC:ASID=as1***,DESTPID=tweety,ORIGPID=vortex,***
OPERST=INACT***;***
DELETE-SGCASTFC:ASID=as1***,DESTPID=tweety,ORIGPID=vortex;***
DISPLAY-SGCASTFC:ASID=AS1***,DESTPID=***tweety***;***
DISPLAY-SGCASTFC:ASID=as1***,ORIGPID=***vortex***;***
DISPLAY-SGCASTFC:;

SAMPLE OUTPUT

```
MML_TH>disp-sgcastfc:;

-----------------------------------------------------------------------

ASID DESTPID ORIGPID OPERST STATUS TYPE

-----------------------------------------------------------------------

AS SGP-C IPSP-AACTACTSGP

IPASIPSP-B IPSP-A ACT ACT IPSP
```

Table 7-17: SGCASTFC Display Values

STATUS
• ACT - AS is active • INACT - AS is inactive

7.5.4 Destination Point Code (SGCDPC)

NAME

SGCDPC	This managed object defines remote SS7 destination point code that is reachable through a particular SG in a particular network appearance.

COMMANDS

ADD Adds a new point code that an SG can reach.

ADD-SGCDPC:SGID=*sgid****,DPC=****dpc****,NAID=****naid****[,PRIORITY=****priority****];***

MODIFY Modifies a point code that an SG can reach.

MODIFY-SGCDPC:SGID=*sgid****,DPC=****dpc****,NAID=****naid****,PRIORITY=****priority****;***

DELETE Deletes a point code that an SG can reach. At least one of the optional parameters must be entered.

DELETE-SGCDPC:SGID=*sgid****,DPC=****dpc****,NAID=****naid****;***

DISPLAY Displays the point code that an SG can reach.

DISPLAY-SGCRK:[SGID=*sgid****][,DPC=****dpc****][,NAID=****naid****][,PRIORITY=****priority****];***

PARAMETERS

sgid	The identification of the SG that can reach the specified point code. It is a string of up to 15 characters.
dpc	The point code that the given SG can reach. It is entered in the *x-y-z* format, up to a maximum of 11 characters. See page 7-188 for more information about this point code format.
naid	Network Appearance ID of the specified point code. Valid values are 0-0xFFFFFFFF.

Note*: When NAID=4294967295 (0xFFFFFFFF), NA field is not sent.*

aspid	The identification of the ASP. It is a string of up to 15 characters. This parameter is used to get the status of the remote point code as seen from the specified ASP.

ERRORS

Database operation failed

Configuration limit exceeded

Record already exists

Record does not exist

Invalid key

Missing key

Missing mandatory parameters

Invalid PC format

NA does not exist

DPC exists for SP %d

Error configuring M3UA stack

EXAMPLES

ADD-SGCDPC:SGID=sg***,DPC***=8-8-8***,NAID***=Ø***;***
DELETE-SGCDPC:SGID=sg***,DPC***=8-8-8***,NAID***=Ø***;***
DISPLAY-SGCDPC:SGID=sg***;***
DISPLAY-SGCDPC:NAID=Ø***;***
DISPLAY-SGCDPC:;

SAMPLE OUTPUT

```
MML_TH>dis-sgcdpc:;
-------------------------------------------------------------------
            DPC        NAID             SGID   PCST            ASPID
--------------------------------------------------------------------      1-
            10-1         100              sg1    ACC              chip
<SUCCESS>:: 1 record found
```

7.5.5 Routing Key (SGCRK)

NAME

SGCRK

Defines the route key associated with an AS.

COMMANDS

ADD

Adds a route key.

ADD-SGCRK:RKID=rkid***,TYPE***=type***,DPC***=dpc***;***

MODIFY

Modifies the definition of a route key.

MODIFY-SGCRK:RKID=rkid***[,DPC***=dpc***];***

DELETE

Deletes a route key. A route key can be deleted only if it is not assigned to any AS.

DELETE-SGCRK:RKID=rkid***;***

DISPLAY

Displays the parameters of a route key.

DISPLAY-SGCRK:[RKID=rkid***];***

PARAMETERS

rkid

The identification of a route key. It is a string of up to 15 characters.

type

Route key combination type. It is a character string that accepts the following values:

- **DPC** Routing key that is only a DPC *(default)*
- **DPC_OPC**Combination of DPC and OPC
- **DPC_OPC_CIC**Combination of DPC, OPC and CIC
- **DPC_SIO** Combination of DPC and SIO
- **DPC_OPC_SIO**Combination of DPC, OPC and SIO
- **DPC_CIC_SIO**Combination of DPC, CIC and SIO

dpc

Destination Point Code. It is entered in the *x-y-z* format, up to a maximum of 11 characters. See page 7-188 for more information about this point code format.

ERRORS

Database operation failed

Configuration limit exceeded

Record already exists

Record does not exist

Invalid key

Missing key

Missing mandatory parameters

EXAMPLES

ADD-SGCRK:RKID=r1***,TYPE***=DPC***,DPC***=1-1-1***;***
ADD-SGCRK:RKID=r2***,TYPE***=DPC***,DPC***=1-1-2***;***
ADD-SGCRK:RKID=r3***,TYPE***=DPC_OPC_CIC***,DPC***=2-2-2***;***
MODIFY-SGCRK:RKID=r1***,TYPE***=DPC_OPC***;***
MOD-SGCRK:RKID=r1***,DPC***=3-2-1***;***
DELETE-SGCRK:RKID=r3***;***
DISPLAY-SGCRK:RKID=r1***;***
DIS-SGCRK:;

SAMPLE OUTPUT

```
MML_TH>DISPLAY-SGCRK:;

-------------------------------------------

             RKID         TYPE          DPC

-------------------------------------------

               r1          DPC     11-11-11

               xx  DPC_OPC_CIC        9-9-1

<SUCCESS>:: 2 record found
```

7.5.6 Routing Key Range (SGCRKRNG)

NAME

SGCRKRNG Defines a range that is associated with an existing route key (Routing Key (SGCRK) on page 7-249) that has any TYPE except DPC. One range at a time can be added or deleted.

COMMANDS

ADD

Adds a range to an existing route key.

ADD-SGCRKRNG:RKID=rkid***[,SI***=si***][,OPC***=opc***][,CICMIN***=cicmin***] [,CICMAX***=cicmax***][,SSN***=ssn***];***

DELETE

Deletes a route key range. Only route keys that are not associated with any AS can be deleted.

DELETE-SGCRKRNG:RKID=rkid***[,SI***=si***][,OPC***=opc***] [,CICMIN***=cicmin***][,CICMAX***=cicmax***][,SSN***=ssn***];***

DISPLAY

Displays the parameters of a route key range.

DISPLAY-SGCRKRNG:[RKID=rkid***];***

PARAMETERS

rkid

The identification of an existing route key. It is a string of up to 15 characters.

si

Service Indicator. It is a character string that accepts the following values:

- **SCCP**
- **SUP**
- **TUP**

opc

Origination Point Code of the originating SS7 node that is connected to the SG. It is entered in the x-y-z format. See page 7-188 for more information about this point code format.

cicmin

The minimum number in a CIC range. It is a string of Ø to 65535 characters maximum.

cicmax

The maximum number in a CIC range. It is a string of Ø to 65535 characters maximum.

ssn

Subsystem Number. It is an integer from 2 to 255.

ERRORS

Database operation failed

Configuration limit exceeded

Record already exists

Record does not exist

Invalid key

Missing key

Missing mandatory parameters

Route key range already exists

Route key range does not exist

Route key does not exist

CIC range overlaps with other ranges in this RK

CIC range error (CICMAX must be >= CICMIN)

RKRNG not applicable for RK of DPC type

EXAMPLES

ADD-SGCRKRNG:RKID=4***,OPC***=9-9-9***,SI***=ISUP***;***
DELETE-SGCRKRNG:RKID=4***;***
DISPLAY-SGCRKRNG:RKID=Ø***;***

SAMPLE OUTPUT

```
MML_TH>DIS-SGCRKRNG:;
-----------------------------------------------------------------
             RKID    SI          OPC      CICMIN      CICMAX SSN
-----------------------------------------------------------------
               xx     -        9-9-9        1000        2000   -
<SUCCESS>:: 1 records found
```

7.5.7 Signaling Gateway (SGCSG)

NAME

SGCSG — Configures remote Signaling Gateways.

COMMANDS

ADD

Adds a remote signaling gateway configuration.

ADD-SGCSG:SGID=sgid***,[MODE***=mode***];***

MODIFY

Modifies the configration of a remote signaling gateway.

MODIFY-SGCSG:SGID=sgid***,[MODE***=mode***];***

DELETE

Deletes a configuration of a remote signaling gateway process (SGP).

DELETE-SGCSG:SGID=sgid***;***

Caution: *A remote SG can only be deleted when it is not associated with any remote SGPs.*

DISPLAY

Displays the configuration of a remote SG.

DISPLAY-SGCSG:[SGID=sgid]***;***

PARAMETERS

sgid

Identification of the signaling gateway. It is a string of up to 15 characters maximum.

mode

Traffic modes in which the remote SGPs operate in this SG. It is a character string that accepts the following values:

- **OVERRIDE**SGPs operate in override mode, where there is only one primary SGP actively processing traffic while the rest are in standby mode.
- **LOADSHARE**SGPs operate in loadshare mode, where the traffic is distributed across the active SGPs.

Default value is LOADSHARE.

ERRORS

Database operation failed
Configuration limit exceeded
Record already exists
Record does not exist
Invalid key
Missing key
Missing mandatory parameters
Error configuring M3UA stack

SG does not exist

SGP exists in SG

DPC exists in SG

EXAMPLES

ADD-SGCSG:SGID=sg;
DELETE-SGCSG:SGID=sg***;***
DISPLAY-SGCSG:SGID=sg***;***
DISPLAY-SGCSG:;

SAMPLE OUTPUT

```
MML_TH> DISPLAY-SGCSG:;

----------------------------

             SGID       MODE

----------------------------

               sg  LOADSHARE

<SUCCESS>:: 1 record found
```

7.5.8 Signaling Gateway Process (SGCSGP)

NAME

SGCSGP

Configures a remote Signaling Gateway Process (SGP).

COMMANDS

ADD

Adds a configuration for a new remote signaling gateway process.

ADD-SGCSGP:SGPID=sgpid***,SGID***=sgid***[,IP1***=ip1***][,IP2***=ip2***][,HOSTNAME***=hostname***][,SCTPPORT***=sctpport***];***

MODIFY

Changes the configuration of a remote signaling gateway process.

MODIFY-SGCSGP:SGPID=sgpid***[,SGID***=sgid***][,IP1***=ip1***][,IP2***=ip2***][,HOSTNAME***=hostname***] [,SCTPPORT***=sctpport***][OPERST***=operst***][ASPID***=aspid***];***

DELETE

Deletes the configuration of a remote signaling gateway process.

DELETE-SGCSGP:SGPID=sgpid***;***

Caution: *A remote SGP can only be deleted when it is **NOT** connected to an ASP.*

DISPLAY

Displays the configuration of a remote signaling gateway process.

