Troubleshooting Cisco IP Telephony

Reveals the methodology you need to resolve complex problems in an IP telephony network

Paul Giralt, CCIE® No. 4793
Addis Hallmark
Anne Smith

ciscopress.com
Active Directory domain name problems 841
Active Directory integration troubleshooting and overview 837
Active Directory schema modifications 840
Active Directory—users added in Active Directory don’t show up in CallManager Administration 843
Adding a user fails 824
Alarms (red and yellow) on a digital interface 208
“Already in conference” message 597
Attendant Console client configuration 781
Attendant Console—fast busy on calls to a pilot point 782
Attendant Console—line states won’t update 782
Attendant Console—lines are disabled 782
Attendant Console—login failed 782
Attendant Console—longest idle algorithm is not working properly 782
Attendant Console—new user doesn’t display 782
Attendant Console server configuration 780
Attendant Console—some line states show Unknown status 782
Attendant Console troubleshooting methodology 782
Attendant Console—wrong directory list displays 782
Audio problems 384, 389, 396, 400, 402, 405, 410
Audio Translator problems 617
Automated alternate routing (AAR) troubleshooting 637
Busy signal not heard on an IP phone 310
Calling name display problems 270
Calling search spaces, overview 469
CallManager Serviceability 82
CallManager wildcard summary 460
Call preservation, overview 551
Call routing problems 515
CCM traces—how to read 42
CCM traces—how to read Skinny messages 148
CDRs are not being written properly 813
CDRs are not generated by Subscriber 810
Choppy audio 396
“CM Down, Features Disabled” message 158
“CM Fallback Service Operating” message 707
CMI—reading traces 674
Codec selection between devices 570
Conference bridge—out of resources 578
Conference Connection doesn’t work 789
Corporate directory—add or delete users fails in CallManager Administration 843
Corporate directory—Error: “The phone administrator is currently not allowed to add or delete users” 843
CRA Administration page does not load 738
CRA Application Engine problems 745
CRA directory configuration troubleshooting 741
CRA trace files (MIVR) 748
Customer Directory Configuration Plugin troubleshooting 839, 842
Database Layer Monitor is not running properly 812
Database replication problems 796, 804, 807
D-channel won’t establish on PRI 210
DC Directory—reconfiguring in CallManager 3.3 828, 835
DC Directory—reconfiguring in pre-3.3 CallManager 830, 835
Directory access troubleshooting 823
Directory troubleshooting 824
Delayed audio 384
Delayed routing 466
Dial peer matching in IOS, overview 175
Dick Tracy 101
Digit discard instructions (DDIs), overview 486
directories button doesn’t work 160
Disconnected calls with cause code 0xE6, “Recovery on timer expiry.” 271
DPA 7610/7630—MWI problems 702
Dropped calls 157
Dropped packets 397
DTMF relay, overview 303
E1 interface troubleshooting 208
Echo problems 410, 418, 421, 428
“Exceeds maximum parties” message 597
Extension mobility—common error messages 765, 777
Extension mobility problems on CallManager release 3.1 or 3.2 756
Extension mobility problems on CallManager release 3.3 773
Extension mobility troubleshooting methodology for CallManager release 3.1 or 3.2 765
Failover—phone behavior and causes 155, 158
Fax machine troubleshooting methodology 446, 447, 449
Fax passthrough configuration 441
Fax passthrough, overview 437
Fax relay debugs, enabling 455
Fax relay, overview 444
Fax takes twice as long to complete 451
FXO port will not disconnect a completed call 205
Garbled audio 396
Gatekeeper call admission control 638, 640
H.323 call flow (H.225 and H.245) 283, 284, 287
Hold and resume problems 522
Hold doesn’t play music 611
Intercluster trunk troubleshooting 311
IOS gateway—call routing and dial peer debugs 196
IOS gateway debugs 185
IOS gateway—debugs and show commands 184
IOS gateway—diagnosing the state of ports 187
IOS gateway—TDM interfaces 187
IOS gateway won’t register with CallManager (MGCP) 240
iPlanet integration troubleshooting 844
ISDN cause codes (Q.850) 262
ISDN messages, overview 258
ISDN timers, overview 271
Jitter 389, 400
Live audio source problems 619
LMHOSTS file, overview 796
Locations-based call admission control 624, 636
Masks, overview 495
Methodology for troubleshooting 4
MGCP overview 218
Microsoft Performance (PerfMon) 68
Modem passthrough configuration 441
Modem passthrough, overview 437
Modem troubleshooting methodology 447
MOH—live audio source problems 619
MOH—multicast and unicast problems 615
MOH—no music when calls are on hold 611, 617
MOH—reading CCM traces 607
MOH—troubleshooting methodology 611
MWI problems (Personal Assistant) 659
MWI problems (SMDI) 682
MWI problems (Unity) 659
MWI problems (VG248) 690
“No conference bridge available” message 587
No-way audio 405, 410
Octel integration 693
One-way audio 405, 406, 410
Outside dial tone played at the wrong time 465
Park problems 531
Partitions, overview 469
Personal Assistant is not intercepting calls 785
Personal Assistant—MWI problems 659
Phone—busy signal not heard 310
Phone—failover and failback 154, 155, 158
Phone—inline power problems 114
Phone—network connectivity and Skinny registration 117
Phone stuck in SRST mode 730
Phone—switch port operation 161
Phone—TFTP configuration file 121
Phone—understanding the difference between restart and reset 156
Phone—understanding the Skinny protocol 139
Phone—VLAN configuration 118
Phone won’t register 127
Pickup/group pickup problems 533
PRI backhaul channel status 256
PRI—CallManager sends the proper digits to the PSTN, but call won’t route properly 269
PRI signaling troubleshooting 210, 262
Publisher-Subscriber model, overview 793, 796
Q.931 Translator 95
Registration problems on IP phone 127
Replication problems 796, 804, 807
Reset vs. restart 156
Ringback problems 307, 309
Route filters, overview 506
SDL traces—how to read 60
Search for a user fails 824
services button doesn’t work 160
Silence suppression—effect on voice quality 402
SMDI—check configuration parameters 686
SMDI—integration 662
SMDI integration with VG248 686
SMDI—MWI problems 682
SoftPhone has no lines 788
SoftPhone—one-way audio over VPN 787
SoftPhone shows line but won’t go off-hook 786
SQL database replication problems 796, 804, 807
SQL—re-establishing a broken subscription 807
SQL—reinitializing a subscription 809
SRST and phone registration 712
SRST—DHCP issues 731
SRST—features lost during operation 707
SRST—phones still registered after WAN connection is restored 730
SRST—transfer problems 729
SRST—voice mail and forwarding issues 731
T1 CAS signaling troubleshooting 214
T1 interface troubleshooting 208
“Temporary Failure” message 561
Time synchronization 38
Toll fraud prevention 544, 548
Tone on hold plays instead of music 615, 617
Transcoder—out of resources 578
Transcoder—understanding codec selection between devices 570
Transfer problems 529
Transformation troubleshooting 513
Transformations and masks, overview 486
Transformations, overview 496, 500
Translation pattern troubleshooting 501
Unity—MWI problems 659
Unity—TSP configuration 656
VAD—effect on voice quality 402
VG248—MWI problems 690
Voice mail—MWI problems (DPA 7610/7630) 702
Voice mail—MWI problems (SMDI) 682
Voice mail—MWI problems (Unity) 659
Voice mail—MWI problems (VG248) 690
Voice mail—Octel integration 693
Voice quality problems 384, 389, 396, 400, 402, 405, 410
WS-X6608/6624 gateway troubleshooting 313, 314, 320, 324
WS-X6608—D-channel is down 326, 337, 340, 343, 344
WS-X6608—dropped calls 326
WS-X6608—T1 CAS problems 359
WS-X6608 T1/E1 configuration troubleshooting 325, 326
WS-X6608—unexpected resets 326, 345
WS-X6624 FXS analog gateway configuration 367
Warning and Disclaimer
This book is designed to provide information about troubleshooting the various components of a Cisco IP Telephony network. Every effort has been made to make this book as complete and accurate as possible, but no warranty or fitness is implied.

The information is provided on an “as is” basis. The authors, Cisco Press, and Cisco Systems, Inc., shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the discs or programs that may accompany it or be referenced by it.

The opinions expressed in this book belong to the authors and are not necessarily those of Cisco Systems, Inc.

 Portions of Chapter 6 are extracted from RFC 2705 which defines MGCP. The following copyright statement applies to any information derived from RFC 2705:

Full Copyright Statement
Copyright © The Internet Society (1999). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an “AS IS” basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Feedback Information
At Cisco Press, our goal is to create in-depth technical books of the highest quality and value. Each book is crafted with care and precision, undergoing rigorous development that involves the unique expertise of members of the professional technical community.

Reader feedback is a natural continuation of this process. If you have any comments regarding how we could improve the quality of this book, or otherwise alter it to better suit your needs, you can contact us through e-mail at feedback@ciscopress.com. Please be sure to include the book title and ISBN in your message.

We greatly appreciate your assistance.
About the Authors

**Paul Giralt**, CCIE No. 4793, is an escalation engineer at the Cisco Systems Technical Assistance Center in Research Triangle Park, N.C., where he has worked since 1998. He has been troubleshooting complex IP Telephony networks since the release of CallManager 3.0 as a TAC engineer, a technical lead for the Enterprise Voice team, and now as an escalation engineer supporting the complete Cisco line of IP Telephony products. Paul has troubleshooted problems in some of Cisco’s largest IP Telephony deployments and has provided training for TAC teams around the globe. Prior to working on IP Telephony, he was a TAC engineer on the LAN Switching team. He holds a B.S. in computer engineering from the University of Miami.

**Addis Hallmark**, CCNA, CIPT, is a senior technical marketing engineer with Cisco Systems. He has been installing, configuring, administering, and troubleshooting the Cisco IP Telephony solution since the 2.3 release of CallManager. He has contributed to numerous design guides, application notes, and white papers on a variety of IP Telephony subjects, including CallManager, IP Phones, and IP gateways.

**Anne Smith** is a technical writer in the CallManager engineering group at Cisco Systems. She has written technical documentation for the Cisco IP Telephony solution since CallManager release 2.0 and was part of the Selsius Systems acquisition in 1998. Anne writes internal and external documents for CallManager, IP phones, and other Cisco IP Telephony products. She is a co-author of *Cisco CallManager Fundamentals* (ISBN: 1-58705-008-0) and *Developing Cisco IP Phone Services* (ISBN: 1-58705-060-9), both from Cisco Press.
About the Technical Reviewers

Shawn Armstrong is an IT engineer working in Cisco’s Core Hosting group. She has been with Cisco for four years and is responsible for managing NT and Windows 2000 servers within Cisco’s Information Technology group.

Dave Goodwin, CCIE No. 4992, is a customer diagnostic engineer for Cisco’s Advanced Engineering Services. He is responsible for discovering and resolving problems in new Cisco IP Telephony products while administering internal field trials for these systems. He also works closely with Cisco’s development and TAC support teams to provide support for anything from troubleshooting to quality issues to tools. He has been at Cisco for almost five years and has worked as a network engineer for eight years.

Christina Hattingh is a member of the Technical Marketing organization at Cisco Systems. In this role she works closely with product management and engineering. Christina focuses on helping Cisco sales engineers, partners, and customers design and tune enterprise and service provider Voice over Packet network infrastructures with particular focus on QoS. Prior to this she was a software engineer and engineering manager of PBX Call Center products at Nortel Networks. Her earlier software development experience in X.25 and network management systems provide background for the issues involved today in migrating customers’ traditional data and voice networks to packet-based technologies. Christina has a graduate degree in computer science and mathematical statistics.

Phil Jensen, CCIE No. 2065, is a consulting systems engineer for Cisco in the southeastern U.S. He has focused on helping Cisco’s largest customers design and troubleshoot AVVID IP Telephony solutions for the past three years. He has worked as a network engineer for more than 14 years.

Ketil Johansen, CCIE No. 1145, is a business development manager with Cisco Systems, working with companies integrating their applications with Cisco CallManager. He has worked with networking technologies for more than 18 years and has been a CCIE since 1994. The last three years he has focused on IP Telephony technologies.

Chris Pearce is a technical leader in the Cisco CallManager software group at Cisco Systems, Inc. He has ten years of experience in telecommunications. His primary areas of expertise include call routing, call control, and telephone features. He was a member of the team that developed and implemented the Cisco CallManager software from its early stages, and he was directly involved in developing the system architecture and design.

Ana Rivas, CCIE No. 3877, is an escalation engineer in Cisco’s EMEA region. She is one of the technical leaders for AVVID solutions in the Cisco TAC. She is responsible for technically leading the resolution of some of the most critical problems in voice and IP Telephony, spreading technical knowledge to other teams, and working with Cisco business units and the field to head IP Telephony solutions. She has been working as a network engineer for more than five years.

Markus Schneider, CCIE No. 2863, is a diagnostic engineer for Cisco’s Advanced Engineering Services. He is responsible for helping Cisco customers design, implement, and troubleshoot IP Telephony solutions in their environment. He has been working for Cisco as a network engineer for more than six years.

Gert Vanderstraeten has been working as a telecom/datacom engineer for companies such as Alcatel, Bell, and Lucent Technologies since 1993. Since 1998 he has been an independent contractor for the Cisco Systems’ IT department. During the course of his tenure, his main focus has been the design, implementation, and maintenance of VoIP, IP Telephony, voice and video applications, and the integration of AVVID technologies into solutions. He is currently operating within the Cisco Systems global Enterprise Architecture Solutions team.

Liang Wu is a software engineer in the CallManager software group at Cisco Systems, Inc. For the last seven years, he has been focusing on PBX/Enterprise communication systems. He spent more than eight years in the Class 4/5/AIN telephone switching industry.
Dedications

Paul Giralt
I dedicate this book to my parents, Vicia and Pedro, for being the best parents anyone could ask for and always providing the opportunity and encouragement to continue learning.

Addis Hallmark
I want to dedicate this book to my lovely wife, Stephanie. Her companionship is the most precious thing in the world to me. Her patience, understanding, and encouragement helped me write this book. I love and appreciate her dearly.

Anne Smith
For Herb for seeing me through the long nights and weekends without complaint. And, of course, for all those backrubs.
Acknowledgments

Paul Giralt

I want to first thank Anne Smith for all her hard work and guidance throughout this entire project. There is no way this book would exist without her constant dedication and attention to detail.

Thanks to Chris Cleveland for his excellent work as development editor on this book and for being so flexible when it comes to the unpredictable schedules of a TAC engineer.

Thank you to the worldwide Enterprise Voice and AVVID TAC teams, especially the RTP Enterprise Voice team for being such a world-class group of engineers to work with.

Thanks to the RTP Voice Network Team (VNT) for all the excellent VoX documentation. Special thanks to Gonzalo Salgueiro and Mike Whitley for the VoX boot camp material and to Steve Penna for knowing everything.

Thanks to Dave Hanes for his excellent fax troubleshooting presentations and Andy Pepperell for his explanation of fax and modem passthrough.

Thanks to all the technical reviewers—Ana Rivas, Chris Pearce, Dave Goodwin, Ketil Johansen, Markus Schneider, Phil Jensen, Gert Vanderstraeten, Liang Wu, Shawn Armstrong, and especially Christina Hattingh—for always being on top of everything in the world of Cisco IOS gateways.

Thanks to all the developers in Richardson and San Jose that I have worked with over the years. Your insight into the inner workings of CallManager has helped me understand how to better troubleshoot the product. Special thanks to Bill Benninghoff for always answering any question I throw his way and for always being so thorough in his explanations. Also thanks to Chris Pearce for his excellent grasp on the intricacies of call routing.

Thank you to all the contributors to the VNT Voice University website as well as the AVVID TAC tips website on Cisco.com. Also thanks to all the other unnamed authors for the documentation scattered throughout various web pages.

Thanks to all the customers I have worked with over the past several years on AVVID issues for being my teachers. Every customer I work with helps me understand a little more about IP Telephony.

Addis Hallmark

First, I’d like to thank Paul Hahn and Richard Platt for bringing me on at Cisco. Paul in particular spent a lot of time with me, bringing me up to speed on these technologies, and for that, I am indebted to him.

I’d also like to thank all the brilliant development engineers who patiently helped me understand CallManager so well over the past few years.

I’d like to thank Susan Sauter. She is a brilliant engineer, and so much of what I know about IP Phones came from her patient instruction.

Chris Pearce has also helped me so much over the last few years in understanding dial plans.

The chapter on applications is based on the hard work of Dave Bicknell. Without his efforts, that chapter would not be even close to what it should be.

Manish Gupta and his team were a tremendous source of help on the LDAP Directory chapter. Stefano Giorcelli’s excellent directory documentation also was so very helpful!

The TAC is on the front lines of troubleshooting, and much of the help I received was from the experiences that only solid TAC engineers could provide.

Also, the technical reviewers of this book were so helpful. Thank you so much to everyone for their hard work!

I really believe this is a great book, and one of the biggest reasons for that is Paul Giralt’s invaluable contribution and hard work on this project. I couldn’t have done this without him!
My manager, Shaik Kaleem, was very supportive of this project that I undertook on my own time, and I greatly appreciate that support.

Finally, I’d like to thank Anne Smith. This project would never have happened without her tireless work and skillful help. I am so grateful for Anne’s effort. She worked so very hard over this past year, and Paul Giralt and I would have been lost without her.

Anne Smith

My many thanks go to Paul Giralt and Addis Hallmark for making this book a reality with their knowledge, experience, hard work, and sacrifice. In particular, I thank Paul for a highly enjoyable working experience. Paul’s dedication to the quality, accuracy, and comprehensiveness of this book was unsurpassed; he spent countless hours reviewing every page of technical information and his experience with the many components in the Cisco AVVID IP Telephony solution made his extensive contribution invaluable. At every turn, Paul’s dedication, commitment to quality, tireless drive for accuracy, and constant positive attitude made working with him a rewarding experience.

As always, my thanks and great admiration go to Richard Platt and Scott Veibell. Without their continued support there would be no Cisco IP Telephony-related Cisco Press books.

I would like to thank Chris Pearce for his help on the Call Routing chapter, Travis Amsler for his assistance on the Cisco CRA and extension mobility sections, and Brian Sedgley and Ken Pruski for their help with CCM and SDL tracing. Appreciation and recognition also go to the engineers who created and developed Dick Tracy: Rick Baugh, Jim Brasher, Long Huang, and David Patton.
## Contents at a Glance

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreword</td>
<td>xxv</td>
</tr>
<tr>
<td>Introduction</td>
<td>xxvi</td>
</tr>
<tr>
<td><strong>Chapter 1</strong> Troubleshooting Methodology and Approach</td>
<td>3</td>
</tr>
<tr>
<td><strong>Chapter 2</strong> IP Telephony Architecture Overview</td>
<td>23</td>
</tr>
<tr>
<td><strong>Chapter 3</strong> Understanding the Troubleshooting Tools</td>
<td>37</td>
</tr>
<tr>
<td><strong>Chapter 4</strong> Skinny Client Registration</td>
<td>113</td>
</tr>
<tr>
<td><strong>Chapter 5</strong> IP Phones</td>
<td>139</td>
</tr>
<tr>
<td><strong>Chapter 6</strong> Voice Gateways</td>
<td>169</td>
</tr>
<tr>
<td><strong>Chapter 7</strong> Voice Quality</td>
<td>383</td>
</tr>
<tr>
<td><strong>Chapter 8</strong> Fax Machines and Modems</td>
<td>433</td>
</tr>
<tr>
<td><strong>Chapter 9</strong> Call Routing</td>
<td>459</td>
</tr>
<tr>
<td><strong>Chapter 10</strong> Call Preservation</td>
<td>551</td>
</tr>
<tr>
<td><strong>Chapter 11</strong> Conference Bridges, Transcoders, and Media Termination Points</td>
<td>565</td>
</tr>
<tr>
<td><strong>Chapter 12</strong> Music on Hold</td>
<td>601</td>
</tr>
<tr>
<td><strong>Chapter 13</strong> Call Admission Control</td>
<td>623</td>
</tr>
<tr>
<td><strong>Chapter 14</strong> Voice Mail</td>
<td>655</td>
</tr>
<tr>
<td><strong>Chapter 15</strong> Survivable Remote Site Telephony (SRST)</td>
<td>707</td>
</tr>
<tr>
<td><strong>Chapter 16</strong> Applications</td>
<td>735</td>
</tr>
<tr>
<td><strong>Chapter 17</strong> SQL Database Replication</td>
<td>793</td>
</tr>
<tr>
<td><strong>Chapter 18</strong> LDAP Integration and Replication</td>
<td>819</td>
</tr>
<tr>
<td><strong>Appendix A</strong> Cisco IP Telephony Protocol and Codec Information and References</td>
<td>849</td>
</tr>
<tr>
<td><strong>Appendix B</strong> NANP Call Routing Information</td>
<td>857</td>
</tr>
<tr>
<td><strong>Appendix C</strong> Decimal to Hexadecimal and Binary Conversion Table</td>
<td>881</td>
</tr>
<tr>
<td><strong>Appendix D</strong> Performance Objects and Counters</td>
<td>891</td>
</tr>
<tr>
<td><strong>Glossary</strong></td>
<td>927</td>
</tr>
<tr>
<td><strong>Index</strong></td>
<td>947</td>
</tr>
</tbody>
</table>
Contents

Foreword xxv
Introduction xxvi

Chapter 1 Troubleshooting Methodology and Approach 3
  Developing a Troubleshooting Methodology or Approach 4
    Production Versus Nonproduction Outages 5
  Step 1: Gathering Data About the Problem 6
    Identifying and Isolating the Problem 6
    Using Topology Information to Isolate the Problem 7
    Gathering Information from the User 10
    Determining the Problem’s Timeframe 10
  Step 2: Analyzing the Data Collected About the Problem 11
    Using Deductive Reasoning to Narrow the List of Possible Causes 11
    Verifying IP Network Integrity 12
    Determining the Proper Troubleshooting Tool 13
  Case Study: Resolving a Problem Using Proper Troubleshooting Methodology 13
    Gathering the Data 14
    Analyzing the Data 18
    Conclusions 20

Summary 21

Chapter 2 IP Telephony Architecture Overview 23
  Network Infrastructure 23
  IP Telephony Infrastructure 23
    Call Processing 24
      Single-Site Deployment Model 24
      Multiple-Site Deployment Model 25
      Centralized Deployment Model 26
      Distributed Deployment Model 27
  Cisco AVVID IP Telephony Infrastructure 28
  Clients 29
    Cisco IP Phone Models 7960 and 7940 31
    Cisco IP Phone Expansion Module 7914 31
    Cisco IP Phone 7910 32
    Cisco IP Conference Station 7935 32
    Voice Gateways 32

  Cisco AVVID IP Telephony Applications 33

Summary 34
Chapter 3  Understanding the Troubleshooting Tools 37

Time Synchronization 38
  Configuring Automatic Time Synchronization on CallManager Servers 39
  Synchronizing Time Manually on CallManager Servers 40
  Synchronizing Time on Cisco IOS Devices 40
  Synchronizing Time on CatOS Devices 41

Reading CCM (or SDI) Traces 42
  Setting the Appropriate Trace Level and Flags 42
  Reading CCM Traces 50
  A Sample CCM Trace for a Call Between Two IP Phones 51
  Tracing a Call Through an MGCP T1 PRI Gateway 58

Reading SDL Traces 60
  SDL Overview 60
  Enabling SDL Trace and Setting the Appropriate SDL Trace Level 63

Microsoft Performance (PerfMon) 68
  Comparing PerfMon and the Real-time Monitoring Tool (RTMT) 68
  PerfMon Advantages 68
  RTMT Advantages 68
  Using PerfMon to View Real-Time Statistics 69
  Using Counter Logs 71
  Using Alerts 75

CCEmail 76
  Alerting Methods During Production and Non-production Hours 81
  Acquiring CCEmail 82

CallManager Serviceability 82
  Alarms 82
  Tracing 83
    Using XML-enabled Traces 83
    Searching for Devices with XML Traces 84
    Web-based Q.931 Translator 84
  Service Activation 84
  Control Center 85
  Real-Time Monitoring Tool (RTMT) 85
    Performance Tab 86
    Devices Tab 86
    CTI Apps Tab 88