DISPLAY-SGCSGP:[SGPID=sgpid***][ASPID***=aspid***];***

PARAMETERS

sgid	Identification of the remote signaling gateway. It is a string of up to 15 characters maximum.
sgpid	Identification of the remote signaling gateway process. It is a string of up to 15 characters maximum.
ip1	Primary IP address used by the SGP to communicate over SCTP. It is a string of up to 15 characters maximum, entered in the dot notation, decimal format.
ip2	Secondary IP address used by the SGP to communicate over SCTP. It is a string of up to 15 characters maximum, entered in the dot notation, decimal format.
hostname	Domain name of the system where the remote SGP runs. This field can be used in place of the IP addresses if Internet Domain Name System (DNS) is available. Input is a string of 128 characters.
sctpport	Remote port number to which SCTP is bound. Default value is 2905.
operst	Operation performed on a remote SGP from the specified ASP. It is a string that accepts the following value:

- **DISCONN**Disconnects an SCTP association between an ASP and a remote SGP.
- **CONN** Establishes an SCTP association between an ASP and a remote SGP.
- **UP** Activates an ASP so that it is available to all application servers it serves for an SGP.
- **DOWN** Deactivate an ASP so that it is not available to all application servers it serves for an SGP. This does not affect the SCTP association.

aspid The identification of the ASP. It is a string of up to 15 characters. This field is applicable only when OPERST is specified. It indicates the ASP in which the OPERST is performed. If not specified, the OPERS T is applied to all ASPs.

ERRORS

Database operation failed

Configuration limit exceeded

Record already exists

Record does not exist

Invalid key

Missing key

Missing mandatory parameters

Invalid IP address

IP addresses cannot be identical

Invalid operation state request

Error configuring M3UA stack

Connection is already down

Connection exists or is being established

ASP is up

Cannot delete SGP when AS exists

EXAMPLES

ADD-SGCSGP:SGID=SG1***,IP1***=155.226.147.189***,IP2***=155.226.147.189***;***
ADD-SGCSGP:SGID=SG1***,IP1***=155.226.147.4***,IP2***=155.226.147.4***, SCTP-PORT***=29Ø6***;***
MODIFY-SGCSGP:SGPID=tweety***,IP1***=155.226.147.4***,SCTPPORT***=29Ø5***;***
DELETE-SGCSGP:SGPID=tweety***;***
DISPLAY-SGCSGP:;

SAMPLE OUTPUT

```
MML_TH>dis-sgcsgp:;
--------------------------------------------------------------------------
```

```
SGPID: sylvester      SGID: sg1
HOSTNAME:
IP1: 155.226.145.142   IP2:   SCTPPORT: 2905   ASSOCID: 1   OPERST: UP   STATUS: UP
ASPID: chip
SGPID: tweety      SGID: sg1
HOSTNAME:
IP1: 155.226.145.143   IP2:   SCTPPORT: 2905   ASSOCID: 2   OPERST: UP   STATUS: UP
ASPID: chip

<SUCCESS>:: 2 records found
```

7.5.9 Signaling Point to Network Appearance Mapping (SGCSPNA)

NAME

SGCSPNA

This managed object defines a mapping between a Network Appearance (NA) to a Signaling Point (SP). Multiple SPs can share the same NA.

COMMANDS

ADD

Adds a new mapping between a Network Appearance and a Signaling Point.

ADD-SGCSPNA:SPID=spid***,NAID***=naid***;***

MODIFY

Modifies the Network Appearance and Signaling Point mapping or registration status of a particular Signaling Point.

MODIFY-SGCSPNA:SPID=spid***[,NAID***=naid***][,OPERST***=operst***] [,ASPID***=aspid***];***

DELETE

Deletes a mapping between a Network Appearance and a Signaling Point.

DELETE-SGCSPNA:SPID=spid***;***

Caution: *Before an SGCSPNA is deleted, the NA status **must be** INACT. Also, there must not be any AS configured for the associated SP.*

DISPLAY

Displays the Network Appearance and Signaling Point mapping and registration status of an SP.

DISPLAYSG-SGCSPNA:[SPID=spid***][,ASPID***=aspid***];***

PARAMETERS

spid

Signaling Point identification. It is an unsigned integer from 0 to 7.

naid

Network Appearance identification. Valid values are 0-0xFFFFFFFF. Multiple SPs may share the same NAID. This field can only be modified when the NA status is INACT.

Note*: When NAID=4294967295 (0xFFFFFFFF), NA field is not sent.*

operst

The operation state that is performed on the SP. It is a character string that accepts the following values:

- **ACT** Activate an SP by registering the ASP to MTP as a gateway process for the specified SP.
- **INACT** Inactivate an SP by removing the ASP's registration to MTP for the specified SP. *(default)*

aspid	The identification of the ASP. It is a string of up to 15 characters. This field is optional: • The OPERST is applied to the specified ASP if it is specified • The OPERST is applied to all ASPs if it is NOT specified

ERRORS

Database operation failed
Configuration limit exceeded
Record already exists
Record does not exist
Invalid key
Missing key
Missing mandatory parameters
Failed to establish communication with UPMD
SP is active
SP is inactive
MTP not ready or not configured
Failed to query MTP (check upmd status)
Protocol mismatch between SPs with same NA
NA is used in other SP
AS exists for SP %d
SP activation in progress, please wait

EXAMPLES

ADD-SGCSPNA:SPID=Ø***,NAID***=Øxffffffff***;***
ADD-SGCSPNA:SPID=1***,NAID***=1ØØ***;***
MOD-SGCSPNA:SPID=Ø***,NAID***=Øxffffffff***,OPERST***=ACT***;***
MODIFY-SGCSPNA:SPID=1***,NAID***=Øxffffffff***,OPERST***=ACT***;***
DELETE-SGCSPNA:SPID=1***;***
DISPLAY-SGCSPNA:SPID=Ø***;***
DIS-SGCSPNA:;

SAMPLE OUTPUT

```
MML_TH>DISPLAY-SGCSPNA:;
-----------------------------------------------
 SPID        NAID OPERST STATUS            ASPID
-----------------------------------------------
    0           0     ACT    ACT           vortex
    0           0     ACT    ACT           atomic
<SUCCESS>:: 1 record found
```

Table 7-18: SGCSPNA Display Values

SGPID	STATUS
SGP identification	SP registration Status

7.5.10 IP Server Process

NAME

SGCIPSP Configures an IP Application Server Process (IPSP).

COMMANDS

ADD

Adds an IPSP.

ADD-SGCIPSP:IPSPID=ipspid[,IP1=ip1][,IP2=ip2][,HOSTNAME=hostname][,SCTPPORT=sctpport][,NWASPID=nwaspid][,MODE=mode];

MODIFY

Modifies the configuration of an IPSP.

MODIFY-SGCIPSP:IPSPID=ipspid[,IP1=ip1][,IP2=ip2][,HOSTNAME=hostname][,SCTPPORT=sctpport][,NWASPID=nwaspid][,OPERST=operst][,ASPID=aspid] [,MODE=mode];

DELETE

Deletes an existing IPSP.

DELETE-SGCIPSP:SGCIPSPID=ipspid;

DISPLAY

Displays the configuration of one or more IPSPs.

DISPLAY-SGCIPSP:[IPSPID=ipspid][,ASPPID=aspid];

PARAMETERS

Ipspid	The identification of the IP Application Server Process (IPASP).
Ip1	Primary IP address used by the IPSP to communicate with the ASPs
Ip2	Secondary IP address used by the IPSP to communicate with the ASPs.
Hostname	Domain name of the system where the remote IPSP runs.
Sctpport	The remote port number to which SCTP is bound.
Associd	SCTP association ID for the association between the ASP and the IPSP.
Nwaspid	Network ASP ID. It is a unique value that identifies an IPSP within an IPAS.
Operst	The operation performed on a remote IPSP from a specified ASP. This parameter is used only with the MODIFY and DISPLAY commands. **The handling of this parameter, together with the ASPID parameter, has to be performed in a similar manner to the SGCSGP MO. Mimicking the connections and availability scenarios between the ASP and the SGP.** It can assume the following values:

- **DISCONN -** Disconnect the SCTP association between an ASP and a remote IPSP.

- **CONN -** Establishes the SCTP association between an ASP and a remote IPSP (if mode is set to SERVER), or stands ready to accept an association request from the peer-IPSP (if mode is set to CLIENT).
- **UP** - Activates an ASP so that it is available to all application servers it serves for an IPSP.
- **DOWN -** Deactivates an ASP so that it is not available to any application servers it serves for an IPSP. This does not affect the SCTP association.

Mode

Defines the client/server role for the purpose of SCTP association setup. It can assume the following values:

- **SERVER -** Remote IPSP is defined as server, local side should initiate SCTP association setup (default).
- **CLIENT -** Remote IPSP is defined as client, remote side should initiate SCTP association setup.

Aspid

The identification of the ASP forming the local side of IPSP communication. This field is applicable only when OPERST is specified. It is used to indicate the ASP for which the OPERST is applied. If not specified, the OPERST is applied to all ASPs.

Status

This is a display-only parameter, which shows the actual connection status between the IPSP and ASPs and assumes the same values as the OPERST *(i.e. status can be DOWN while the OPERST is UP, if the UP action has not been completed successfully).*

ERRORS

Database operation failed
Licensed number of IPSP reached
Configuration limit exceeded
Record already exists
Record does not exist
Invalid key
Missing key
Missing mandatory parameters
Invalid IP address
IP addresses cannot be identical
Invalid operation state request
Error configuring M3UA stack
Connection is already down
Connection exists or is being established
ASP is down

EXAMPLE

ADD-SGCIPSP:IPSPID=0,IP1=155.226.147.226.MODE=SERVER;

SAMPLE OUTPUT

```
MML_TH>disp-sgcipsp:;
------------------------------------------------------------------
           -------
IPSPID: shenyang      NWASPID: 72      MODE: SERVER
HOSTNAME:       IP1: 10.20.2.72      IP2:                 SCTPPORT: 2905
ASPID: jilin      OPERST: UP      STATUS: UP      ASSOCID: 2
```

7.5.11 IP Application Server (SGCIPAS)

NAME

SGCIPAS Configures a remote IP Application Server (IPAS) for a specific Signaling Point (SP).

COMMANDS

ADD Adds a remote IPAS configuration for a particular Signaling Point.

ADD_SGCIPAS:IPASID=IPASID,SPID=spid,OWN_RKID=own_rkid, RMT_RKID=rmt_rkid,RCID=rcid,IPSPLIST=ipsplist [, MODE=mode][,MINASP=minasp][PDTIMER=pdtimer];

MODIFY Modifies the configuration of an AS.

MODIFY-SGCIPAS:IPASID=IPASID[,SPID=spid][,OWN_RKID=rkid]

[,RMT_RKID=rmt_rkid][,RCID=rcid][,MODE=mode][,IPSPLIST=ipsplist][,PDTIMER=pdtimer];

DELETE Deletes an AS configuration.

DELETE-SGCIPAS:IPASID=IPASID;

✓ ***Note****: An IPAS must have no registered IPSP actively processing traffic BEFORE an IPAS can be deleted.*

DISPLAY Displays the configuration of one or more AS(s).

DISPLAY-SGCIPAS:[IPASID=IPASID];

PARAMETERS

Ipasid	IPAS identification name.
Spid	Signaling Point ID of an IPAS.
Rmt_rkid	The remote Route Key ID that this IPAS is serving. The routing key this parameter identifies defines the type of outgoing traffic, which should be routed using this IPAS. This routing key behaves similar to the SGCIPAS routing key.