Call Detail Records (CDR) and the CDR Analysis and Reporting (CAR) Tool 89

CDR Time Converter 90
  Acquiring the CDR Time Converter 91
Event Viewer 91
Q.931 Translator and Enhanced Q.931 Translator 95
   Enhanced Q.931 Translator 98
   Acquiring Enhanced Q.931 Translator 100
Dick Tracy 101
   Using the Dick Tracy Tool 102
   Using the CLI Tracy/Embedded Tracy Tool 105
   Acquiring Dick Tracy 105
Sniffer Traces 106
Voice Codec Bandwidth Calculator 106
Bug Toolkit (Formerly Bug Navigator) 106
Remote Access Tools 107
   Terminal Services 107
   Virtual Network Computing (VNC) 108
Websites and Further Reading 108
Best Practices 109
   VNC Best Practices 109
Summary 110

Chapter 4 Skinny Client Registration 113
Troubleshooting Inline Power 114
Troubleshooting Network Connectivity and Skinny Registration 117
   Verifying VLAN configuration 118
   Verifying IP Addressing Information 118
   Verifying TFTP Configuration File Download 121
   Understanding Skinny Registration 127
   Troubleshooting Skinny Registration 130
Additional Tools for Troubleshooting Skinny Client Registration Problems 133
   Checking IP Phone Status Messages 133
   Checking Registration with the Real-Time Monitoring Tool 135
Best Practices 137
Summary 137
Chapter 5  IP Phones 139

Understanding IP Phone Behavior 139
  Understanding the Skinny Protocol 139
  Call Processing Behavior 140
  Examining Skinny Protocol Messages in a CCM Trace 148
Understanding Failover and Failback 154
  Failover Behavior 155
  Failback Behavior 156
Understanding the Difference Between Restart and Reset 156

Troubleshooting IP Phone Problems 157
  Dropped Calls 157
  “CM Down, Features Disabled” 158
  Reasons for Failover 158
  Directory and Service Problems 160

79xx Series IP Phone 3-port Switch Operation 161

Best Practices 165
  Check Your Firmware 165
  Press the Help (i or ?) Button Twice During Active Calls 165
  Use a Custom Phone Service That Tracks Voice Quality Statistics 166
  Check the IP Phone Configuration Via Web Browser 167

Summary 167

Chapter 6  Voice Gateways 169

Cisco IOS Voice Gateways 169
  Cisco VG200 170
  Cisco 2600 Series Routers 171
  Cisco 3600 Series Routers 172
  Cisco 3700 Series Routers 173
  Cisco Catalyst 4224 173
  Cisco Catalyst 4000 Access Gateway Module (AGM) 174
  Cisco WS-SVC-CMM Communications Media Module (CMM) 174
  Other Cisco IOS Gateways 174

Understanding Dial Peer Matching in Cisco IOS Software 175

Understanding Cisco IOS Debugs and show Commands 184
  Correctly Setting the Timestamps 185
  Enabling Cisco IOS Software Debugs 185

Troubleshooting TDM Interfaces on Cisco IOS Gateways 187
  Useful show Commands for Troubleshooting TDM Interfaces 187
  Using debug Commands to Troubleshoot TDM Interfaces 192
  Understanding Cisco IOS CCAPI Debugs 196
Understanding the FXO Disconnect Problem 205

Troubleshooting Digital Interfaces 208
  Checking Physical Layer Connectivity on Digital Interfaces 208
  Troubleshooting ISDN PRI Signaling 210
  Troubleshooting T1 CAS 214

Understanding MGCP 218
  MGCP Endpoint Identifiers 219
  MGCP Commands 219
  MGCP Parameter Lines 221
  MGCP Packages 229
    Generic Media Package (G) 231
    DTMF Package 231
    MF Package (M) 232
    Trunk Package (T) 233
    Line Package (L) 234
    Handset Emulation Package (H) 235
    RTP Package (R) 236
    DTMF Trunk Package (DT) 236
    MF Trunk Package (MS) 237
  MGCP Response Headers and Response Codes 238

Cisco IOS MGCP Gateways 240
  MGCP FXS/FXO 249

Cisco IOS MGCP PRI 256
  Reading ISDN Messages 258
  Table of Q.850 Cause Codes 262
  Numbering Type and Plan Mismatches 269
  Troubleshooting Calling Name Display Problems 270
  Understanding ISDN Timers 271

Cisco IOS MGCP T1 CAS 276

Cisco IOS Gateways Using the H.323 Protocol 281
  H.225 Signaling 283
    H.225 Messages 283
    H.225 Information Elements 284
    H.225 Call Flow 287
  H.245 Signaling 295
    Master/Slave Determination 296
    Terminal Capabilities Exchange 297
    Logical Channel Signaling 300
    DTMF Relay 303
  Additional H.323 Debugs in Cisco IOS Software 305

Troubleshooting Problems with Ringback and Other Progress Tones 307
  No Ringback on an IP Phone When Calling the PSTN 308
  No Ringback on a PSTN Phone When Calling an IP Phone 309
No Ringback When Transferring a Call 309
The IP Phone User Does Not Hear In-band Messages When a Call Is Disconnected 310
Intercluster Trunks 311
Troubleshooting the WS-X6608 and WS-X6624 Voice Gateways 313
Recognizing and Powering the Module 313
Troubleshooting DHCP, TFTP, and Registration Problems 314
Troubleshooting DHCP Problems 314
Troubleshooting TFTP Problems 320
Troubleshooting Registration Problems 324
Catalyst WS-X6608 T1/E1 Digital Gateway Configuration 325
Troubleshooting Configuration Issues 326
Getting the D-channel Established 337
Checking Physical Layer Statistics on the WS-X6608 340
Verifying D-channel Configuration 343
Advanced Troubleshooting for D-channel Problems 344
Unexpected Resets 345
Using Dick Tracy to Analyze a WS-X6608 Port 345
Troubleshooting T1 CAS Problems on the WS-X6608 359
Catalyst WS-X6624 FXS Analog Gateway Configuration 367

Best Practices 380
Summary 381

Chapter 7  Voice Quality 383

Fixed and Variable Delays 384
Fixed Delay Sources 385
Coder (Processing) Delay 386
Packetization Delay 386
Serialization Delay 387
Propagation Delay 389
Variable Delay Sources 389
Queuing/Buffering Delay 390
Low-speed Links 391
Dejitter Delay 393
The Effects of Delay on Signaling 395

Analyzing and Troubleshooting Choppy and Garbled Audio 396
Packet Drops 397
Queuing Problems 400
The Effect of VAD on Voice Quality 402

Troubleshooting Problems with One-way or No-way Audio 405
Verifying IP Connectivity 405
One-way Audio on Cisco IOS Software Gateways 406
NAT, PAT, and Firewalls 410
Troubleshooting Echo Problems 410  
Sources of Echo 411  
  Electrical Echo 411  
  Acoustic Echo 412  
Talker Versus Listener Echo 412  
What Makes Echo a Problem 414  
How an Echo Canceller Works 416  
Eliminating Echo 418  
  Eliminating Echo on Cisco IOS Software Gateways 421  
  Eliminating Echo on the WS-X6608 and DT-24+/DE-30+ 424  
  Eliminating Echo Problems on Cisco IP SoftPhone 428  
Best Practices 429  
Summary 430  

Chapter 8  
Fax Machines and Modems 433  
Understanding Fax Machine Operation 433  
  Basic Fax Machine Operation 434  
  T.30 Messages 435  
Understanding Fax/Modem Passthrough Versus Fax Relay 437  
  Fax/Modem Passthrough 437  
  Named Service Events and Named Telephony Events 438  
  Basic Fax/Modem Passthrough Operation 439  
  Modem Passthrough Operation 439  
  Fax Passthrough Operation 440  
  Verifying Fax and Modem Passthrough Configuration 441  
  Fax Relay Basics 444  
The Effect of Packet Loss and Jitter on Fax  
  Machines and Modems 446  
First Steps in Troubleshooting Fax and Modem Problems 447  
  Checking for Physical Layer Problems on Digital Circuits 447  
Isolating and Troubleshooting Fax Problems 449  
  Adjusting the Fax Relay Data Rate 451  
  Disabling Error Correction Mode 452  
  Changing the Nonstandard Facilities Field 453  
  Changing the Fax Protocol 454  
  Checking the fax interface-type Command 454  
  Enabling Fax Relay Debugs 455  
Best Practices 457  
Summary 457
Chapter 9  Call Routing 459

Understanding Closest-match Routing 461
  Common Problems Associated with Closest-match Routing 465
  Outside Dial Tone Played at the Wrong Time 465
  Delayed Routing When Placing Seven-digit Local Calls 466

Understanding Calling Search Spaces and Partitions 469
  Calling Search Space/Partition Rules 474
  The First Partition Takes Precedence 474
  The Line-level Calling Search Space Takes Precedence over the Device-level
    Calling Search Space 476
  Event-specific Calling Search Spaces 478
  Call Forwarding Calling Search Spaces 479
    Call Forward No Answer (CFNA) 479
    Call Forward Busy (CFB) 480
    Call Forward All (CFA) 480
    Call Forward on Failure (CTI Ports and CTI Route Points Only) 485

Understanding and Troubleshooting Transformations and Masks 486
  Digit Discard Instructions (DDIs) 486
  Understanding the Concept of Masks 495
  Transformation Rules 496
    Order of Applied Transformations 496
    Cumulative Transformations 497
      Cumulative Transformation on Calling Party Number Example 497
      Cumulative Transformation on Called Party Number Example 498
    Overwritten Transformations 499
    Service Parameter-related Transformations 500

Understanding and Troubleshooting Translation Patterns 501

Understanding Route Filters 506

Digit Transformation Troubleshooting 513

Call Routing Troubleshooting 515
  Reading CCM Traces for Call Routing Information 516

Troubleshooting Hold, Transfer, Park, and Call Pickup 521
  Call Hold and Resume 522
  Call Transfer 529
  Call Park 531
  Call Pickup 533

Getting the Dialing Forest Traces 538

Best Practices 544
  Toll Fraud Prevention 544
    Preventing Transfers to Extension 9011 or Your Equivalent International
      Access Code 545
Using PLAR to Control Rogue Auto-registered IP Phones 545
Restricting the Call Forward All Field on IP Phones 546
Restricting Voice Mail Systems by Using Calling Search Spaces 547
Blocking Certain Area Codes 548

Summary 549

Chapter 10 Call Preservation 551

Understanding Call Preservation 551
Survivable Endpoints 552
IP Phones 552
MGCP Gateways 553
Nonsurvivable Endpoints 557
Skinny Gateways 557
H.323 Gateways 558
CTI/TAPI Endpoints 559
Media Processing Resources 560

Troubleshooting Call Preservation Issues 561
Best Practices 562

Summary 562

Chapter 11 Conference Bridges, Transcoders, and Media Termination Points 565

Media Resource Groups (MRGs) and Media Resource Group Lists (MRGLs) 566
MRGL Selection 567

Understanding Codec Selection 568

Transcoder Resources 570
Regions and the Regions Codec Matrix 570
Out-of-resource Conditions 578
Use of Transcoders in Conjunction with Other Media Resources 580
Transcoders in Conjunction with Conference Bridge Resources 581
Transcoders in Conjunction with MOH Servers 585

Conference Bridge Devices 586
Types of Conference Bridges 586
Troubleshooting “No Conference Bridge Available” 587
Troubleshooting Conference Failures 591
Other Conferencing Error Messages 597
“Already In Conference” 597
“Exceeds maximum parties” 597

Best Practices 598

Summary 598
Chapter 12  Music on Hold 601

Understanding MOH 601

Troubleshooting Data Points 603
  Performance Counters 604
  CCM Trace Files 607

Troubleshooting MOH 611
  Resolving Problems Related to Multicast and Unicast 615
  Determining Why Tone on Hold Is Playing 617
  Troubleshooting the Audio Translator 617
  Troubleshooting the Live Audio Source 619
    Configuring the Correct MOH Fixed Audio Source Device 619
    Selecting the Proper Recording Input 620

Best Practices 620

Summary 621

Chapter 13  Call Admission Control 623

Locations-based CAC 624
  Setting LocationsTraceDetailsFlag and CDCC Values 626
  The Role of Regions in CAC 627
  Locations-based CAC in Action 627
  Locations Reservations for Media Resources 631
    Locations-based CAC Reservations for Music on Hold Resources 631
    Locations-based CAC Reservations for Ad Hoc or Meet-Me Conferences 633
  Finding Bandwidth Leaks 635
  Locations and Call Preservation Interaction 636
  Troubleshooting Automated Alternate Routing 637

Gatekeeper Call Admission Control 638
  Checking Gatekeeper Configuration 640
  Verifying Gatekeeper Configuration on CallManager 641
  CallManager Registration with Gatekeeper 645
  Call Setup with Gatekeeper 647

Best Practices 652

Summary 652

Chapter 14  Voice Mail 655

Cisco Unity 655
  CallManager Integration 655
    Verifying Version Compatibility 656
    Verifying TSP Configuration 656
Verifying Cisco Unity Switch Configuration 658
Message Waiting Indicator (MWI) 659
Dual-Tone Multifrequency (DTMF) Relay Problems 661
Additional Unity Troubleshooting 662
More Troubleshooting Resources for Unity 662

SMDI Integration 662
Understanding SMDI Messages 663
  Call History Information for Calls to Voice Mail from CallManager 664
  Message Waiting Indicator On/Off Messages 665
  Error Messages 666
Cisco Messaging Interface 666
  CMI Configuration Parameters 667
  Reading CMI Traces 674
  Using HyperTerminal to Diagnose SMDI Problems 679
Message Waiting Indicator Problems 682
Cisco VG248 SMDI Integration 686
  Verifying Configuration Parameters 686
  Message Waiting Indicator Problems 690
Octel Voice Mail Digital Integration Via a DPA Voice Mail Gateway 693
  Verify Cabling 693
  Check Port Status 697
  Troubleshooting DPA MWI Problems 702
  Using the DPA Event Log 703

Best Practices 703
Summary 704

Chapter 15  Survivable Remote Site Telephony (SRST) 707

SRST Operation 707
  SRST Configuration 709
  IP Phone Registration 712
  SRST Dial Plan 718
  Debugging Call Control in SRST Mode 719
  Problems with Transferring Calls in SRST Mode 729
  IP Phones Stuck in SRST Mode 730
  Voice Mail and Forwarding Features in SRST Mode 731
  DHCP Considerations When Using SRST 731

Best Practices 732
Summary 733
Chapter 16  Applications 735

Customer Response Applications (CRA) 736
Checking TSP or JTAPI Plugin Versions 736
IP IVR and IP AA 737
   CRA Administration Problems 738
   Directory Configuration 741
   Verifying Configuration 744
   Engine Status 745
   Collecting Traces 748
Extension Mobility for CallManager 3.1 and 3.2 756
   CallManager Extension Mobility Configuration 758
   CRA Extension Mobility Configuration 759
   Configuration Summary 762
   Understanding the Login and Logout process 763
   Troubleshooting Extension Mobility on CallManager 3.1 and 3.2 765
Extension Mobility for CallManager 3.3 773
   Understanding the Login and Logout Process 775
   Troubleshooting Extension Mobility on CallManager 3.3 777

Cisco CallManager Attendant Console 779
   Understanding the Server Components 780
   Understanding the Attendant Console Client 781
   Troubleshooting Attendant Console 782

Cisco Personal Assistant 785
   Call Routing Problems and Personal Assistant 785
   Personal Assistant and Message Waiting Indicator Issues 786

Cisco IP SoftPhone 786
   Line Number Displays, But No Dial Tone 786
   Echo Problems with Cisco IP SoftPhone 787
   One-way Audio and Using Cisco IP SoftPhone over VPN 787
   Cisco IP SoftPhone Has No Lines 788

Cisco IP Phone Services 788

Cisco IP Videoconferencing (IP/VC) 789

Cisco Conference Connection 789
   Ensure the Necessary Services Are Started 790
   Using Event Viewer with Conference Connection 791

Cisco Emergency Responder (ER) 791

Summary 791
Chapter 17  SQL Database Replication 793

Understanding the Publisher-Subscriber Model 793
  Troubleshooting the Publisher-Subscriber Relationship 796

The Role of Name Resolution and Passwords in Replication 796

Microsoft SQL Server Enterprise Manager 802

Correcting Replication Errors 804
  Re-establishing a Broken SQL Replication Subscription 807
  Deleting the Subscription from the Publisher 807
  Adding the Subscription to the Subscriber SQL Server 808
  Starting the Snapshot Agent 809
  Reinitializing a Subscription 809

CDR Replication Issues 809
  Subscriber Is Not Configured to Generate CDRs 810
  Database Layer Monitor Is Not Running Properly 812
  Additional Problems with Writing CDRs 813

Best Practices 815

Summary 816

Chapter 18  LDAP Integration and Replication 819

Directory Integration Versus Directory Access 820
  Providing Endpoints with Corporate Directory Access 821
  Troubleshooting Corporate Directory Access 823

Using the CallManager Embedded Directory 823
  Troubleshooting the CallManager Embedded Directory 824
  Reconfiguring DC Directory on the Publisher 827
  CallManager 3.3 Reconfiguration Steps 828
  CallManager 3.0–3.2 Reconfiguration Steps 830
  Reconfiguring DC Directory on Subscribers 835

Understanding and Troubleshooting Active Directory Integration 837
  Troubleshooting Common Problems with Installing the Customer Directory Configuration Plugin 839
  Preparing Active Directory to Allow Schema Modifications 840
  Ensuring Domain Name Accuracy for Active Directory 841
  Verifying Distinguished Name Administrative Rights After Cisco Customer Directory Configuration Plugin Failure 842
  Checking Log Files for Errors 842
  Miscellaneous Troubleshooting Items 843

Understanding and Troubleshooting Netscape iPlanet Integration 844

Best Practices 845

Summary 846
Appendix A  Cisco IP Telephony Protocol and Codec Information and References 849

  Protocols 849
  Codecs 855

Appendix B  NANP Call Routing Information 857

Appendix C  Decimal to Hexadecimal and Binary Conversion Table 881

Appendix D  Performance Objects and Counters 891

  Cisco Performance Objects and Counters 891
    Cisco Analog Access Object 892
    Cisco CallManager Object 893
    Cisco CallManager Attendant Console Object—Release 3.3(2) 898
    Cisco CallManager System Performance Object 900
    Cisco CTI Manager Object 903
    Cisco Gatekeeper Object 904
    Cisco H.323 Object 904
    Cisco HW Conference Bridge Device—Release 3.3(2) 904
    Cisco Lines Object 905
    Cisco Locations Object 906
    Cisco Media Streaming App Object 906
    Cisco Media Termination Point Object—Through Release 3.3(2) 909
    Cisco Messaging Interface Object 910
    Cisco MGCP FXO Device Object 911
    Cisco MGCP FXS Device Object 912
    Cisco MGCP Gateways Object 912
    Cisco MGCP PRI Device Object 913
    Cisco MGCP T1 CAS Device Object 914
    Cisco MOH Device Object 915
    Cisco MTP Device Object 916
    Cisco Music on Hold Server Object—Through Release 3.3(2) 916
    Cisco Phones Object 918
    Cisco SW Conference Bridge Object—Through Release 3.3(2) 918
    Cisco SW Conference Bridge Device Object—Release 3.3(2) 919
    Cisco TFTP Object 920
    Cisco Transcode Device Object 923
    Cisco Unicast Hardware Conference Object 924
    Cisco Unicast Software Conference Bridge Device Object—Through Release 3.3(1) 924
    Cisco WebAttendant Object—Through Release 3.3(1) 924

Windows 2000 Objects 924

Glossary  927

Index  947
Foreword

In November of 1998, Cisco Systems acquired a small startup called Selsius Systems. For over a year this small company had been shipping the world’s first IP phones and Windows NT-based call management software consisting of close to a million lines of C++ code with a small development staff of about 40 engineers. Since the acquisition, the code base has evolved into many millions of lines of C++, XML, and Java code, and the development staff now has over 500 engineers. The level of sophistication and capability has increased dramatically and is a key component of the Cisco Architecture for Voice, Video, and Integrated Data (AVVID). Current deployments range from extremely distributed enterprises with hundreds of remote offices to small 50-person offices. Geographically, systems are deployed across the world, including exotic locations such as Antarctica and the International Space Station!

AVVID’s IP Telephony components (including the IP phones, gateways, and Cisco CallManager) comprise a telephony system that is both richer than and different from traditional TDM-based phone systems. For example, manageability and serviceability are achieved through either a browsable interface or an XML SOAP-based protocol for integration with existing IT systems. Geography disappears as a problem because telephony functions, manageability, and serviceability all traverse the IP network. Proprietary databases disappear in favor of standard SQL databases and LDAP directories. Nevertheless, this unification and standardization of telephony on IP networks also presents unique challenges. Voice quality can be impacted by poor IP network design. Capacity planning requires consideration of IP address numbering. Music on Hold as a multicast stream requires proper switch and router configuration. These are only a few examples of the unique considerations that must be given to IP Telephony deployments.

This book incorporates the authors’ real-life experiences in planning and troubleshooting IP Telephony within the AVVID solution. The wisdom contained herein has been gained over the course of thousands of real customer experiences. Paul Giralt and Addis Hallmark are two of the very best troubleshooters in the industry, and Anne Smith has written about and worked with the system since the earliest releases. Paul has been with Cisco’s customer support organization for several years. His depth and breadth of knowledge across all Cisco products are legendary, including his most recent focus on IP Telephony. I have seen him in action at some very large and sensitive customer installations, where he resolved extremely difficult problems and provided excellent guidance during upgrades and installations. We were fortunate to get him back, inasmuch as our customers were loathe to let him leave! Addis has been involved in the development and testing of many AVVID products. He has been personally engaged with many key customers during deployment and operation and has received numerous rave reviews from customers. Addis also has been instrumental in the security design aspects of Cisco CallManager. Anne is an author and the technical editor for this and several other AVVID books. She has been engaged with the technology since its inception at Selsius Systems.

I highly recommend this book to any individual or organization involved in installing, operating, or troubleshooting one of the most exciting advances in the long history of telephony. Written by three of its pioneers, this book serves as a guide for the rest of the pioneers who aren’t afraid to help their organization communicate in its own way, the better way, the IP way.

Richard B. Platt
Vice President for Enterprise Voice, Video Business Unit
Cisco Systems, Inc.
Introduction

This book teaches you the troubleshooting skills you need to isolate and resolve IP telephony problems. IP telephony is a relatively new technology with many different components. The Cisco IP Telephony (CIPT) solution revolves around Cisco CallManager, the core call processing engine. CIPT includes many different endpoints, such as IP phones, various gateways, and various applications such as Cisco IP IVR, Cisco CallManager Attendant Console, Cisco IP SoftPhone, Cisco Conference Connection, extension mobility, and more. Additionally, the network infrastructure plays an important role in prioritizing voice packets to ensure quality of service (QoS).

With all these components involved in transmitting voice across packet networks, it is essential that you be able to identify and resolve issues in the entire solution. This requires knowledge of the functionality of these components and how they interact with each other, as well as what tools are available to help you find the root cause when problems arise. This book educates you about the techniques, tools, and methodologies involved in troubleshooting an IP telephony system.

Target CallManager Release

This book is written to CallManager release 3.3. Updates to this book may be provided after publication. You should periodically check the ciscopress.com web site for updates (go to ciscopress.com and search for “Troubleshooting Cisco IP Telephony”).

Goals and Methods

This book intends to deliver a methodology you can follow when troubleshooting problems in an IP telephony network, particularly a Cisco IP Telephony solution. This book provides detailed troubleshooting information that applies to a variety of problems that can occur in any IP telephony deployment.

“Best Practices” sections in each chapter provide tips and design considerations to help you avoid common configuration problems.

Who Should Read This Book?

This book is designed to teach you how to isolate and correct problems in an IP telephony network. If you are a networking professional responsible for administering a Cisco IP Telephony (CIPT) system, this book is for you. Although this book’s main focus is on CIPT, some concepts apply to IP telephony in general as well.

You will best be able to assimilate the information in this book if you already have a working knowledge of a CIPT network.

How This Book Is Organized

Although you could read this book cover-to-cover, it is designed to help you find solutions to specific problems. The chapters are organized by the various components of a Cisco IP Telephony solution. Four appendixes provide reference information.

- **Chapter 1, “Troubleshooting Methodology and Approach”** — You can troubleshoot even the most complex problems if you have a good methodology in place for finding the root cause. This chapter focuses on teaching that methodology: learning how to find clues and track down your “suspect” by breaking the problem into smaller pieces and tackling each piece individually.
- **Chapter 2, “IP Telephony Architecture Overview”**—Cisco AVVID includes many different components that come together to form a comprehensive architecture for voice, video, and integrated data. This chapter covers the basic components of the IP Telephony architecture in order to provide a big-picture view of the system.