Own_rkid	The local Route Key ID that this IPAS is serving. The routing key this parameter identifies defines the type of incoming traffic routed using this IPAS. This routing key behaves similar to the SGCAS routing key.
Rcid	The Route Context associated with both local and remote routing keys. It is a unique value assigned to represent the route key or traffic range served by this AS, and it must be identical to the one provisioned on the peer IPAS. The same rcid is used for both incoming and outgoing traffic for a specific IPAS. Hence for this IPAS, this value is expected to be the same for all incoming/outgoing ASP Active, ASP Active Ack, ASP Inactive, ASP Inactive Ack and DATA messages.
Mode	The traffic mode of the IPSPs for the specified IPAS. The value can be one of the following: • LOADSHARE - IPSPs are operating in a load-sharing mode, and the traffic is distributed to all active IPSPs. • OVERRIDE - IPSPs are operating in an override mode, in which there is only one primary IPSP that is actively processing traffic, and the other IPSPs are in standby mode.
Ipsplist	The list of IPSPs for the specified IPAS.
Minasp	The minimum number of Asps in this AS that must be active to handle load and keep the AS active. This parameter is required only when the MODE parameter is LOADSHARE.
Pdtimer	The pending timer is the amount of time, in seconds, to wait for an ASP to become active before deleting any messages in the pending queue.

Note: The IP address and hostname fields are mutually exclusive, i.e. only IP addresses OR a hostname can be specified at any given time, not both.

ERRORS

Database operation failed
Licensed number of ASP reached
Configuration limit exceeded
Record already exists
Record does not exist
Invalid key
Missing key
Missing mandatory parameters
Invalid IP address
IP addresses cannot be identical
Invalid operation state request
Error configuring M3UA stack
Connection is already down
Connection exists or is being established
ASP is down

EXAMPLE

ADD-SGCIPAS:IPASID=1,OWN_RKID=r1, RMT_RKID=r2, RCID=1, SPID=1, IPSPLIST=IPSP1/IPSP2, MODE=LOADSHARE, MINASP=1;

SAMPLE OUTPUT

```
MML_TH>disp-sgcipas:;

--------------------------------------------------------------------------
            -------

IPASID: IPAS      SPID: 7      OWN_RKID: rk_local      RMT_RKID: rk_remote
            RCID: 7572

MODE: LOADSHARE      MINASP: 1      IPSPLIST: shenyang      PDTIMER: 1
```

Chapter 8: System Processes

8.1 ChapterOverview

This chapter provides descriptions of the Signaling Gateway Client system processes (daemons) and their configuration files. The system processes are summarized in the following table.

Table 8-1: Process Summary

New Command	Description
AccessAlarm (alarmd)	Starts the Alarm handling process
AccessMOB (mob)	Starts the Managed Object Browser graphical user interface
AccessOMAP (omapd)	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
aspd	Starts the Signaling Gateway Client Application Server processes
spmd	Starts the Distributed7 infrastructure
tcmd	Sets up the TCAP multiplexer.
upmd	Starts User Part Multiplexer

8.2 ProcessListing

***Important**: Please refer to Chapter 2: Distributed7 Overview for a list of the user commands with external dependencies to make sure your environment has the necessary software libraries.*

To use 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
```

8.2.1 AccessAlarm (alarmd)

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 or

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.*

Related Information

- Section 8.2.7, **apmd** on page 8-280
- Section 9.2.25, ebs_report

8.2.2 AccessMOB (mob)

NAME

AccessMOB Starts the graphical user interface.

SYNOPSIS

AccessMOB sp

mob sp

DESCRIPTION

AccessMOB or ***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 ***sgc_stop***.

Important*: The* ***upmd*** *process* ***MUST*** *be running before executing this command. Use* ***ebs_ps*** *to confirm that these processes exist.*

Related Information

- Section 8.2.13, **mml** on page 8-292
- Section 8.2.20, **upmd** on page 8-303

8.2.3 AccessOMAP (omapd)

NAME

AccessOMAP or	
omapd	Starts the SS7 OMAP process.

SYNOPSIS

AccessOMAP [-q report_frequency] sp

omap [-q report_frequency] sp

DESCRIPTION

AccessOMAP or

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.

Related Information

- Section 8.2.7, **apmd** on page 8-280

8.2.4 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] -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.

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

8.2.5 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

EXAMPLE

AccessStatus 0

Starts up AccessStatus for signalling point 0.

8.2.6 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.

8.2.7 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 variables settings and the following logic:

- •f ***$DOMID*** is set, it assumes an AccessSERVICES environment.
- •f ***$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***

Related Information

- Section 8.2.12, **mlogd** on page 8-290
- Section 9.3.8, **apm_start** on page 9-401
- Section 9.3.9, **apm_stop** on page 9-403
- Section 9.3.3, **apm_kill** on page 9-390
- Section 9.3.4, **apm_killall** on page 9-392
- Section 9.3.16, **apm_update** on page 9-417
- Section 9.3.5, **apm_ps** on page 9-394
- Section 8.3.1, **apmconfig** on page 8-305

8.2.8 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.

Related Information

- Section 8.2.7, **apmd** on page 8-280
- Section 8.2.15, **netd** on page 8-294

8.2.9 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 Signaling Gateway Client 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 8.3*) 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.*

Related Information

- Section 8.2.7, **apmd** on page 8-280
- Section 8.2.15, **netd** on page 8-294

8.2.10 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***

Related Information

- Section 8.2.7, **apmd** on page 8-280
- Section 8.2.18, **spmd** on page 8-300
- Section 8.2.20, **upmd** on page 8-303
- Section 8.2.17, **scmd** on page 8-298

8.2.11 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 output, 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.

Related Information

- Section 9.2.12, **ebs_log** on page 9-347
- spm_log() from the SGC API Reference Manual

8.2.12 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:

- f ***$DOMID*** is set, it assumes an AccessSERVICES environment.
- f ***$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

Related Information

- Section 8.2.7, **apmd** on page 8-280
- Section 8.3.1, **apmconfig** on page 8-305

8.2.13 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

Related Information

- Section 8.2.7, **apmd** on page 8-280
- Section 8.2.2, **AccessMOB (mob)** on page 8-271
- Section 8.2.14, **mmi** on page 8-293

8.2.14 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

Related Information

- Section 8.2.7, **apmd** on page 8-280
- Section 8.2.2, **AccessMOB (mob)** on page 8-271
- Section 8.2.13, **mml** on page 8-292

8.2.15 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 10 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

Related Information

- Section 9.2.3, **ebs_audit** on page 9-332

8.2.16 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:

•f te ***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.

•f te ***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.

Related Information

- Section 8.2.7, **apmd** on page 8-280
- Section 3.2.20, **apm_spawn()** on page 3-19 in the *SS8 SGC API Reference Manual*
- Section 10.2.18, **tcm_rmtopen()** on page 10-24 in the *SS8 SGC API Reference Manual*

8.2.17 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***

Related Information

- Section 8.2.7, **apmd** on page 8-280
- Section 8.2.20, **upmd** on page 8-303

8.2.18 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 and artic8260) 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

Related Information

- Section 8.2.7, **apmd** on page 8-280
- Section 8.2.20, **upmd** on page 8-303
- Section 8.2.15, **netd** on page 8-294
- Section 9.3.8, **apm_start** on page 9-401

- Section 8.2.17, **scmd** on page 8-298

8.2.19 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 8.3). 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.*

8.2.20 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***

Related Information

- Section 8.2.7, **apmd** on page 8-280
- Section 8.2.17, **scmd** on page 8-298

8.3 ConfigurationFiles

8.3.1 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 8-313.

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.
  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
  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.
  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**
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.

retrydel

hstate - 1) number of attempts to restart process in the last

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.

Related Information

- Section 8.2.7, **apmd** on page 8-280
- Section 9.3.16, **apm_update** on page 9-417
- Section 9.3.2, **apm_getstate** on page 9-389
- Section 9.3.7, **apm_setstate** on page 9-399
- Section 8.3.2, **apmconfig.old** on page 8-318
- 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
```

8.3.2 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.

- **WA I T**: 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 comma (**,**); 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.*

Related Information

- Section 8.2.7, **apmd** on page 8-280
- Section 9.3.16, **apm_update** on page 9-417
- Section 9.3.2, **apm_getstate** on page 9-389
- Section 9.3.7, **apm_setstate** on page 9-399
- Section 8.3.1, **apmconfig** on page 8-305

8.4 Signaling Gateway Client Processes

8.4.1 aspd

NAME

aspd	Application Server Process daemon

SYNOPSIS

aspd [-h] [-p ipc port | -a ipc path] [-f cfg file]

DESCRIPTION

aspd is a Application Server Process (ASP) process running on Signaling Gateway Client. It processes traffic from MTP3 user parts, translates them to M3UA (MTP3 User Adaptation) format, and sends them to the IP network over SCTP. Similarly, it receives from messages from the IP network over SCTP, translates them to MTP3 user primitives, and sends them to the MTP3 user parts running in Signaling Gateway Client. There can be only one instance of aspd running on each host.

-h	Print help information.
-f cfg file	Specify path/file name of the configuration file which contains various timer parameters.

FILES

The ASP configuration parameters are defined in the ***$SGCHOME/sgc/RUN/config/ASP/aspd.conf*** file. Edit this file to modify any parameters. After modification, send a HUP signal to sctp so that it re-reads the configuration file, e.g. kill -HUP pid where pid is the process ID of aspd.

Caution: *Never start aspd manually. To be sure that process management is in effect, it should always be started using the sgc_start command. Edit the apmconfig file in the $EBSHOME/access/RUN/config/PMGR/apmconfig directory to modify the default command line arguments.*

Caution: *The configuration options for aspd are intended for debugging purposes* ***ONLY****. Modification of this file is not recommended without first getting technical assistance from NewNet Communication Technologies.*

ENVIRONMENT

The $EBSHOME and $SGCHOME environment variables must be set to the base installation directory of Distributed7 and Signaling Gateway Client software. These variables
are set automatically when the user logs in as sgcadm.

Related Information

- Section 8.3.1, **apmconfig** on page 8-305
- Section 8.4.1, **aspd** on page 8-323
- Section 9.8.2, **sgc_start** on page 9-479

Chapter 9: User Commands

9.1 ChapterOverview

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 9-1: User Command Summary

Command	Description
ebs_alarm	Trigger alarm condition.
ebs_apidemo	Demonstrates SPM API capabilities introduced with Distributed7.
ebs_audit	Audits dynamic data of a host.
ebs_brdfinfo	Get and clear SS7 board crash dump.
ebs_cfgbrd	Configure and unconfigure SS7 board.
ebs_config	Configure system operation mode.
ebs_dbconfig	Save or restore configuration data
ebs_dnlbrd	Reset and download SS7 board.
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_mngbrd	Start and stop SS7 board sanity checks.
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_oldapidemo	Demonstrates SPM API capabilities of releases prior to Distributed7.
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 9-1: 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 9-1: 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
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
AccessStatus	Monitors signaling point configuration, MTP level 2 and level 3 status, and traffic capacity utilization of SS7 links.
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.
sgc_pkgrm	Removes an installed version of the Signaling Gateway Client software
sgc_start	Starts the Signaling Gateway Client software
sgc_stop	Stops the Signaling Gateway Client software
sgc_setrelease	Updates the Signaling Gateway Client and Distributed7 files to the newest release
sgc_trace	Activates the tracing function

***Important**: Please see Chapter 3 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
```

Online 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
```