- **Chapter 3, “Understanding the Troubleshooting Tools”**—To effectively troubleshoot problems in a Cisco IP Telephony network, you must be familiar with the many tools at your disposal. In addition, you need to know how to best use those tools to achieve maximum results. This chapter describes the various tools and their different uses.

- **Chapter 4, “Skinny Client Registration”**—IP phone registration is a common source of problems. This chapter describes how Skinny protocol-based device registration works, including discussions of inline power, network connectivity, and potential TFTP and CallManager issues.

- **Chapter 5, “IP Phones”**—IP phones can encounter various problems, from unexpected resets to directory and service problems, and more. This chapter explains proper IP phone behavior and examines problems that can occur after an IP phone successfully registers.

- **Chapter 6, “Voice Gateways”**—Voice gateways are the interface that bridges the Voice over IP (VoIP) world with the Public Switched Telephone Network (PSTN). Voice gateways can be Cisco IOS Software gateways or modules within voice-enabled LAN switches. They can be analog or digital, and they can use a wide variety of signaling protocols. This chapter teaches you how to identify and resolve gateway problems by breaking these components into logical groups and following a methodical troubleshooting approach.

- **Chapter 7, “Voice Quality”**—Voice quality is a broad term that covers the following conditions: delayed audio, choppy or garbled audio, static and noise, one-way or no-way audio, and echo. This chapter focuses on the information you need to investigate and resolve voice quality problems in an IP Telephony network.

- **Chapter 8, “Fax Machines and Modems”**—Fax machines and modems present unique challenges when carried over an IP Telephony network, primarily due to their unforgiving nature concerning any modification to the audio stream. This chapter discusses the effect of packet loss and jitter, fax passthrough, fax relay, and how to troubleshoot modems and faxes.

- **Chapter 9, “Call Routing”**—Possessing a strong understanding of call routing is arguably one of the most important aspects of a smooth-operating CIPT solution. This chapter discusses closest-match routing, calling search spaces and partitions, transformations, and translation patterns as well as troubleshooting hold, transfer, park, and call pickup.

- **Chapter 10, “Call Preservation”**—Call preservation is easier to predict when you understand the protocol interaction with CallManager. This chapter provides guidelines for determining call survivability based on endpoint type and protocol.
• Chapter 11, “Conference Bridges, Transcoders, and Media Termination Points” — Conference bridges, transcoders, and media termination points are media resources. This chapter discusses the role of media resource groups and media resource group lists, codec selection, and troubleshooting transcoder and conference bridge resources.

• Chapter 12, “Music on Hold” — The Music on Hold feature allows callers to hear streaming audio while on hold. This chapter describes this feature and provides steps to take if you encounter problems.

• Chapter 13, “Call Admission Control” — Call admission control is used in situations where a limited amount of bandwidth exists between telephony endpoints such as phones and gateways. This chapter discusses the two types of call admission control — locations-based and gatekeeper — and the mechanisms available to reroute calls through the PSTN in the event of WAN congestion.

• Chapter 14, “Voice Mail” — CallManager is compatible with a variety of voice mail systems that integrate with CallManager through various methods. This chapter focuses on troubleshooting the integration of CallManager and three types of voice mail systems: Cisco Unity, third-party voice mail systems integrated via Simple Message Desk Interface (SMDI), and Octel Voice Mail, integrated through Cisco DPA Voice Mail gateways.

• Chapter 15, “Survivable Remote Site Telephony (SRST)” — SRST allows a router at a remote branch to assume call processing responsibilities in the event that phones at a remote site are unable to contact the central CallManager. This chapter describes SRST and provides detailed information about the various problems that can occur.

• Chapter 16, “Applications” — Cisco AVVID allows for the creation of many different applications to interoperate within the converged network. This chapter discusses some of the primary applications in a Cisco AVVID IP Telephony solution, such as IP AA and IP IVR, extension mobility, Cisco IP SoftPhone, Personal Assistant, and Cisco CallManager Attendant Console.

• Chapter 17, “SQL Database Replication” — The SQL relational database stores the majority of CallManager configuration information. This chapter discusses the Publisher-Subscriber model for database replication, name resolution, Enterprise Manager, Replication Monitor, broken subscriptions, and CDR database replication.

• Chapter 18, “LDAP Integration and Replication” — User information is stored in a Lightweight Directory Access Protocol (LDAP) database. This chapter describes directory integration versus directory access, using the CallManager embedded directory, and integrating with Active Directory and Netscape iPlanet.

• Appendix A, “Cisco IP Telephony Protocol and Codec Information and Reference” — Cisco IP Telephony employs many different protocols and codecs. This appendix provides a list of applicable protocols and codecs with descriptions and the standards body corresponding to the protocol or the Request for Comments (RFC) number. Compression rates are given for each codec.
• Appendix B, “NANP Call Routing Information”—CallManager provides a built-in dial plan for the North American numbering plan (NANP). This appendix provides information from the NANP file located in the C:\Program Files\Cisco\Dial Plan directory. This file shows you how each part of an NANP number corresponds to a specific placeholder. It is particularly useful when you’re learning how to apply route filters.

• Appendix C, “Decimal to Hexadecimal and Binary Conversion Table”—This appendix provides a cheat sheet that shows you how to quickly convert between decimal, hexadecimal, and binary values.

• Appendix D, “Performance Objects and Counters”—Microsoft Performance (PerfMon) and the Real-Time Monitoring Tool allow you to monitor your system through the use of performance counters. This appendix lists and describes the performance objects and counters in a Cisco IP Telephony network. Some pertinent Windows 2000 counters are also described.

• Glossary—The glossary defines terms and acronyms used in this book.

Best Practices
In a perfect world, there would be no need for this book, because systems would always run perfectly. Unfortunately, in the real world, problems do arise, and they usually don’t go away on their own. However, an administrator/installer can proactively take steps to ensure reliability and high availability and minimize the number of problems that arise.

Best practices include not only design considerations but also monitoring and management. A properly monitored system can detect failures before they become service-affecting. Each chapter contains a section outlining best practices as they apply to the chapter topic.

In a properly designed network, you can achieve 99.999 percent reliability—a rating that is expected of a telephone system.

High Availability in an IP Telephony Environment
High availability for IP telephony is based on distribution and core layers in the network and servers (call processing, application servers, and so on). BellCore Specification GR-512 defines what criteria must be met to achieve “five 9s” (99.999 percent) reliability. A careful examination of this document is recommended if you are interested in understanding 99.999 percent reliability. Note that many “events” are not counted against five 9s reliability. Some of these events include the following:

• Outages of less than 64 devices
• Outages less than 30 seconds in duration
• Outages due to outside causes, such as power loss from utility or network circuit failures caused by the provider
• Outages due to planned maintenance

The Cisco AVVID IP Telephony solution can achieve 99.999 percent reliability per the BellCore FR-512 specification.
Command Syntax Conventions
The conventions used to present command syntax in this book are the same conventions used in the IOS Command Reference. The Command Reference describes these conventions as follows:

- Vertical bars | separate alternative, mutually exclusive elements.
- Square brackets [ ] indicate an optional element.
- Braces { } indicate a required choice.
- Braces within brackets [{ } ] indicate a required choice within an optional element.
- **Boldface** indicates commands and keywords that are entered literally as shown. In actual configuration examples and output (not general command syntax), boldface indicates commands that the user inputs (such as a `show` command).
- *Italic* indicates arguments for which you supply actual values.

OSI Reference Model
Throughout the book, a few references are made to the OSI model. Table I-1 provides a brief primer on the OSI reference model layers and the functions of each. You can learn more about the OSI model in any of the Cisco Press books that target the CCNA certification.

<table>
<thead>
<tr>
<th>OSI Layer Name</th>
<th>Functional Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical (Layer 1)</td>
<td>Responsible for moving bits of data between devices. Also specifies characteristics such as voltage, cable types, and cable pinouts.</td>
<td>EIA/TIA-232, V.35</td>
</tr>
<tr>
<td>Data link (Layer 2)</td>
<td>Combines bytes of data into frames. Provides access to the physical media using a Media Access Control (MAC) address, which is typically hard-coded into a network adapter. Also performs error detection and recovery for the data contained in the frame.</td>
<td>802.3/802.2, HDLC</td>
</tr>
<tr>
<td>Network (Layer 3)</td>
<td>Uses logical addressing which routers use for path determination. Can fragment and reassemble data if the upper-layer protocol is sending data larger than the data link layer can accept.</td>
<td>IP, IPX</td>
</tr>
<tr>
<td>Transport (Layer 4)</td>
<td>Provides reliable or unreliable delivery of data packets. Allows for multiplexing of various conversations using a single network-layer address. Can also ensure data is presented to the upper layers in the same order it was transmitted. Can also provide flow control.</td>
<td>TCP, UDP</td>
</tr>
<tr>
<td>Session (Layer 5)</td>
<td>Sets up, coordinates, and terminates network connections between applications. Also deals with session and connection coordination between network endpoints.</td>
<td>Operating systems and application access scheduling</td>
</tr>
</tbody>
</table>
Table I-1  OSI Reference Model Overview (Continued)

<table>
<thead>
<tr>
<th>OSI Layer Name</th>
<th>Functional Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Presentation (Layer 6)</td>
<td>Defines how data is presented to the application layer. Can perform special processing,</td>
<td>JPEG, ASCII</td>
</tr>
<tr>
<td></td>
<td>such as encryption, or can perform operations such as ensuring byte-ordering is correct.</td>
<td></td>
</tr>
<tr>
<td>Application (Layer 7)</td>
<td>Interface between network and application software.</td>
<td>Telnet, HTTP</td>
</tr>
</tbody>
</table>

Comments for the Authors
The authors are interested in your comments and suggestions about this book. Please send feedback to the following address:
troubleshootingcipt@external.cisco.com

Further Reading
The authors recommend the following sources for more information.

Cisco Documentation
This book provides comprehensive troubleshooting information and methodology. However, details about common procedures might not be provided. You should be familiar with and regularly use the documentation that is provided with the Cisco IP Telephony system to supplement the information in this book.
You can find Cisco IP Telephony documentation by searching for a specific product on Cisco.com or by starting at the following link:
www.cisco.com/univercd/cc/td/doc/product/voice/index.htm
You can examine the following books at a technical bookseller near you or online by entering the title in the search box at www.ciscopress.com.

Cisco CallManager Fundamentals: A Cisco AVVID Solution
You can find detailed information about CallManager’s inner workings in the book Cisco CallManager Fundamentals (ISBN 1-58705-008-0).

Developing Cisco IP Phone Services: A Cisco AVVID Solution
You can find instructions and tools for creating custom phone services and directories for Cisco IP Phones in the book Developing Cisco IP Phone Services (ISBN 1-58705-060-9).

Cisco IP Telephony
Integrating Voice and Data Networks


Cisco Router Configuration, Second Edition

*Cisco Router Configuration, Second Edition* (ISBN 1-57870-241-0) provides example-oriented Cisco IOS Software configuration for the three most popular networking protocols used today—TCP/IP, AppleTalk, and Novell IPX.
Icons Used in This Book
Throughout this book, you will see a number of icons used to designate Cisco-specific and general networking devices, peripherals, and other items. The following icon legend explains what these icons represent.

**Network Device Icons**
- **CallManager**
- **IP Phone**
- **Stations**
- **SRST Router**
- **Router**
- **Switch**
- **Layer 3 Switch**
- **PIX Firewall**
- **Gateway or 3rd-party H.323 Server**
- **Modem**
- **Access Server**
- **ATM Switch**
- **PBX/PSTN Switch**
- **PBX (Small)**
- **Cisco Directory Server**
- **Local Director**
- **PC**
- **Laptop**
- **Server**
- **PC w/Software**
- **POTS Phone**
- **Relational Database**
- **Fax Machine**

**Media/Building Icons**
- **Network Cloud**
- **Ethernet Connection**
- **Serial Connection**
- **Telecommuter**
- **Building**
- **Branch Office**
To effectively troubleshoot problems in a Cisco IP Telephony (CIPT) network, you must be familiar with the many tools at your disposal. In addition, you need to know how best to use those tools to achieve maximum results. This chapter describes the various tools and their different uses.

Depending on the problem you encounter and your particular skill set, you might find certain tools more helpful than others. Nevertheless, knowing what tools are available and how they can help you solve the problem is essential to a successful resolution. Also, some tools might be more useful to you based on your past experience. If you are strong in IP, a sniffer trace might be your preferred tool when applicable. However, if you are stronger in call processing-related traces, the Cisco CallManager (CCM, also sometimes called SDI) traces might prove helpful once you understand how they work.