9.2 PlatformUtilities

9.2.1 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 sub-system 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 command line 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
```

Related Information

- alarmd
- Section 6.2.2, **alm_report()** on page 6-4 in the API Reference Manual
- Section 2.2.1, **spm_alarm()** on page 2-2 in the API Reference Manual

9.2.2 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 end points

• bind and unbind addresses at individual service end points

• 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 end point

• activate and deactivate message loopback at a service end point

• 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

Related Information

• Section 9.2.15, **ebs_oldapidemo** on page 9-353

9.2.3 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 signaling 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.*

Related Information

- Section 9.2.21, **ebs_ps** on page 9-359
- Section 9.2.27, **ebs_showlink** on page 9-373
- Section 9.2.31, **ebs_sync** on page 9-379

9.2.4 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 signaling 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...
```

Related Information

- spmd
- Section 9.3.8, **apm_start** on page 9-401
- Section 9.2.8, **ebs_dnlbrd** on page 9-341
- Section 9.2.5, **ebs_cfgbrd** on page 9-335
- Section 9.2.14, **ebs_mngbrd** on page 9-351

9.2.5 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 signaling 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
```

Related Information

- spmd
- Section 9.2.4, **ebs_brdfinfo** on page 9-333
- Section 9.2.8, **ebs_dnlbrd** on page 9-341
- Section 9.3.8, **apm_start** on page 9-401
- Section 9.2.14, **ebs_mngbrd** on page 9-351
- streamio

9.2.6 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.*

Caution: *This script should be run 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
```

Related Information

- add_drv
- Section 9.2.16, **ebs_modinstall** on page 9-354
- Section 9.2.18, **ebs_modunload** on page 9-356
- Section 9.2.17, **ebs_modremove** on page 9-355
- Section 9.7.16, **db2date** on page 9-475
- Section 9.7.17, **db2text** on page 9-476

9.2.7 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

9.2.8 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 signaling 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
```

Related Information

•spmd

- Section 9.2.5, **ebs_cfgbrd** on page 9-335
- Section 9.2.4, **ebs_brdfinfo** on page 9-333
- Section 9.2.14, **ebs_mngbrd** on page 9-351
- Section 9.3.8, **apm_start** on page 9-401

9.2.9 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.

Related Information

- Section 2.2.36, **spm_strerror()** on page 2-57 in the *API Reference Manual*
- Section 2.2.7, **spm_errgroup()** on page 2-15 in the *API Reference Manual*

9.2.10 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 end point 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.

Related Information

•netd

• Section 9.2.27, **ebs_showlink** on page 9-373

• Section 9.2.31, **ebs_sync** on page 9-379

9.2.11 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.

9.2.12 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 end point, 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 end points that have message logging currently active.

When the message logging capability is activated, a copy of each message received or sent through the service end point 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 end points will automatically be deactivated. Also, if a process being logged terminates, logging at all service end points 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 end point. Deactivation of message logging at an end point 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 signaling 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 Signaling Network Management are uniquely identified by the logical signaling point number and the user part number of 0.
- The ISUP process is identified by the logical signaling 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 signaling point number. This information may be seen in the output of *ebs_ps*.

Related Information

- logd
- Section 9.2.21, **ebs_ps** on page 9-359

9.2.13 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 end point, 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 end points that have message loopback currently active.

When the message loopback capability at a particular service end point is activated, all messages sent out and/or about to be received through the end point 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 end point affects only the messages originated from the end point, 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 end point. Deactivation of message logging at an end point 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 end points associated with that process as well as at all appropriate end points under the Distributed7 platform.

After entering the command to activate or deactivate loopback, prompts will appear for the object type of the end point 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.

Related Information

•Section 9.2.21, **ebs_ps** on page 9-359

9.2.14 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 signaling 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
```

Related Information

- spmd
- Section 9.3.8, **apm_start** on page 9-401
- Section 9.2.5, **ebs_cfgbrd** on page 9-335
- Section 9.2.4, **ebs_brdfinfo** on page 9-333
- Section 9.2.8, **ebs_dnlbrd** on page 9-341
- streamio

9.2.15 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

Related Information

- Section 9.2.2, **ebs_apidemo** on page 9-331

9.2.16 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.*

Related Information

- Section 9.2.17, **ebs_modremove** on page 9-355

9.2.17 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.*

Related Information

• Section 9.2.16, **ebs_modinstall** on page 9-354

9.2.18 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.*

Related Information

- Section 9.2.17, **ebs_modremove** on page 9-355

9.2.19 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 end point, 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 signaling 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.*

Related Information

- upmd
- Section 9.2.21, **ebs_ps** on page 9-359

9.2.20 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.

Related Information

- pkgrm
- Section 9.2.26, **ebs_setrelease** on page 9-372

Note: *You must have "root" privileges to execute this script.*

9.2.21 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 signaling 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 end point 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 9-2: 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 signaling 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 signaling 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 Signaling Message Handling (SMH) protocol.
		snm: MTP Signaling Network Management (SNM) Multiplexer.
		sccp: Signaling Connection Control Part (SCCP) Multiplexer.
		tcap: Transaction Capabilities Application Part (TCAP) Multiplexer.

Table 9-2: 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 end point 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/#/#	Signaling Connection Control Part (SCCP) Multiplexer.
	snm/#/#	MTP Signaling Network Management (SNM) Multiplexer.
	spm/#	Signaling Point Multiplexer.
	tcap/#/#	Transaction Capabilities Application Part (TCAP) Multiplexer.
	upm/#/#	Message Transfer Part (MTP) User Part Multiplexer. Used for implementing the MTP Signaling 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.

Table 9-2: ebs_ps Output Description

Column Heading	Valid Values	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.
TIME		The time at which the process registered.

Related Information

- Section 9.2.31, **ebs_sync** on page 9-379
- Section 9.2.32, **ebs_sysinfo** on page 9-381
- Section 9.2.23, **ebs_qlist** on page 9-365

9.2.22 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 end point 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 end point 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 9-3: ebs_qinfo Output Description

Column Heading	Description
MUX	The STREAMS multiplexer used for establishing the service end point 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.

Related Information

- Section 9.2.21, **ebs_ps** on page 9-359
- Section 9.2.23, **ebs_qlist** on page 9-365
- Section 9.2.24, **ebs_qstat** on page 9-367

9.2.23 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 end point 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 end point 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 9-4: ebs_qlist Output Description

Column	Description
MUX	The STREAMS multiplexer used for establishing the service end point 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 signaling 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.

Table 9-4: ebs_qlist Output Description

Column	Description
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.
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.

Related Information

- Section 9.2.21, **ebs_ps** on page 9-359
- Section 9.2.22, **ebs_qinfo** on page 9-363
- Section 9.2.24, **ebs_qstat** on page 9-367

9.2.24 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 end point, i.e., STREAMS connection, when the end point is established during an ***spm_open()*** call and maintains them until the end point 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 end point 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 9-5: ebs_qstat Output Description

Field Name	Description
MUX	Specifies the STREAMS multiplexer used for establishing the service end point 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 9-5: 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.

Related Information

- Section 9.2.21, **ebs_ps** on page 9-359
- Section 9.2.22, **ebs_qinfo** on page 9-363
- Section 9.2.23, **ebs_qlist** on page 9-365
- spm_qparams()
- spm_stroptions()

9.2.25 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.*

Related Information

- date(1)
- alarmd
- Section 9.3.6, **apm_report** on page 9-397

9.2.26 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.

9.2.27 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 signaling 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 9-6).

-x

Indicates that ***ebs_showlink*** should not bind an address to the service end point 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 9-6: 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.
	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.

Table 9-6: ebs_showlink Output Description

Column	Valid Values	Description
HOST		Name of the host machine associated with the link. For SS7 board connections, name identifies the host where the SS7 signaling 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.
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

Related Information

- netd
- spmd
- Section 9.2.10, **ebs_hbeat** on page 9-344

9.2.28 ebs_shutdown

NAME

ebs_shutdown Stops Distributed7 software, remotely

SYNOPSIS

ebs_shutdown [hostname(s) ... ***]***

DESCRIPTION

ebs_shutdown Requeststhe*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.

Related Information

- Section 9.2.30, **ebs_stop** on page 9-378
- Section 9.3.9, **apm_stop** on page 9-403

9.2.29 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).*

Related Information

•apmd

•spmd

• Section 9.3.8, **apm_start** on page 9-401
• Section 9.2.28, **ebs_shutdown** on page 9-376
• Section 9.2.30, **ebs_stop** on page 9-378
• Section 9.3.9, **apm_stop** on page 9-403

9.2.30 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.*

Related Information

- Section 9.2.21, **ebs_ps** on page 9-359
- Section 9.2.28, ebs_shutdown
- Section 9.2.29, **ebs_start** on page 9-377

9.2.31 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.

Caution: *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 signaling 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 signaling 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 end point 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.

✓ ***Important**: Dynamic data synchronization is only relevant to distributed Distributed7 configurations. Using this command in a stand-alone configuration has no effect.*

Related Information

- Section 9.2.3, **ebs_audit** on page 9-332
- Section 9.2.10, **ebs_hbeat** on page 9-344

9.2.32 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 end point 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.

9.2.33 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 ADC Telecommunications, Inc. 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 end point 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.

✓ ***Note**: 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.*

9.2.34 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

Related Information

- Section 9.2.16, **ebs_modinstall** on page 9-354
- sytem(4)
- sysdef(1)

9.2.35 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 an output in column format informing the user about the driver, type of board, physical number of the slot carrying the board, and instance number of the boardand driver state (if the operating system is AIX).

*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.

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.
- 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 sixtyfour 64 Kpbs 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.*

Slot - the physical number of slot carrying the board. Its value depends on the hardware configuration of the host computer.

Instance - the instance number of the board among other boards of its type. its value can be in the range of 0 to 7.

State - the state of the device driver (applicable to AIX systems only).

State - the state of the device driver. Applicable only to AIX systems.

SAMPLE OUTPUT

```
Solaris version output sample (CompactPCI bus):

Driver    Board Type Slot   Instance
--------- ---------- ------ --------
cpc3xpq   cpc370pq   1      0
cpc3xpq   cpc372pq   2      1
pmc8260   pmc8260    3-pmc1 0
pmc8260   pmc8260    3-pmc2 1
artic8260 artic1000  4      0

Solaris version output sample (PCI bus):

Driver    Board Type Slot   Instance
--------- ---------- ------ --------
pci334    pci370     1      0
pci334    pci334     3      1
pci3xpq   pci372pq   2      0
pci3xapq  pci372apq  4      0
pmc8260   pmc8260    5      0
artic8260 artic2000  6      0

Solaris version output sample (Sbus):

Driver    Board Type  Slot  Instance
--------  ----------  ----  --------
sbs334    sbs370       1     0
sbs334    sbs334       3     1
sbs334    sbs370       2     0

AIX version output sample:

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
```

AIX output sample: **Board TypeSlotInstanceState**

Driver **-------------------------------**

-------- **pci3701 0** **Available**

pci334 **pci3343 1** **Available**

pci334

	pci3xpq pci372pq20	**Available**

9.3 APMUtilities

9.3.1 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***.

Related Information

- apmd
- Section 9.3.8, **apm_start** on page 9-401

9.3.2 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.

Related Information

- apmd
- Section 9.3.7, **apm_setstate** on page 9-399

9.3.3 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.

Related Information

- apmd
- Section 9.3.9, **apm_stop** on page 9-403
- Section 9.3.4, **apm_killall** on page 9-392
- Section 9.3.3, **apm_kill** on page 9-390

9.3.4 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.

Related Information

•apmd

- Section 9.3.9, **apm_stop** on page 9-403
- Section 9.3.3, **apm_kill** on page 9-390
- Section 9.3.7, **apm_setstate** on page 9-399

9.3.5 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 9-7. 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 9-7: 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.

Table 9-7: apm_ps Output Description

Column Heading	Possible Values	Description
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.
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.

Related Information

- p(1)
- Section 9.2.21, **ebs_ps** on page 9-359
- apmd
- Section , **apm_init()** on page 3-5 in the API Reference Manual

9.3.6 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).

	Includes only the log messages from the alternate [secondary] log files.
-a ***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.*

Related Information

- date(1)
- mlogd
- Section 9.2.25, **ebs_report** on page 9-370

9.3.7 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.*

Related Information

•apmd

- Section 9.3.2, **apm_getstate** on page 9-389
- Section 9.3.4, **apm_killall** on page 9-392
- Section 9.3.16, **apm_update** on page 9-417

9.3.8 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.

Related Information

- apmd
- Section 9.3.9, **apm_stop** on page 9-403

9.3.9 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.

Related Information

•apmd

- Section 9.3.8, **apm_start** on page 9-401
- Section 9.3.4, **apm_killall** on page 9-392
- Section 9.3.3, **apm_kill** on page 9-390
- Section 9.3.7, **apm_setstate** on page 9-399

9.3.10 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`

Related Information

- Section 9.3.13, **apm_trinit** on page 9-411
- Section 9.3.11, **apm_trclear** on page 9-408
- Section 9.3.12, **apm_trgetmask** on page 9-409

9.3.11 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* <u>*permanently erase*</u> *all trace messages logged for the specified environment.*

Related Information

- Section 9.3.13, **apm_trinit** on page 9-411
- Section 9.3.12, **apm_trgetmask** on page 9-409
- Section 9.3.14, **apm_trsetmask** on page 9-413
- Section 9.3.15, **apm_trshow** on page 9-415
- Section 9.3.10, **apm_trcapture** on page 9-405
- Section 3.2.21, **apm_trace()** on page 3-20 in the API Reference Manual

9.3.12 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.

Related Information

- Section 9.3.13, **apm_trinit** on page 9-411
- Section 9.3.11, **apm_trclear** on page 9-408
- Section 9.3.14, **apm_trsetmask** on page 9-413
- Section 9.3.15, **apm_trshow** on page 9-415
- Section 9.3.10, **apm_trcapture** on page 9-405
- Section 3.2.21, **apm_trace()** on page 3-20 in the API Reference Manual

9.3.13 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.*

Related Information

- Section 9.3.11, **apm_trclear** on page 9-408
- Section 9.3.12, **apm_trgetmask** on page 9-409
- Section 9.3.14, **apm_trsetmask** on page 9-413
- Section 9.3.15, **apm_trshow** on page 9-415
- Section 9.3.10, **apm_trcapture** on page 9-405
- Section 3.2.21, **apm_trace()** on page 3-20 in the API Reference Manual

9.3.14 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,...,mask63Modifies 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,...,mask63Modifies 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,...,progid511Manipulates 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).

Related Information

- Section 9.3.13, **apm_trinit** on page 9-411
- Section 9.3.11, **apm_trclear** on page 9-408
- Section 9.3.12, **apm_trgetmask** on page 9-409
- Section 9.3.15, **apm_trshow** on page 9-415
- Section 9.3.10, **apm_trcapture** on page 9-405
- Section 3.2.21, **apm_trace()** on page 3-20 in the API Reference Manual

9.3.15 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.

Related Information

- Section 9.3.13, **apm_trinit** on page 9-411
- Section 9.3.11, **apm_trclear** on page 9-408
- Section 9.3.12, **apm_trgetmask** on page 9-409
- Section 9.3.14, **apm_trsetmask** on page 9-413
- Section 9.3.10, **apm_trcapture** on page 9-405
- Section 3.2.21, **apm_trace()** on page 3-20 in the API Reference Manual

9.3.16 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.*

Related Information

- apmd
- Section 9.3.7, **apm_setstate** on page 9-399

9.4 DSMUtilities

9.4.1 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 end point 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

9.4.2 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.

9.4.3 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.

9.4.4 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 9-8: 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.	-

Table 9-8: dsm_list Output Column Description

Output Field Name	Description	Keyword Values
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.	-
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.

Related Information

- Section 9.4.6, **dsm_stat** on page 9-426
- Section 4.2.6, **dsm_getstat()** on page 4-10 in the API Reference Manual

9.4.5 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.*

9.4.6 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 9-9: 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.

Table 9-9: dsm_stat Output Column Description

Output Field Name	Description
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.
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.

9.5 DKMUtilities

9.5.1 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

Related Information

- dkmd

9.5.2 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.

Related Information

- dkmd

9.5.3 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.

Related Information

- dkmd
- Section 9.5.4, **dkm_list** on page 9-431
- Section 9.5.7, **dkm_stat** on page 9-439

9.5.4 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.

Related Information

- `dkmd`
- Section 9.5.7, **dkm_stat** on page 9-439
- Section 9.5.3, **dkm_dump** on page 9-430

9.5.5 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.

Related Information

- dkmd

9.5.6 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).

Related Information

- dkmd
- Section 9.5.4, **dkm_list** on page 9-431

9.5.7 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.

Related Information

- dkmd
- Section 9.5.4, **dkm_list** on page 9-431
- Section 9.5.3, **dkm_dump** on page 9-430