Later chapters demonstrate the use of these tools in different scenarios you might encounter.

This chapter covers the following topics:

- **Time synchronization** — Explains how to synchronize the clocks on all devices in an IP telephony network to ensure that timestamps on your trace files and debugs are synchronized with each other.

- **Reading CCM (or SDI) traces** — Describes one of the most important trace files you use to troubleshoot CallManager-related problems. CCM trace files provide information about call processing events and all messages exchanged between Skinny, MGCP, and H.323 endpoints.

- **Reading SDL traces** — Discusses the components of the less-used Signal Distribution Layer (SDL) trace files in CallManager. SDL traces describe the events occurring in the CallManager software at a code level. These traces are usually reserved for Cisco development engineering use; however, there are a few key pieces of information you can use to your advantage when troubleshooting CallManager problems.

- **Microsoft Performance (PerfMon)** — Details the capabilities of PerfMon, a built-in Windows 2000 utility that helps you troubleshoot CallManager.

- **CCEmail** — Details the third-party alerting tool that can be used in conjunction with PerfMon to configure alerts for the performance counters.
• **CallManager Serviceability**—Discusses various web-enabled tools provided with CallManager for reading alarms and XML-based tracing.

• **Real Time Monitoring Tool (RTMT)**—Describes the Cisco web-based monitoring application that allows you to view CallManager cluster details, monitor performance objects (much like PerfMon), and monitor devices and CTI applications.

• **Call detail records (CDR) and the CDR Analysis and Reporting (CAR) Tool**—Describes the CAR tool that helps you analyze the raw data that comprises the CDR database and create reports based on your search criteria.

• **CDR Time Converter**—Describes how to use this small utility that allows you to convert the UNIX Epoch-based date and time format stored in a CDR to standard date and time format.

• **Event Viewer**—Briefly explains the function of another built-in Windows 2000 tool that plays a key role in troubleshooting CallManager.

• **Q.931 Translator and Enhanced Q.931 Translator**—Describes two tools that read a CCM trace file and analyze all Q.931 and H.225 messages. The various information elements are decoded for ease of reading. The Enhanced Q.931 Translator also adds additional search and filtering options and decodes more information elements than the original Q.931 Translator.

• **Dick Tracy**—Describes an important tool used to troubleshoot the WS-X6608 and WS-X6624 voice gateways for the Catalyst 6000 series switches.

• **Sniffer traces**—Discusses when and why to use a network packet-capture tool.

• **Voice Codec Bandwidth Calculator**—Describes how to use the Voice Codec Bandwidth Calculator to determine the bandwidth used by different codecs with various voice protocols over different media.

• **Cisco Bug Toolkit (formerly Bug Navigator)**—Describes the web-based tool that allows you to find known bugs based on software version, feature set, and keywords. The resulting matrix shows when each bug was integrated or fixed if applicable.

• **Remote Access Tools**—Describes applications like Terminal Services and Virtual Network Computing, which allow you to access a server from a remote location.

• **Websites and Further Reading**—Provides URLs for websites that contain additional troubleshooting information. Also points you to a section in the “Introduction” with a recommended reading list.

### Time Synchronization

Time synchronization is simply making sure that all the participating CallManager servers and network devices have the same exact time. Time synchronization is critical. A large CallManager cluster can have eight or more separate servers, not including any voice mail servers or application servers. This distributed architecture creates a highly available and
scalable system. It also makes the troubleshooting process more involved because you have to collect traces from all participating servers to see the full picture of what happened.

As endpoints such as IP phones and voice gateways call each other, signaling occurs between CallManager and the endpoint device. Signaling also occurs between the respective CallManagers of each endpoint device. If a problem occurs, you need to consolidate trace files from all involved servers. CallManager Serviceability, discussed later in this chapter, can collect this information into one file. However, if the timestamps of each file are mismatched, it can be impossible to tell what the real series of call processing events is.

All the CallManager servers can be time synched using the Network Time Protocol (NTP). When you synchronize the time on all involved servers, all the trace files are timestamped the same. When trace files are collected for analysis, you can follow the true series of call processing events with accuracy.

In addition to CallManager servers, you should ensure all network devices such as switches, routers, and voice gateways are synchronized to the same time source as the CallManager servers. This ensures consistent timestamps regardless of which device you are looking at.

**Configuring Automatic Time Synchronization on CallManager Servers**

Use the following steps to configure the CallManager server to automatically synchronize—and stay synchronized—with a Time Server.

**Step 1** Verify that the NetworkTimeProtocol service is configured to launch automatically upon startup. Right-click My Computer and select Manage.

**Step 2** Expand the Services and Applications section, and select Services.

**Step 3** Double-click the NetworkTimeProtocol service, and ensure that Startup Type is set to Automatic.

**Step 4** Configure the C:\WINNT\ntp.conf file. This file contains the list of time servers that CallManager will synchronize with. You can configure CallManager to point to specific time servers (see Example 3-1), or you can configure it to receive NTP broadcasts (see Example 3-2) on the local LAN segment from the router (as long as the router is configured to do so).

**Example 3-1 Sample ntp.conf File Using Static Time Servers**

```
server 10.0.0.10
server 10.1.0.10
driftfile %windir%\ntp.drift
```
Synchronizing Time Manually on CallManager Servers

Use the following steps to manually configure time synchronization.

Step 1  Stop the NetworkTimeProtocol service in the Services Control Panel.

Step 2  Synchronize the clock by using one of the following commands from a command prompt.

To synchronize with a remote Time Server, use the following command where x.x.x.x is the IP address of the Time Server:

```
ntpdate x.x.x.x
```

To synchronize with a Broadcast Router, use the command where x.x.x.x is the IP address of the Ethernet port of the router:

```
ntpdate x.x.x.x
```

Step 3  Restart the NetworkTimeProtocol service in the Services Control Panel.

Synchronizing Time on Cisco IOS Devices

In addition to your CallManager servers, you should also ensure the time is synchronized on all your network devices, including IOS voice gateways, switches, and routers.

For Cisco IOS devices, you can use the Network Time Protocol (NTP) to synchronize the time. To enable NTP, you must configure the time zone the IOS device is in along with the IP address of the NTP server. You should configure the IOS device to take daylight savings time into account as well if you live in a time zone that observes daylight savings time.

First you must configure the time zone. Use the command `clock timezone`

`name_of_time_zone offset_from_GMT` to configure the time zone. For example, the following command configures the IOS device for Eastern Standard Time (EST):

```
clock timezone EST -5
```

If you are in a time zone that observes daylight savings time, use the command `clock summer-time name_of_time_zone recurring` to enable automatic daylight savings time. For example, the following configures Eastern Daylight Savings Time:

```
clock summer-time EST recurring
```
**clock summer-time EDT recurring**

You can determine time zones by performing a search at Google.com; search for a string such as “time zone GMT.” Any one of the many hits should point you to a table showing the various time zones around the world, including daylight savings time.

Once your time zone is configured properly, you can enable NTP. If you do not have a device on the network running NTP, you can make an IOS device into an NTP master clock. You can do this by configuring the command `ntp master` on the IOS device. If you are not making the device an NTP master, you should configure the IP address of the NTP server using the command `ntp server NTP_server_IP_address`. For example, the following tells the IOS device to synchronize its clock with the NTP server at IP address 172.18.109.1:

```
ntp server 172.18.109.1
```

Once you have enabled NTP, all IOS devices should synchronize their clocks with the central NTP server. You should also synchronize the clocks on non-IOS network devices such as switches running CatOS software.

---

**Synchronizing Time on CatOS Devices**

You can use NTP to synchronize the clocks on switches running CatOS software. The configuration is similar to configuring NTP on a device running Cisco IOS Software.

First you must configure the time zone the switch is in using the command `set timezone name_of_time_zone offset_from_GMT`. For example, the following sets the clock to Eastern Standard Time:

```
set timezone EST -5
```

You can also configure daylight savings time using the command `set summertime enable name_of_time_zone`. For example, the following enables Eastern Daylight Savings Time:

```
set summertime enable EDT
```

You can determine time zones by performing a search at Google.com; search for a string such as “time zone GMT.” Any one of the many hits should point you to a table showing the various time zones around the world, including daylight savings time.

Once you have the time zone configured properly, you can set the NTP server IP address and enable the NTP client on the switch. First set the NTP server IP address using the command `set ntp server NTP_server_IP_address`. Then enable the NTP client using the command `set ntp client enable`. The following shows the configuration to use the NTP server with IP address 172.18.109.1:

```
set ntp server 172.18.109.1
set ntp client enable
```

Once you have enabled proper time synchronization in your network, you can move on to reading the variety of trace files available to you for troubleshooting problems in an IP Telephony network.
Reading CCM (or SDI) Traces

CCM (also known as System Diagnostic Interface or SDI) traces are the user-friendliest call processing trace files you have available to you. After you learn a few tricks, it is easy to follow the call flows and find the potential problem. Although you might see “SDI” and “CCM” used interchangeably to refer to this type of trace, in this book we primarily refer to this type of trace as a CCM trace. Some pages in CallManager Serviceability refer to CCM traces as SDI traces.

Because CallManager is at the heart of a CIPT network, CCM traces are usually the first place to look when troubleshooting most problems. You can analyze problems related to device registration, call flow, digit analysis, and related devices such as IP phones, gateways, gatekeepers, and more. By the end of this section, you should be able to follow some basic call flows in the CCM trace. Future chapters continue to show CCM trace examples as you learn how to troubleshoot more problems. Later in this chapter you also learn about the Q.931 Translator, which is useful for quickly troubleshooting a variety of gateway problems. It is not a substitute for learning how to read the CCM trace, but it helps you quickly examine a trace in a graphical format without having to wade through irrelevant debugs. See the later “Q.931 Translator and Enhanced Q.931 Translator” section for more information.

Setting the Appropriate Trace Level and Flags

CallManager allows you to select from a variety of different options that adjust which events are logged to the CCM trace files. If you know exactly what you are looking for, you can configure CallManager to log only specific events in the CCM trace file. If you configure the trace to collect too much information, the trace becomes more difficult to analyze. However, if you do not configure the trace to collect enough information, you might miss the problem you are trying to find. When you are beginning to learn how to read the CCM trace files, it is best to enable more debugs than less. As you learn which trace settings are required for specific problems, you can enable just the settings you need.

Unfortunately you can never predict what problems will come up, so if you do not have a high level of tracing enabled, you may have to wait for a problem to happen a second time after enabling the appropriate trace settings before you can troubleshoot. For this reason, it is usually best to leave a majority of the trace flags enabled during normal operating conditions so you have trace data available if a problem occurs.

You configure a CCM trace in Cisco CallManager Serviceability (Trace > Configuration). Figure 3-1 shows the top half of the Trace Configuration page, and this section discusses the relevant fields and settings on that page.
First thing to note is the **Trace On** checkbox. This must be selected for any of the trace settings to be available.

The **Apply to All Nodes** checkbox allows you to apply the specified trace settings to all CallManager servers in the cluster. This is useful when you are troubleshooting a problem that is occurring on more than one CallManager server. You generally want to keep the same level of tracing enabled on all the servers in the cluster unless you are absolutely certain the problem is isolated to a single server in the cluster. Because of the distributed nature of CallManager, you may think a process only involves a single server when in reality part of the processing for a particular call is occurring on another server in the cluster.

The Trace Filter Settings area allows you to specify the exact parameters of your trace. First, you set the level of tracing you want to perform in the **Debug Trace Level** field.
CallManager Serviceability provides six different levels, but for nearly every kind of problem that you’ll want traces for, the recommended levels are either Detailed or Arbitrary:

- **Detailed** — Provides detailed debug information and highly repetitive messages that are primarily used for debugging, including KeepAlives and responses. For CallManager releases 3.1 and earlier, be cautious about using this level during normal production hours on a heavily loaded system. It can cause performance degradation on CallManager. In release 3.2 and later, CallManager uses asynchronous tracing to reduce the impact that trace file generation has on call processing.

- **Arbitrary** — Provides low-level debug traces. This level is best suited for debugging difficult problems. This level includes nearly everything that is included in Detailed with the exception of KeepAlives. If you are not troubleshooting problems related to missed KeepAlives, this level is good for day-to-day troubleshooting.

Other trace levels are provided, but generally, they don’t offer as much information as the Detailed and Arbitrary levels. The other levels are

- **Error** — Provides traces generated in abnormal conditions, such as coding errors or other errors that normally should not occur.