9.5.8 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)
```

Related Information

- Section 16.2.11, **dra_new_record()** on page 16-13 in the API Reference Manual
- Section 16.2.3, **dra_del_record()** on page 16-6 in the API Reference Manual
- Section 16.2.2, **dra_del_locked()** on page 16-5 in the API Reference Manual
- Section 16.2.6, **dra_find_record()** on page 16-9 in the API Reference Manual
- Section 16.2.5, **dra_find_inseq()** on page 16-8 in the API Reference Manual
- Section 16.2.16, **dra_validate()** on page 16-18 in the API Reference Manual
- Section 16.2.13, **dra_relock_sync()** on page 16-15 in the API Reference Manual
- Section 16.2.12, **dra_relock_async()** on page 16-14 in the API Reference Manual
- Section 16.2.14, **dra_rls_lock()** on page 16-17 in the API Reference Manual
- Section 16.2.10, **dra_lock_seg_priv()** on page 16-13 in the API Reference Manual
- Section 16.2.15, **dra_rls_seg_priv()** on page 16-18 in the API Reference Manual
- Section 16.2.8, **dra_get_dkm_id()** on page 16-12 in the API Reference Manual
- Section 16.2.4, **dra_destroy()** on page 16-7 in the API Reference Manual
- Section 15.2.4, **dkm_get()** on page 15-6 in the API Reference Manual
- sdkm_getlist ()

9.6 TCAPUtilities

9.6.1 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 signaling 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 signaling point number associated.
- SSN - The SCCP subsystem number associated.

✓ ***Note****: 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.*

Related Information

- Section 9.6.2, **rtc_stat** on page 9-450
- rtc_agent
- tcm_rmtopen

9.6.2 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 signaling 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.

Note*: 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.*

Related Information

- Section 9.6.1, **rtc_dump** on page 9-448
- rtc_agent
- tcm_rmtopen

9.6.3 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.

Related Information

- tcmd
- Section 9.6.4, **tcm_list** on page 9-452
- Section 9.6.5, **tcm_stat** on page 9-454

9.6.4 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

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.

Related Information

- tcmd
- Section 9.6.5, **tcm_stat** on page 9-454
- Section 9.6.6, **tcm_tune** on page 9-455

9.6.5 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).

Related Information

- tcmd
- Section 9.6.4, **tcm_list** on page 9-452
- Section 9.6.6, **tcm_tune** on page 9-455

9.6.6 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).

Related Information

- tcmd
- Section 9.6.4, **tcm_list** on page 9-452
- Section 9.6.5, **tcm_stat** on page 9-454

9.7 Virtual Board Utilities

9.7.1 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.

Related Information

- Section 9.7.4, **vb_connhosts** on page 9-462
- Section 9.7.3, **vb_config** on page 9-460
- Section 9.7.2, **vb_bridge** on page 9-458

9.7.2 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***

Related Information

- Section 9.7.4, **vb_connhosts** on page 9-462
- Section 9.7.3, **vb_config** on page 9-460

9.7.3 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 ADC Telecommunications, Inc. 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.

Related Information

- Section 9.7.4, **vb_connhosts** on page 9-462
- Section 9.7.10, **vb_startup** on page 9-468
- Section 9.7.2, **vb_bridge** on page 9-458
- Section 9.7.1, **vb_addhost** on page 9-457

• Section 9.7.5, **vb_connports** on page 9-463
• Section 9.7.6, **vb_discport** on page 9-464
• Section 9.7.9, **vb_reset** on page 9-467

9.7.4 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

Related Information

- Section 9.7.3, **vb_config** on page 9-460
- Section 9.7.2, **vb_bridge** on page 9-458
- Section 9.7.10, **vb_startup** on page 9-468
- vbrd

9.7.5 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.

Related Information

- Section 9.7.4, **vb_connhosts** on page 9-462
- Section 9.7.3, **vb_config** on page 9-460
- Section 9.7.2, **vb_bridge** on page 9-458

9.7.6 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).

Related Information

- Section 9.7.4, vb_connhosts
- Section 9.7.3, vb_config
- Section 9.7.2, **vb_bridge** on page 9-458
- Section 9.7.10, **vb_startup** on page 9-468

9.7.7 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.

Related Information

- Section 9.7.4, **vb_connhosts** on page 9-462
- Section 9.7.3, **vb_config** on page 9-460
- Section 9.7.2, **vb_bridge** on page 9-458

9.7.8 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.

Related Information

- Section 9.7.4, **vb_connhosts** on page 9-462
- Section 9.7.3, **vb_config** on page 9-460
- Section 9.7.2, **vb_bridge** on page 9-458

9.7.9 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.

Related Information

- Section 9.7.4, **vb_connhosts** on page 9-462
- Section 9.7.3, **vb_config** on page 9-460
- Section 9.7.2 on page 9-458

9.7.10 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.

Related Information

- Section 9.7.4, **vb_connhosts** on page 9-462
- Section 9.7.3, **vb_config** on page 9-460
- Section 9.7.2, **vb_bridge** on page 9-458
- Section 9.7.1, **vb_addhost** on page 9-457
- Section 9.7.5, **vb_connports** on page 9-463
- Section 9.7.6, **vb_discport** on page 9-464
- Section 9.7.9, **vb_reset** on page 9-467

9.7.11 snmptest

NAME

snmptest Communicates with a network entity using SNMP Requests.

Copyright 1988, 1989, 1991, 1992 by Carnegie Mellon University - All Rights Reserved

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.

9.7.12 snmptrapd

NAME

snmptrapd Receives and prints SNMP traps.

Copyright 1988, 1989, 1991, 1992 by Carnegie Mellon University - All Rights Reserved

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.

Related Information

• 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*

9.7.13 snmpwalk

NAME

snmpwalk Communicates with a network entity using SNMP GET Next Requests

Copyright 1988, 1989, 1991, 1992 by Carnegie Mellon University - All Rights Reserved

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.*

9.7.14 snmpget

NAME

snmpget Communicates with a network entity using SNMP GET Requests

Copyright 1988, 1989, 1991, 1992 by Carnegie Mellon University - All Rights Reserved

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.

9.7.15 AccessStatus

NAME

AccessStatus Monitors signaling 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 Signaling 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	Signaling 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	Signaling 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 (Signaling 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

EXAMPLE

AccessStatus 0

Starts up AccessStatus for signaling point Ø.

9.7.16 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.

Related Information

- Section 9.7.17, **db2text** on page 9-476

9.7.17 db2text

NAME

db2text Converts all release ADC Telecommunications, Inc. 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 ADC Telecommunications, Inc. Distributed7 executables.*

SYNOPSIS

db2text <directoryname>

DESCRIPTION

db2text Utility to convert the binary database files in the ***$EBSHOME/access/RUN<sp#>/DBfiles*** directories into text files containing the appropriate MML commands to recreate the configuration. The utility determines the current ADC Telecommunications, Inc. Distributed7 version by checking the executables in ***$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 signaling point (0, 1, .. 7). The file names have the format of *mml_<layer>_<sp#>*.txt. For example, if MTP and SCCP were configured for signaling points 0 and 1, then the command converts the database files in directories ***$EBSHOME/access/RUN0/DBfiles*** for signaling point 0 and ***$EBSHOME/access/RUN1/DBfiles*** for signaling 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]?*

9.8 Signaling Gateway Client Application Utilities

9.8.1 sgc_pkgrm

NAME

sgc_pkgrm	Removes a version of the Signaling Gateway Client package software.

SYNOPSIS

sgc_pkgrm version

DESCRIPTION

The **sgc_pkgrm** command removes a version of the Signaling Gateway Client package (package name SGCcore). If there are files left in the software directory after removing the software, (files that are not part of the original package), then the software directory remains on the disk and **sgcadm** user account is preserved. Otherwise, the entire software directory and user account are removed.

version	Version number of the software to be removed. Use the UNIX **pkginfo** command to find out what version(s) are installed if you are not sure of the software version: ***pkginfo | grep SGCcore***

9.8.2 sgc_start

NAME

sgc_start	Starts the Signaling Gateway Client software.

SYNOPSIS

sgc_start [d7] [sp[Ø-7]] [sc[0-7]][is[0-7]] [tc] [om[0-7]] [ip] [asp]

DESCRIPTION

sgc_start

The sgc_start command starts the Signaling Gateway Client and Distributed7 software. The Signaling Gateway Client system consists of the
basic platform processes, SS7 node processes and *Signaling Gateway Client processes*. This command allows the user to start the system in different phases in which one or more subset of processes are running. The following states can be used as command line arguments to start Signaling Gateway Client:

d7	Starts only the basic platform processes on the local host.
sp[Ø-7]	Starts the MTP3 process for a signaling point on the local host. Valid signaling points are Ø through 7. *All basic platform processes are also started automatically, if they have not already been started, when this state is specified.*
sc[0-7]	Starts the SCCP process for a signaling point on the local host. Valid signaling points are 0 through 7. All basic platform processes and MTP3 process are started automatically, if they have not already been started, when this state is specified.
is[0-7]	Starts the ISUP process for a signaling point on the local host. Valid signaling points are 0 through 7. All basic platform processes and MTP3 process are started automatically, if they have not already been started, when this state is specified.
tc	Starts the TCAP process on the local host. All basic platform processes are started automatically, if they have not already been started, when this state is specified.
om[0-7]	Starts the OMAP process for a signaling point on the local host. Valid signaling points are 0 through 7. All basic platform processes are started automatically, if they have not already been started, when this state is specified.
ip	Starts the SCTP process on the local host.
asp	Starts the ASP process for a signaling point on the local host. Basic platform processes and SCTP process are also started automatically, if they have not already been started, when this state is specified.

EXAMPLE

1. To start MTP3 and SCCP on signaling point 0:

$ sgc_start sc0

This command starts basic platform processes, MTP3 for signaling point 0, and SCCP for signaling point 0. This is equivalent of executing:

$ sgc_start d7 sp0 sc0

2. To start MTP3, ISUP and SCCP on signaling point 0 and TCAP and ASP:

$ sgc_start sc0 sc1 tc asp

This command starts basic platform processes, MTP3 for signaling point 0, SCCP for signaling point 0, SCCP for signaling point 0, MTP3 for signaling point 1, SCCP for signaling point 1, TCAP, SCTP and ASP processes. The following command performs the same operation:

$sgc_start d7 sp0 sc0 sp1 sc1 tc ip asp

9.8.3 sgc_stop

NAME

sgc_stop	Stops the Signaling Gateway Client software.

SYNOPSIS

sgc_stop [d7] [sp[Ø-7]] [sc[0-7]][is[0-7]] [tc] [om[0-7]] [asp]

DESCRIPTION

sgc_stop	The sgc_stop command stops the Signaling Gateway Client and Distributed7 software. The Signaling Gateway Client system has the *basic platform processes*, *SS7 node processes* and *Signaling Gateway Client processes*. This command allows the user to stop one or more subset(s) of these processes. The following states can be used as command line arguments to stop Signaling Gateway Client:
d7	Stops all Signaling Gateway Client and Distributed7 processes on the local host.
sp[Ø-7]	Stops the MTP3 process for a signaling point on the local host. Valid signaling points are Ø through 7. SCCP and ISUP processes stop automatically when this state is specified.
is[0-7]	Stops the ISUP process for a signaling point on the local host. Valid signaling points are 0 through 7.
tc	Stops the TCAP process on the local host.
om[0-7]	Stops the OMAP process for a signaling point on the local host. Valid signaling points are Ø through 7.
asp	Stops the ASP process on the local host.

EXAMPLE

To stop MTP3, SCCP on signaling point 0 and also the common TCAP process:

$ sgc_stop sp0 tc

This command is equivalent to the following:

$ sgc_stop sc0 sp0 tc

9.8.4 sgc_setrelease

NAME

sgc_setrelease Sets the Signaling Gateway Client software.

SYNOPSIS

sgc_setrelease version | -i

DESCRIPTION

sgc_setrelease The **sgc_setrelease** command activates a specific version of the Signaling Gateway Client and Distributed7 software. The following arguments are supported:

version	Version of the software to be activated.
-i	The -i option prints the information about the currently activated software version, such as the base installation directory, Signaling Gateway Client version and Distributed7 version.

9.8.5 sgc_trace

NAME

sgc_trace Runtime tracing utility.

SYNOPSIS

sgc_trace [asp]

sgc_trace [-c]

sgc_trace -s | -u | -m asp | all

DESCRIPTION

sgc_trace The sgc_trace command is used to collect traces for the ASP process for debugging purposes. Runtime tracing can be enabled or disabled dynamically, and traces that are collected are stored in a circular memory buffer, with new traces overwriting old ones when the buffer is full. This command supports the following command line options:

-s [asp]	The -s option enables the tracing for one or more processes. • Specify *asp* as the argument to trace the ASP process
-u [asp] | all	The -u option disables the tracing for one or more processes. • Specify *asp* as the argument to disable tracing for the ASP process • Specify *all* as the argument to disable tracing for all processes

-m [asp] | all	The -m option displays the current trace statuses to show whether they are enabled or disabled on the system. The arguments are entered the same way as in the -u option (see the previous bullet).
-l [asp]	The -l option displays the traces collected. The arguments are entered the same way as in the -u option (see the previous bullet). All existing traces in the buffer are displayed when no argument is specified because the traces are not filtered.
-c	The -c option clears all existing traces in the buffer.

Chapter 10: Maintenance and Troubleshooting

10.1 Overview

This chapter identifies the alarms and troubleshooting tools of the Signaling Gateway Client.

If necessary, the Technical Assistance Center (TAC) can be reached by phone at *(800) 416-1624 US,* or *(408) 432-2600 International,* or also by email at *support@newnet.com*.

10.2 Software Maintenance

10.2.1 Configuration Backup

Periodically the configurations on each of the distributed hosts should be backed up to a tape to prevent data loss in the event of a hard disk failure or file corruption. It is recommended that the following configuration directories be backed up:

- ***$EBSHOME/access/RUN*** and ***$EBSHOME/access/RUNx***, where ***x*** is the signaling point number
- $SGCHOME/sgc/RUN

To back up a directory to tape, insert a tape to the tape drive connected to the host, then execute the following UNIX command:

t**ar cvf *tape_device backup_dir***

where ***tape_device*** is the device name of the tape drive, and ***backup_dir*** is the directory to be copied to tape.

10.2.2 Monitoring Alarms

By default, alarms are displayed on the console. The operator should pay attention to alarms that are displayed, or check for any outstanding alarms using MML, for example:

display-strdalm:;

The operator should also use MML to delete obsolete alarms of SET_ALARM type that are non-self-clearing, for example:

delete-strdalm:hostname=x,group=x,module=x,type=x,last_occur=x;

Periodically it is also recommended to check the alarm log files in **$EBSHOME/access/RUN/alarmlog** to observe for potential problems or any unusual system behavior.

10.2.3 Monitoring Logs

The Master Log (MLog) is used to log system events that are noteworthy, including system start/stop, major configuration changes, and faulty events. Master Log files are created daily and are stored under the **$EBSHOME/access/RUN/mlog** directory. Periodically monitoring these log files may help detect problems or unusual system behavior early.

10.3 Troubleshooting Tools

10.3.1 Alarms

When problem occurs, alarms should be checked both in the alarm log file and through MML. The following tables lists the Signaling Gateway Client alarms and the corresponding actions that can be taken to remedy the problem. If the error condition persists, contact the Technical Assistance Center (TAC) for help. See Section 6.5.6.2 Alarm Groups for information about the Distributed7 alarm groups.

Table 10-1: ASP Alarm Group

Alarm No.	Severity	Type	Message	Operation
020101	INFO	EVENT	ASP: ASP terminated	Indicates the termination of ASP. Check the mlog file to determine reason for termination. No action is required for normal termination.
020102	CRITICAL	EVENT	ASP: EBSHOME environment variable not set	Make sure that Signaling Gateway Client is started by sgcadm. If it was started by sgcadm and this alarm still happens, then check the sgcadm's .cshrc file and be sure that the environment variable EBSHOME is exported.
020103	CRITICAL	EVENT	ASP: Failed to read license file	Be sure that you have valid license before starting Signaling Gateway Client.
020104	CRITICAL	EVENT	ASP: License feature not found in license file	Be sure that you have valid license before starting Signaling Gateway Client.
020105	CRITICAL	EVENT	ASP: License key corrupted	Be sure that you have valid license before starting Signaling Gateway Client.
020106	CRITICAL	EVENT	ASP: Not licensed for local host	Be sure that you have valid license before starting Signaling Gateway Client.
020109	CRITICAL	EVENT	ASP: Licensed number of ASP exceeded system capacity (max %d)	Be sure that you have valid license before starting Signaling Gateway Client.

Table 10-1: ASP Alarm Group (Continued)

Alarm No.	Severity	Type	Message	Operation
02010a	CRITICAL	EVENT	ASP: MTP API initialization error	Try stopping and restarting MTP: 1. sg_stop sgp spX, where X is the sp number. 2. sg_start spX, where X is the sp number. 3. sg_start sgp Contact TAC if problem persists.
02010b	CRITICAL	EVENT	ASP: Gateway registration failed	Change the OPERST of SGCSPNA to INACT, then change it to ACT. Contact TAC if problem persists.
02010c	CRITICAL	EVENT	ASP: SCTP IPC initialization failed	Make sure SCTP process is running. If not, start it by running sg_start ip, then restart SGP by running sg_start sgp. Contact TAC if problem persists.

Table 10-2: OAM Alarm Group

Alarm No.	Severity	Type	Message	Operation
020201	MAJOR	EVENT	ASP: Database error	ASP failed to perform database operation. Check mlog file for the failure reason. Verify that databases under $EBSHOME/RUNx/DBfiles have write permission for sgcadm, where x is the signaling point number.
020202	MAJOR	EVENT	ASP: Managed Object creation error	ASP failed to create Managed Object definitions during startup. Check the mlog file for the failure reason. If the error indicates that MO already exists, then no action is required. Otherwise, verify that the MML commands can be run on all Signaling Gateway Client hosts. If the commands can be run, no action is required. Otherwise, restart Signaling Gateway Client.

Table 10-3: M3UA Alarm Group

Alarm No.	Severity	Type	Message	Operation
020301	CRITICAL	EVENT	ASP: M3UA stack initialization failed	ASP failed to initialize M3UA stack during startup. Possible reasons may be: SCTP is not running, or an incorrect network appearance. Check the mlog file for the failure reason.
020303	INFO	CLR_ALARM	ASP: SPMC for network appearance %d is up	Indicates that SPMC for a network appearance is up. No action is required.
020304	INFO	SET_ALARM	ASP: SPMC for network appearance %d is down	Indicates that SPMC for a network appearance is down. No action is required.
020305	INFO	EVENT	ASP: Connection with SGP %s is up	Indicates that connection with an ASP is up. No action is required.
020306	INFO	EVENT	ASP: Connection with SGP %s is down	Indicates that connection with SGP is down. Verify that the connection did not go down due to local problem such as cable failure, incorrect IP, or congestion at the SCTP side.
020307	INFO	EVENT	ASP: Connection with SGP %s is congested at level 1	Indicates that the connection between SG and an ASP is congested at level 1.
020308	INFO	EVENT	ASP: Connection with SGP %s is congested at level 2	Indicates that the connection between SG and an ASP is congested at level 2.
020309	INFO	EVENT	ASP: Connection with SGP %s is congested at level 3	Indicates that the connection between SG and an ASP is congested at level 3.
02030a	INFO	SET_ALARM	ASP: ASP %s is down	Indicates that an ASP is down. No action is required.
02030b	INFO	CLR_ALARM	ASP: ASP %s is inactive	Indicates that an ASP is inactive. No action is required.
02030c	INFO	EVENT	ASP: ASP %s for AS %s is inactive	Indicates that an ASP is inactive for an AS. No action is required.
02030d	INFO	EVENT	ASP: ASP %s for AS %s is active	Indicates that an ASP is active for an AS. No action is required.
02030e	INFO	SET_ALARM	ASP: AS %s is down	Indicates that an AS is down. No action is required.
02030f	INFO	CLR_ALARM	ASP: AS %s is inactive	Indicates that an AS is inactive. No action is required.
020310	INFO	EVENT	ASP: AS %s is active	Indicates that an AS is active. No action is required.
020311	INFO	EVENT	ASP: AS %s is pending	Indicates that an AS is pending before going down. No action is required.

Table 10-3: M3UA Alarm Group (Continued)

Alarm No.	Severity	Type	Message	Operation
020312	INFO	EVENT	ASP: No more AS pending buffer, messages may be discarded	Indicates that there is no more pending buffer available to store messages for a particular AS while it is in pending mode, and that messages will be discarded. No action is required.
020313	INFO	SET_ALARM	ASP: ASP %s inhibited	Indicates that an ASP is inhibited, i.e. its traffic is temporarily suspended.
020314	INFO	EVENT	ASP: M3UA Error Indication (error code %d - %s)	Indicates an M3UA error condition. Check the alarm message for the error reason. Call TAC if it is unsolvable.

Table 10-4: M3UA Error Group

Alarm No.	Severity	Type	Message	Operation
020401	INFO	EVENT	ASP: %s M3ua error - invalid version (error code %d)	Contact TAC.
020402	INFO	EVENT	ASP: %s M3ua error - unsupported message class (error code %d)	Contact TAC.
020403	INFO	EVENT	ASP: %s M3ua error - unsupported message type (error code %d)	Contact TAC.
020404	INFO	EVENT	ASP: %s M3ua error - invalid traffic mode (error code %d)	Make sure traffic mode configured is correctly.
020405	INFO	EVENT	ASP: %s M3ua error - unexpected message (error code %d)	Contact TAC.
020406	INFO	EVENT	ASP: %s M3ua error - protocol error (error code %d)	Contact TAC.
020407	INFO	EVENT	ASP: %s M3ua error - invalid stream ID (error code %d)	Contact TAC.
020408	INFO	EVENT	ASP: %s M3ua error - management blocking (error code %d)	Contact TAC.
020409	INFO	EVENT	ASP: %s M3ua error - ASP ID needed (error code %d)	Contact TAC.
02040a	INFO	EVENT	ASP: %s M3ua error - invalid ASP ID (error code %d)	Contact TAC.
02040b	INFO	EVENT	ASP: %s M3ua error - invalid parameter value (error code %d)	Contact TAC.
02040c	INFO	EVENT	ASP: %s M3ua error - field error (error code %d)	Contact TAC.
02040d	INFO	EVENT	ASP: %s M3ua error - unexpected parameter (error code %d)	Contact TAC.
02040e	INFO	EVENT	ASP: %s M3ua error - unknown destination status (error code %d)	Contact TAC.

Table 10-4: M3UA Error Group (Continued)

Alarm No.	Severity	Type	Message	Operation
02040f	INFO	EVENT	ASP: %s M3ua error - invalid network appearance (error code %d)	Make sure the network appearance configured is correctly.
020410	INFO	EVENT	ASP: %s M3ua error - missing parameter (error code %d)	Contact TAC.
020411	INFO	EVENT	ASP: %s M3ua error - invalid RC (error code %d)	Make sure RC configured is correctly.
020412	INFO	EVENT	ASP: %s M3ua error - AS unconfigured (error code %d)	Make sure AS is configured.
020413	INFO	EVENT	ASP: %s M3ua error - unknown error (error code %d)	Contact TAC.

10.3.2 Log Files

The Master Log file can be checked to help identify a problemwhen troubleshooting.

10.3.3 Runtime Tracing

The Signaling Gateway Client system supports runtime tracing in which trace messages are directed to a circular buffer in the shared memory. Tracing can be enabled or disabled dynamically during operation of the system. To utilize the tracing utility, the *program ID* of the process to be traced must be given.

For example, to enable tracing for process 1ØØ:

apm_trsetmask -ma -p 1ØØ

To display traces collected for process 1ØØ:

apm_trshow -p 1Ø

To clear the trace buffer:

apm_trclear

To disable tracing for process 1ØØ:

apm_trsetmask -na -p 1Ø

The following table lists the program ID of the application processes:

Table 10-5: Program IDs for Tracing

Process Name	Program ID
mlogd	1
spmd	2
netd	3
alarmd	4
dsmd	5
dkmd	6
upmd	7
scmd	8
aspd	1Ø1

Caution: *Tracing should not be enabled during normal operation of the system as it may significantly degrade the performance. Be sure to disable the traces when you are done troubleshooting!*

Alternatively, use the sgc_trace command to simplify tracing the SCTP and ASP processes, as shown in the following examples:

Enter the following to disable tracing for ASP:

sgc_trace -u asp Ret

Enter the following to display traces collected for ASP:

sgc_trace asp Ret

Enter the following to clear all traces in the buffer:

sgc_trace -c Ret

Enter the following to enable tracing for SCTP:

sgc_trace -s ip Ret

Appendix A: Glossary

The following table lists Signaling Gateway Client abbreviations, and common SS7 and telecommunications industry acronyms. Brief definitions are included for frequently used terms found in the Signaling Gateway Client manuals.

Acronymn or Term	Definition
AS	Application Server is a logical entity serving a specific routing key, such as a CIC range or a particular Destination Point Code.
ASP	Application Server Process is a process instance of an Application Server. An ASP can be active or standby for an AS.
DPC	Destination Point Code is the address of an SS7 destination node.