- **Special** — Provides traces for all informational, non-repetitive messages such as process startup messages, registration messages, and so on. All system and device initialization traces are at this level.

- **State Transition** — Provides traces for call processing events or normal events traced for the subsystem (signaling layers).

- **Significant** — Provides traces for media layer events.

Second, make sure the **Enable CallManager Trace Fields** checkbox is selected, which gives you the opportunity to select which specific traces you want to run and to choose your traces. Table 3-1 describes the trace fields to choose from. Most are reasonably self-explanatory. For example, if you’re having problems with an H.323 device, you would enable the **H245 Message Trace** and **Enable H225 & Gatekeeper Trace** options. Likewise, for music on hold (MOH) issues, the **Enable Music on Hold** option would display trace relating to all MOH activity.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable H245 Message Trace</td>
<td>Debugs H.245 signaling for H.323 calls, including the media processing messages.</td>
</tr>
<tr>
<td>Enable PRI Trace</td>
<td>Activates a trace of Primary Rate Interface (PRI) devices.</td>
</tr>
<tr>
<td>Enable ISDN Translation Trace</td>
<td>Activates a Layer 3 trace of Q.931 (ISDN messages).</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Enable H225 &amp; Gatekeeper Trace</td>
<td>Activates a trace showing H.225 signaling messaging for H.323 calls.</td>
</tr>
<tr>
<td>Enable Miscellaneous Trace</td>
<td>Activates a trace of miscellaneous devices.</td>
</tr>
<tr>
<td>Enable Conference Bridge Trace</td>
<td>Activates a trace of the conference bridges. Use this level to trace conference bridge statuses such as:</td>
</tr>
<tr>
<td></td>
<td>• Registered with CallManager</td>
</tr>
<tr>
<td></td>
<td>• Unregistered with CallManager</td>
</tr>
<tr>
<td></td>
<td>• Resource allocation processed successfully</td>
</tr>
<tr>
<td></td>
<td>• Resource allocation failed</td>
</tr>
<tr>
<td>Enable Music on Hold Trace</td>
<td>Activates a trace of MOH devices. Use this level to trace MOH device statuses such as:</td>
</tr>
<tr>
<td></td>
<td>• Registered with CallManager</td>
</tr>
<tr>
<td></td>
<td>• Unregistered with CallManager</td>
</tr>
<tr>
<td></td>
<td>• Resource allocation processed successfully</td>
</tr>
<tr>
<td></td>
<td>• Resource allocation failed</td>
</tr>
<tr>
<td>Enable CM Real-Time Information Server Trace</td>
<td>Activates CallManager real-time information traces used by the real-time information server.</td>
</tr>
<tr>
<td>Enable CDR Trace</td>
<td>Enables tracing of call detail record (CDR) processing. However, this trace flag does not provide much information about CDRs because CDR processing is mostly handled by the Database Layer Monitor and CDR Insert services discussed in Chapter 17, “SQL Database Replication.”</td>
</tr>
<tr>
<td>Enable Analog Trunk Trace</td>
<td>Activates a trace of all MGCP-based devices using an analog interface.</td>
</tr>
<tr>
<td>Enable All Phone Device Trace</td>
<td>Activates a trace of phone devices, including Cisco IP SoftPhones, and shows events such as on-hook, off-hook, key presses, and so on.</td>
</tr>
<tr>
<td>Enable MTP Trace</td>
<td>Activates a trace of media termination point devices and transcoders. Use this level to trace MTP device statuses such as:</td>
</tr>
<tr>
<td></td>
<td>• Registered with CallManager</td>
</tr>
<tr>
<td></td>
<td>• Unregistered with CallManager</td>
</tr>
<tr>
<td></td>
<td>• Resource allocation processed successfully</td>
</tr>
<tr>
<td></td>
<td>• Resource allocation failed</td>
</tr>
<tr>
<td>Enable All Gateway Trace</td>
<td>Activates a trace of all analog and digital gateways.</td>
</tr>
</tbody>
</table>
Chapter 3: Understanding the Troubleshooting Tools

Next, you can trace based on a specific device name by selecting the **Device Name Based Trace Monitoring** checkbox. When that checkbox is selected, you can click the **Select Devices** button to choose from a list of devices to trace. Tracing based on specific devices is very useful when you know which devices are involved in the problem, such as specific phones or gateways, and want to see trace output only for those devices.

The non-device option is a catch-all; if you’re having issues that are not device-related, you can select the **Include Non-device Traces** checkbox to see traces not related to devices.

Figure 3-2 shows the bottom half of the Trace Configuration page.

**Figure 3-2**  Trace Configuration (Part 2) in CallManager Serviceability

### Table 3-1  Cisco CallManager Trace Fields  (Continued)

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Forward and Miscellaneous Trace</td>
<td>Activates a trace of call forwarding and all subsystems not covered by another checkbox.</td>
</tr>
<tr>
<td>Enable MGCP Trace</td>
<td>Activates a trace showing media gateway control protocol (MGCP) messages for MGCP-based devices.</td>
</tr>
<tr>
<td>Enable Media Resource Manager Trace</td>
<td>Activates a trace for media resource manager (MRM) activities.</td>
</tr>
</tbody>
</table>

*Note: Device Name Based Trace also applies to SDL Trace*
For traces to be logged to a file, the **Enable File Trace Log** checkbox must be selected. The trace output is then sent to the path specified in the **File Name** field, which is C:\Program Files\Cisco\Trace\CCM\ccm.txt by default.

We’re going to skip discussion of the **Enable XML Formatted Output for “Trace Analysis”** checkbox for a moment to finish the fields related to file settings.

The Trace Configuration page in CallManager Serviceability validates the filename and ensures that it has a .txt extension. Do not use a filename that exists on another computer. Use a filename that exists on the computer running the trace. It is a good practice to have each server use a filename format that has the server name in it. This could be servera-ccm.txt, for example. That way, if trace files are collected from several servers, each file is easily distinguished from those collected on a different server. One downside to this, though, is that you can’t use the **Apply to all nodes** option. If you did, you would have to go back and change the names of the files on all nodes.

The **Maximum No. of Files** field defaults to 250. Normally, a maximum of 250 is never enough files because the files write in round-robin fashion—when the maximum limit is met, the next file starts at the beginning and overwrites the first file. Frequently, this means that the trace information you need to troubleshoot a problem that began occurring yesterday or the day before has already been overwritten by new files. For each server on which you’re running trace, determine how much free disk space you have, deduct a safety net of 500 MB, and then use the rest of your disk space for trace space. Assuming you don’t go above 10,000 lines—which we don’t recommend—you can calculate the size of your trace files by figuring the average CCM trace at 10,000 lines consumes about 2.5 MB; the average SDL trace at 10,000 lines consumes about 3.5 MB. We don’t recommend going above 10,000 lines because you want to keep your trace files at a manageable size. At these average sizes, you can easily zip the file and e-mail it if needed.

The default of 1440 for the **Maximum No. of Minutes per File** is adequate; you’ll probably never reach it.

The **Enable XML Formatted Output for “Trace Analysis”** checkbox takes the trace output and formats it in XML, which is required if you want use the Trace Analysis feature in CallManager Serviceability. With Trace Analysis, you can view the trace files in a web page, and the XML tagging lets you filter the trace results. The downside, however, is that the number of lines per file is limited to 2000 instead of 10,000. In most cases, you’ll probably want to stick with standard text-based tracing because the 2000 line limit severely restricts the amount of information in the trace file. Also, reading XML-formatted traces manually can be far more time-consuming than non-XML-formatted traces because you need to weed out all the extra XML tags that get added to each trace line. If you do not enable XML-formatted output, the log file compiles in text format.

The **Enable Debug Output String** checkbox sends debugging information to a Microsoft development tool useful only to Cisco development engineers. You should never need to or be asked to enable this checkbox.
Click **Update** to save your settings. The new trace settings take effect immediately when you click **Update**.

Once you’ve specified the trace settings, you can go to the Trace Collection page (**Trace > Collection**) to collect traces from one or more servers after an event has occurred. Set the service you want to retrieve the trace files for (such as Cisco CallManager), along with the date and time period you want to trace and click the **Submit Form** button. Depending on the time period you specified, tracing can impact performance, so be sure to trace short intervals that won’t impact CallManager or during non-production hours.

It is usually quicker and easier to manually collect the traces yourself using either Terminal Services or VNC (discussed later in this chapter) to access the CallManager server and copy the files to another machine for analysis.

Once the trace has been gathered, you are given the option to view it in a new window or use **Save As** to save the output to a file. If you choose to view the text-based output in a new window, it looks similar to Figure 3-3.

**Figure 3-3** Viewing Text-based Trace Output in Web Browser

If you choose the **Save As** option, the text-based output displays in the CallManager Serviceability window in the same format as Figure 3-3, and you can click **File > Save As** to save the file.
We generally recommend that you do not use the trace collection utility for anything other than very small traces on a system that is not busy. Manually copy the traces off the server(s) in question and analyze them offline.

If you selected the **Collect XML Trace File(s)** checkbox and choose to view the output in a new window, the Trace Analysis dialog appears. In this dialog box, you can filter the trace results based on the options described in Table 3-2.

**Table 3-2 Trace Analysis Filtering**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CallManager Host</td>
<td>Choose ALL CallManagers or select just one CallManager.</td>
</tr>
<tr>
<td>Device Name</td>
<td>Choose ALL or specify the device’s name. If you specify a device name, only trace information pertaining to that device name appears in the search results.</td>
</tr>
<tr>
<td>IP Address</td>
<td>Choose ALL or a specific server’s IP address.</td>
</tr>
<tr>
<td>Trace Type</td>
<td>Choose ALL, Alarm, or Trace. Trace shows all events as specified in the trace settings. Alarm shows only specific messages that meet the criteria of being an information Alarm message.</td>
</tr>
<tr>
<td>Cluster</td>
<td>Select this checkbox if you want to include the cluster name in the trace output.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>Select this checkbox if you want to include the date and time of each event listed in the trace output.</td>
</tr>
<tr>
<td>CM Node</td>
<td>Select this checkbox if you want to include the CallManager node (IP address or host name) in the trace output.</td>
</tr>
<tr>
<td>Trace Type</td>
<td>Select this checkbox if you want to include the trace type in the trace output.</td>
</tr>
<tr>
<td>IP Address</td>
<td>Select this checkbox if you want to include the device’s source IP address in the trace output.</td>
</tr>
<tr>
<td>Correlation Tag</td>
<td>Select this checkbox if you want to include the number that correlates traces with each other in the trace output.</td>
</tr>
<tr>
<td>Application Name</td>
<td>Select this checkbox if you want to include the directory numbers (DNs) and other service-specific information in the trace output.</td>
</tr>
<tr>
<td>Information</td>
<td>Select this checkbox if you want to include a description of what the trace found in the trace output.</td>
</tr>
<tr>
<td>Device Name</td>
<td>Select this checkbox if you want to include the device name in the trace output.</td>
</tr>
</tbody>
</table>

Figure 3-4 shows a trace file formatted as XML output and filtered to display only one CallManager host, one IP address, and the fields Cluster, Date and Time, CM Node, Source IP, and Information.
Chapter 3: Understanding the Troubleshooting Tools

Figure 3-4  Filtered Trace Output in XML Format

You can click the Back to Selection link to return to the Trace Analysis dialog and filter based on different criteria.

As you can see, reading through a large amount of trace files using the trace collection utility can be very cumbersome and time-consuming in relation to reading the trace files manually. You should learn how to read the CCM traces directly from the text files to help you troubleshoot problems more quickly and accurately. For that reason, all the examples of CCM traces that we show in this book are in plain text files. The following section describes how to read a text-based CCM trace.

Reading CCM Traces

This section shows you a few call flow examples and highlights key information in each example that helps you understand the CCM trace.

For brevity, the header and tail of the trace line have been omitted in many examples. For example, the complete trace would look like the following:

```
03/15/2001 05:34:41.956 CCM |StationInit: 1a3e8b54 OffHook.|<CLID::DLS2-CM152-SRV4-Cluster><NID::DLS2-CM152-SRV4><CT::1,100,96,1.69><IP::172.28.238.62><DEV::SEP003094C2D11F>
```

But for many examples, the header and tail do not add value. So the same trace is shown in this book as

```
StationInit: 1a3e8b54 OffHook.
```
The header portion of the trace line just specifies the date and time when the trace event was generated and which trace file you are looking at. For a CCM trace file, every line starts with the date and time followed by the letters “CCM.” Prior to CallManager 3.3 the trace files begin with the date and time followed by the words “Cisco CallManager.”