failover	A backup operation that automatically switches to a standby database, server or network if the primary system fails or is temporarily shut down for servicing. Failover is an important fault tolerance function of mission-critical systems that rely on constant accessibility. Failover automatically and transparently to the user redirects requests from the failed or down system to the backup system that mimics the operations of the primary system.
IETF	Internet Engineering Task Force is a large open international community concerned with the evolution of the Internet architecture and its smooth operation.
IGP	Interior Gateway Protocol. A protocol that distributes routing information to the routers within an autonomous system. The term "gateway" is historical, as "router" is currently the preferred term.
IP	Internet Protocol is the part of the TCP/IP protocol family dealing with addressing and routing.
IPAS	An IP-based application server. An IPAS is essentially the same as an AS, except that it uses IPSPs instead of ASPs.
IPSP	A process instance of an IP-based application. An IPSP is essentially the same as an ASP, except that it uses M3UA in a point-to-point fashion. Conceptually, an IPSP does not use the services of a Signalling Gateway node.
IS-IS	Intermediate System-Intermediate System. The OSI IGP, Open Systems Interconnection, Interior Gateway Protocol
ISUP	ISDN User Part is an SS7 protocol layer used to set up and tear down calls.
M3UA	MTP3 User Adaptation enables seamless operation of MTP3 user peers in SS7 and IP domains.
MTP	Message Transfer Part provides message handling functions that transfer the signaling messages to the proper signaling link or the user part and network management functions that control the current message routing and configuration of the signaling network.
MML	Man-Machine Language is a human-readable syntax based on text lines for controlling and obtaining status from a network element.
multihoming	Addressing scheme in IS-IS routing that supports assignment of multiple area addresses. It is defined as the separate physical network interfaces of the ASP and the cluster interfaces, each with its own IP address.
NEBS	Network Equipment-Building System is a standard for product safety.
NIF	Nodal Interworking Function is a module that interworks between the SS7 and the SIGTRAN stacks in the SG. It translates SS7 MTP3 messages to M3UA messages, and vice versa.

Acronymn or Term	Definition
OPC	Origination Point Code is the originating node's address in the SS7 signaling messages.
PSTN	Public Switched Telephone Network is the traditional telephone system, using DS0 format to carry digitized modem or voice traffic.
SCCP	Signaling Connection Control Part provides additional functions to MTP to provide connectionless and connection-oriented network services on a node-to-node basis.
SCN	Switched Circuit Network is the traditional telephone network that uses DS0 format to carry digitized modem or voice traffic.
SCTP	Stream Control Transmission Protocol is an IP based transmission protocol designed to carry SS7 signaling.
SIGTRAN	Signaling Transport is name of a work group in IETF that designed the architecture for the transport of SS7 signaling over IP.
SIO	Service Indicator Octet determines the MTP user, such as. ISUP, TUP, and SCCP.
SG	Signaling Gateway is a device that allows SS7 network level services to IP based applications.
SGP	Signaling Gateway Process is a process instance of a Signaling Gateway.
SLS	Signaling Link Selection is used by the MTP users to provide guidance to the MTP layer on signaling link selection.
SNMP	Simple Network Management Protocol is a message-oriented protocol for the control and interrogation of network elements.
SP	Signaling Point is an SS7 network node such as an end office or tandem equipped with signaling link hardware and software.
SPMC	Signaling Point Management Cluster is the complete set of Application Servers represented to the SS7 network under one specific SS7 point code of one specific network appearance.
SSN	Subsystem Number identifies the user of the SCCP service layer.
SS7	Signaling System #7 is the protocol used to transport signaling for the PSTN.
TCP	Transmission Control Protocol is the middle level of a protocol commonly used over IP to transport packet traffic.
TCAP	Transaction Capabilities Application Part provides a means to establish non-circuit related communication between two SS7 nodes.
UDP	User Datagram Protocol is a middle-level protocol used to transport large, fixed-length data blocks in a connectionless manner.

Index

Numerics
1 93, 145, 220
A
access transparency 80, 84, 88, 91, 94
Access Types (attributes) 171
AccessMOB 271
  using 164
AccessMonitor 279
AccessOMAP 272
AccessSNMP 274
AccessStatus 277, 473
Add Operation (GUI) 177
alarm
  reports 370
Alarm process, starting 269
alarmd 269
Alert Message 2
ANSI point code format 188
apm 305
APM Utilities 388
apm_getstate 389
apm_kill 390
apm_killall 392
apm_ps 394
apm_report 397
apm_setstate 399
apm_start 401
apm_stop 403
apm_trcapture 405
apm_trclear 408
apm_trgetmask 409
apm_trinit 411
apm_trsetmask 413
apm_trshow 415
apm_update 417
apmconfig 305, 318
apmconfig old 318
apmd 280
Application Manager 29
Application Programming Guides 32
Architecture 24, 34
AS, SIGTRAN definition 10
ASP, SIGTRAN definition 10
Attributes
  access type 171
  data types 170
  key 170
  Managed Object 56
  Managed Objects 170
audit distributed shared memory 419
B
Backward Compatibility 32
Basic Platform Process 116
building blocks 23
C
Case Sensitivity, MML 189
changing GUI window name 174
Characteristics, Distributed 68
CIC 146
circuit
  0 147
  group ID 146
  number and CIC 147
Circuit Display Report
  ISUP 222
Circuit Group Display Report, ISUP 225, 226
Circuit Status Definitions 223
circuits 147
CNFG 51
Command-Line
  Syntax, Rules 189
Commands
  User 325
Concerned SP
  Provisioning 144
concurrence transparency 80, 85, 88, 92, 94
Configuration 271
  ISUP 145
Configuration Management MML 30
CONNECTION
  DISPLAY 219
Controller Options 48
Conventions, Notations 2
Core Capabilities 26
Core Product Specifications 75
country variants 46
CPC
  ADD 204
  DELETE 204
  DISPLAY 204

Index

Creation 303
CRP 41
Customer Routing Point 41
D
data
  entering 168
  string-constant entry method 189
Data Types (attributes) 170
Database Capacity 47
D-channel 147
Delete Operation (GUI) 179
Dependencies 49
  APIs 50
development platform 23
Dialog Box 168
Disk Space Requirements 99
Display
  ISUP Circuit Report 222
Distributed Alarm 31
Distributed Kernel Memory Management 29
Distributed Operations 26
Distributed Process Management 28
Distributed Shared Memory
  audit 419
  information, display 422
  Management 29
  manager 267, 285
  segment 426
Distributed System Characteristics 68
DKM 29
  Utilities 428
dkm_apidemo 428
dkm_dump 430
dkm_list 431
dkm_rm 436
dkm_sar 437
dkm_stat 439
dkmd 283
DSM
  display 422
  management 29
  segment info 426
  Utilities 418
dsm_apidemo 418
dsm_audit 419
dsm_bm 421
dsm_list 422
dsm_rm 425
dsm_stat 426
dsmd 267, 285
Dynamic Trace 29
E
ebs_apidemo 331
ebs_audit 332
ebs_dbconfig 339
ebs_hbeat 344
ebs_ipcbm 346
ebs_log 347
ebs_loopback 349
ebs_modinstall 354
ebs_modremove 355
ebs_oldapidemo 353
ebs_pkgrm 358
ebs_ps 359
ebs_qlist 365
ebs_qstat 367
ebs_quinfo 363
ebs_report 370
ebs_showlink 373
ebs_shutdown 376
ebs_start 377
ebs_stop 378
ebs_sync 379
ebs_sysinfo 381
ebs_tune 383
Environment Variables 165
Error Log 29
error messages
  MML 191
  MOB 183
Errors from GUI 183
Errors from MO Servers 184
Event Management 27, 31
F
Failure Semantics 77, 83, 88, 91, 94
failure transparency 81, 85, 89, 92, 95
Fault Tolerance 71, 77, 83, 87, 90, 93
Features 24
file, sample SCTP.conf 126
Flexibility 90, 93

Index

Format, point code 188
G
Global Title
  Outgoing Point Code Provisioning 144
Graphical User Interface 31, 164, 271
GT
  ADD 206
  DELETE 206
  DISPLAY 206
GTENTRY
  ADD 208
  DELETE 208
  DISPLAY 208
GUI 30, 31, 164
  Main Window 173
  modes 174
  Requirements 164
  starting 166
  stopping 166
H
High Availability 71, 76, 82, 87, 90, 93
HOST
  MML command 237
I
IETF 10
INE 33
Installation
  Sun 97
installation steps 100
installation, software 98
installing devices 354
Intelligent Network Emulation 33
Inter-Process Communications 23, 27
IPC 23, 27
ISDN User Part 25
ISUP 9, 25
  Circuit Display Report 222
  Circuit Group Display Report 225, 226
  Circuit Groups, configure 146
  Circuits 147
  configuration 145
  DISPLAY 230
  MODIFY 230
  module 36
  Node, configure 145
ISUPCCT
  ADD 220
  DELETE 220
  DISPLAY 220
  MODIFY 220
isupd 287
ISUPGRP
  ADD 224
  DELETE 224
  DISPLAY 224
  MODIFY 224
ISUPNODE
  ADD 227
  DELETE 227
  DISPLAY 227
  MODIFY 227
ISUPTMR
  DISPLAY 233
  MODIFY 233
ITU point code format 188
K
Key Attributes 170
L
Labels
  MML Network Element 188
LAN Configuration, changing 135
Libraries
  APM 36
  DSM 36
  Gateway API 37
  GSM MAP API 38
  IS41-D API 37
  ISUP 37
  OAM 36
  Raw TCAP 37
  SPM 36
  TCAP 37
Library 36, 37
license file 97
link status 277, 473
Links 7
LOCALSUBSYS
  DISPLAY 218
location transparency 80, 85, 88, 91, 94
logd 289

Index

logging 28, 347
loopback 28, 349
M
M3UA 10
Man Machine Language 31
Managed Object 30, 51
    Attributes 56, 170
    Browser 167
        Error Messages 183
        Operation 173
        Windows 173
    Server 51
        Error Messages 184
Managed Object Browser 271
MATE
    ADD 210
    DELETE 210
    DISPLAY 210
Mated Subsystem
    Provisioning 145
Media Server 39
Menus (GUI) 167
Message
    Alert 2
message discrimination 8
message distribution 8
Message Transfer Part 25
Messages, output 191
Messaging 27
mlogd 290
MMI 293
MML 31, 292
    case sensitivity 189
    Commands
        SCCP 204
    Network Element Labels 188
MOB 167, 271
    error messages 183
    Managed Objects, selecting 175
    Operation Mode, selecting 174
    Operation, dialog boxes 176
Modes
    GUI 174
Modify Operation (GUI) 178
mouse actions (valid) 168
MTP 8, 25
    Layer Product Specifications 82
    Level 1,2 35
    Level 3 35
multihoming 16
N
named object 27
netd 294
Network (NTWK) 239
Network Appearance 11
Network Clock Synchronization 32
network connection 294
Node creation 303
Node Management 30
Notations, Conventions 2
NTWK
    MML command 239
O
Object Server 30
OMAP 25
    startup 272
Operation
    MOB 173
Operations
    Maintenance and Application Part 25
Output Messages 191
P
Performance 75
Platform
    Options 47
    Services 26
    Utilities 329
Point Codes, format 188
Process Management 35
process status 359
Product Specifications 82
Provisioning
    Concerned SP 144
    Mated Subsystem 145
PSTN 5
R
READ-CREATE access 172
READ-ONLY access 171
READ-WRITE access 171
Redundant LAN Support 32

Index

Registration 27
Related Documents, References 3
Reliability 76, 82, 87, 90, 93
removing devices 355
replication transparency 81, 85, 89, 92, 95
Report
    ISUP Circuit Display 222
    ISUP Circuit Group Display 225
Report, ISUP Circuit Group Display 226
reports
    alarm 370
Resource Sharing 69, 75, 82, 87
Rules for Command-Line Syntax 189
S
scalability 73, 79, 84, 88, 91, 94
SCCP 9, 25
    initialization 298
    MML Commands 204
    module 35
    Product Specifications 87
scmd 298
SCN 5
SCP 7
SCTP 10
SCTP Association, SIGTRAN definition 11
SCTP multihoming 16
SCTP, sample conf file 126
Selecting
    managed objects 175
    operation mode 174
SEP 7
service endpoint 27
Service Provider Module 35
set a new title for windows 174
Settings, environment variables 165
SG Process 116
SG, SIGTRAN definition 10
sg_setrelease 100
SGCRK
    MML command 249
SGCRKRNG
    MML command 251
SGCSG
    MML command 253
SGCSGP
    MML command 255
SGP 10
SGP, SIGTRAN definition 10
shared memory, distributed 267, 285
Signaling Connection Control Part 25
SIGTRAN 10
Simple Network Management Protocol 31
SMH 83, 86
SNM 361
SNMP 31, 274
SNSP
    ADD 214
    DELETE 214
    DISPLAY 214
software installation 98
Solaris patches 100
SP 6
SPC 6
Specifications 75, 82
    MTP 82
    SCCP 87
    TCAP 90
SPM 35
spmd 300
SS7 5
    Controller 35
    object 27
SS7 Node Process 116
SSP 6
Standards Compliance 46
Starting AccessMANAGER 377
Starting the Managed Object Browser 166
Status
    display 277, 473
    process 359
Stop distributed AccessMANAGER 376
Stopping AccessMANAGER 378
Stopping the GUI 166
STP 6
Streams multiplexors 361, 365
String-Constant, data entry method 189
SUBSYS
    ADD 212, 216
    DELETE 212, 216
    DISPLAY 212, 216

Index

Subsystem
  Mated Provisioning 145
  Provisioning 144
Syntax, Command-Line 189
System
  Applications 39
T
TCAP 9, 25
  module 36
  multiplexor 302
  Product Specifications 90
  starting 302
  Utilities 448
tcm_apidemo 451
tcm_list 452
tcm_stat 454
tcm_tune 455
tcmd 302
timer facilities 23
Timer Services 27
Tracing 29
Transaction Capabilities Application Part 25
transparency 74, 80, 84, 88, 91, 94
  access 74, 80, 84, 88, 91, 94
  concurrence 74, 80, 85, 88, 92, 94
  failure 74, 81, 85, 89, 92, 95
  growth/retrofit 74
  location 74, 80, 85, 88, 91, 94
  replication 74, 81, 85, 89, 92, 95
  scaling 74
Transport Security 17
trunk group ID 146
U
UPM 361
UPM, User Part Multiplexor 303
upmd 303
User Access 17
User Commands 325
Using
  AccessMOB 164
Utilities
  APM 388
  DKM 428
  DSM 418
  platform 329
  TCAP 448
  virtual board 457
V
vb_addhost 457
vb_bridge 458
vb_config 460
vb_connhosts 462
vb_connports 463
vb_discport 464
vb_lhosts 465
vb_lports 466
vb_reset 467
vb_startup 468
View Operation (GUI) 180
Virtual Board Utilities 457
W
Wildcards 171
window manager functions 167
window name, changing 174
Windows
  MOB 173
WRITE-ONLY access 172