You should understand a few things about CCM traces:

- Many places in the trace files use hexadecimal equivalents for the IP addresses—The IP address 172.28.232.164 is shown in trace files as a4e81cac, which is a hexadecimal representation of 172.28.232.164. You can determine the IP address by working backwards: take the last two digits, ac, which is hex for 172; then 1c, which is hex for 28; then e8, which is hex for 232; and finally, a4, which is hex for 164. The IP address is 172.28.232.164. Appendix C, “Decimal to Hexadecimal and Binary Conversion Table,” provides a quick cheat sheet to determine how to quickly convert between decimal, hexadecimal, and binary values.

- Trace files sometimes use ASCII for directory numbers—Consider the value 33 30 30 31, which is how the directory number 3001 is sometimes displayed in the trace file.

- Trace files may include messages for a variety of protocols—Details on each of these protocols is described in the appropriate chapters. For example, the Skinny protocol is discussed in Chapter 4, “Skinny Client Registration,” and Chapter 5, “IP Phones.” Other protocols such as H.323 and MGCP are discussed in Chapter 6, “Voice Gateways.” Appendix A, “Cisco IP Telephony Protocol and Codec Information and References,” lists the protocols in an IP Telephony environment and the standards body or specification governing the protocol.

When you first open a CCM trace file, you might feel intimidated by the large amount of information presented in the trace file. We recommend that you use the default trace settings for CCM traces except you set the trace level to either Arbitrary or Detailed. Click the SetDefault button on the Trace Configuration page for CCM traces in CallManager Serviceability (Trace > Configuration) and then change the Debug Trace Level setting to Detailed.

A Sample CCM Trace for a Call Between Two IP Phones

As a first example at looking at CCM traces, go through a simple call between two IP phones. IP phones use the Skinny protocol to communicate with CallManager. All messages to and from a Skinny device are preceded by either the words StationInit or StationD.

For example, assume you have two phones, Phone A and Phone B. Phone A calls Phone B, Phone A goes off-hook and you see the following line in the CCM trace:

`StationInit: 1a3e8b54 OffHook.`

`StationInit` means that an inbound Transmission Control Protocol (TCP) message from a Skinny station reached CallManager. A Skinny station is any endpoint that uses the Skinny protocol to communicate with CallManager. This includes the Cisco 79xx family of
IP phones. In other words, any message that starts with StationInit is a message from an IP phone.

1a3e8b54 is probably the most important piece of this trace example. It is called a *TCP handle* and it represents a unique value that identifies a specific IP phone registered to this CallManager server. With the TCP handle, you can follow every message to and from that IP phone and see the full series of messages exchanged between CallManager and the phone. When searching through a CCM trace file in Notepad, copy the TCP handle to the clipboard (Ctrl+C), then open the Find box in Notepad (Ctrl+F), and paste the TCP handle (Ctrl+V). Once you get into the habit of highlighting the TCP handle and then pressing Ctrl+C, Ctrl+F, and Ctrl+V to enter the TCP handle into the Find window, you will be able to search for Skinny messages related to a device very quickly.

If you want to find the TCP handle for a particular IP phone, obtain the MAC address of the phone from either CallManager Administration or from the phone itself and search for the MAC address in the CCM trace until you find a KeepAlive to that phone. For example,

```
StationInit - InboundStim - KeepAliveMessage -
Send KeepAlive to Device Controller. DeviceName=SEP003094C2D11F,
TCPHandle=1a3e8b54, IPAddr=10.80.1.147, Port=51763,
Device Controller=[2,89,2992]
```

Notice the KeepAlive message contains both the TCP handle and the device name for the IP phone. Once you find the KeepAlive message, copy the TCPHandle field and use that to search through the CCM trace.

The *OffHook* message means that CallManager received a Skinny message indicating the phone went off-hook.

The next message is

```
StationD:  1a3e8b54 DisplayText text='                    3000        '
```

Notice that instead of StationInit you see StationD. This signifies that CallManager is sending a Skinny message to the phone. StationInit messages are sent from the IP phone to CallManager, while StationD messages are sent from CallManager to the IP phone. Skinny message transmission such as this between the IP phone and CallManager occurs for every action undertaken by the IP phone, including initialization, registration, on-hook, off-hook, dialing of digits, key presses on the phone, and so much more.

Again you see the same TCP handle, 1a3e8b54, listed for this message.

The number **3000** represents the directory number of the phone. If you know the phone number of the calling IP phone, you can often find the beginning of a call by simply searching for the calling phone’s directory number. In CallManager 3.3 and later the DisplayText message actually shows

```
StationOutputDisplayText don’t need to send, because mIsALegacyDevice = 0
```

The reason for this message is that the 79xx series Cisco IP Phones never paid attention to the DisplayText message in the first place, so in CallManager 3.3 and beyond, the message
is no longer sent. This means that if you search for the directory number of the IP phone, you might not find exactly the beginning of the sequence of events. It is best to search for the device name you are looking for and find a KeepAlive to get the TCP handle as discussed earlier.

Other messages sent to the IP phone include the following:

```plaintext
StationD: 1a3e8b54 StartTone tone=33(InsideDialTone), direction=0
StationD: 1a3e8b54 SetLamp stimulus=9(Line) stimulusInstance=1
  lampMode=2(LampOn).
StationD: 1a3e8b54 CallState callState=1 lineInstance=1 callReference=16777217
StationD: 1a3e8b54 DisplayPromptStatus timeOutValue=0
  promptStatus='Enter number' lineInstance=1 callReference=16777217.
StationD: 1a3e8b54 SelectSoftKeys instance=1 reference=16777217
  softKeySetIndex=4 validKeyMask=-1.
StationD: 1a3e8b54 ActivateCallPlane lineInstance=1.
```

Again you see that all the trace lines begin with `StationD` indicating that these are messages from CallManager to the IP phone and you see each line has the same TCP handle.

Do not concern yourself at this point about exactly what each of the pieces in the trace mean. These are all Skinny messages sent to the IP phone. At this point, you should just familiarize yourself with the basic call flow to understand how to read the trace files, not necessarily what each piece of the trace file means. Chapters 4 and 5 provide additional detail relating to the Skinny messaging you see in the preceding output. In particular, see Table 5-1, “Skinny Message Definitions,” and the section, “Examining Skinny Protocol Messages in a CCM Trace” in Chapter 5 for detailed explanations.

You will notice, however, that many of the trace messages are relatively self-explanatory. For example, the `StartTone` message with `tone=33(InsideDialTone)` tells the IP phone to start playing dial tone.

Note the callReference ID. A callReference ID is created for each participant in a call and you can use this ID to track a particular call through a CCM trace. A new callReference ID is created for each participant in a call and when some features are invoked, such as transfer and conference. Each leg of a call gets its own callReference ID assigned, so in a call between two IP phones, each phone gets assigned a separate callReferenceID.

So far you have only seen Skinny protocol messages; however, this callReference ID can help you correlate the Skinny messages with other messages to devices involved in the same call.

Next, the user on the IP phone begins dialing digits. This time, notice the difference between `StationD` and `StationInit` messages, indicating communication back and forth between CallManager and the IP phone.

```plaintext
StationInit: 1a3e8b54 KeypadButton kpButton=3
StationD: 1a3e8b54 StopTone
StationD: 1a3e8b54 SelectSoftKeys instance=1 reference=16777217
  softKeySetIndex=6 validKeyMask=-1
StationInit: 1a3e8b54 KeypadButton kpButton=0
StationInit: 1a3e8b54 KeypadButton kpButton=0
StationInit: 1a3e8b54 KeypadButton kpButton=1
Digit analysis: match(fqcn="3000", cn="3000", pss="IPMA:PA:Line1", dd="3001")
```
Notice that a 3 is dialed and a tone is then stopped. The Skinny protocol does not provide a mechanism to specify which tone to stop, so it sends a generic StopTone message. This stops any tones the IP phone happened to be playing at the time. In this case, remember you saw CallManager instruct the IP phone to play inside dial tone in the previous trace section.

The kpButton= message is always followed by the dialed digit. As soon as the first digit is dialed, the phone is told to stop playing dial tone. That makes sense because when you pick up the phone, you hear dial tone, but as soon as you dial a digit, the tone stops. Notice that after the phone is told to stop the tone, the soft keys are updated on the display. By looking at all the kpButton= messages, you can see that 3001 is the number dialed.

Note that the digit * is shown in a trace as the letter e and a # is shown as f. So for example, the message kpButton=e means the user entered the * key.

CallManager is constantly analyzing the digits the user dials, and once it finds an exact match, digit analysis returns the results for the match. Now it is time to ring Phone B, which is configured with DN 3001. Chapter 9, “Call Routing” provides additional detail about how digit analysis works in CallManager. There is, however, one important concept you should understand about digit analysis: Whenever digit analysis makes a match for a call, it displays the digit analysis results in the CCM trace. For example,

```
Digit analysis: analysis results
| PretransformCallingPartyNumber=3000 |
| CallingPartyNumber=3000             |
| DialingPartition=Line1             |
| DialingPattern=3001                |
| DialingRoutePatternRegularExpression=(3001) |
| DialingWhere=                      |
| PatternType=Enterprise             |
| PotentialMatches=NoPotentialMatchesExist |
| DialingSdiProcessId=(2,34,3500)    |
| PretransformDigitString=3001       |
| PretransformTagsList=SUBSCRIBER     |
| PretransformPositionalMatchList=3001|
| CollectedDigits=3001               |
| UnconsumedDigits=                  |
| TagsList=SUBSCRIBER                |
| PositionalMatchList=3001           |
| VoiceMailbox=                      |
| VoiceMailCallingSearchSpace=IPMA:PA:Line1 |
| VoiceMailPilotNumber=5678          |
| DisplayName=James                 |
| RouteBlockFlag=RouteThisPattern     |
| InterceptPartition=                |
| InterceptPattern=                  |
| InterceptWhere=                    |
| InterceptSdiProcessId=(0,0,0)      |
| InterceptSsType=0                  |
| InterceptSsKey=0                   |
| WithTags=                         |
| WithValues=                       |
| CgpnPresentation=NotSelected       |
| CallManagerDeviceType=UserDevice   |
```

Do not be concerned about what each of the fields means at this point. Chapter 9 explains some of the concepts such as partitions and calling search spaces. The important concept to grasp here is that any time digit analysis makes a match, you see a digit analysis result
Reading CCM (or SDI) Traces

similar to this one in the CCM trace. These digit analysis results are easy to spot in a CCM trace because of the white space to the right of the digit analysis results. If you look at a CCM trace, you will see that the majority of trace lines are over 100 characters in length; however, the digit analysis results are usually no more than 20 to 30 characters long, making the digit analysis results easy to find while scrolling quickly through a CCM trace file.

Because CallManager has collected all the required digits, it is ready to notify the destination IP phone there is an incoming call. The next example shows CallManager sending Skinny messages to Phone B.

```
StationD:  1a3e8af0  DisplayText text='                      3001        '
StationD:  1a3e8af0  CallState callState=4 lineInstance=1 callReference=16777218
StationD:  1a3e8af0  CallInfo callingPartyName='James' callingParty=3000
cgpnVoiceMailbox= calledPartyName='Mary' calledParty=3001 cdpnVoiceMailbox= originalCalledPartyName='Mary' originalCalledParty=3001
originalCgpnVoiceMailbox= originalCgpnRedirectReason=0
lastRedirectingPartyName='Mary' lastRedirectingParty=3001
lastRedirectingVoiceMailbox= lastRedirectingReason=0
callType=1(InBound) lineInstance=1
callReference=16777218
StationD:  1a3e8af0  SetLamp stimulus=9(Line) stimulusInstance=1
lampMode=5(LampBlink)
StationD:  1a3e8af0  SetRinger ringMode=2(InsideRing)
StationD:  1a3e8af0  DisplayNotify timeOutValue=10 notify='From 3000'
StationD:  1a3e8af0  DisplayPromptStatus timeOutValue=0 promptStatus='From 3000'
stationInstance=1 lineInstance=1 callReference=16777218
StationD:  1a3e8af0  SelectSoftKeys instance=1 reference=16777218
softKeySetIndex=3 validKeyMask=-1
```

Notice first that a **StationD** message is generated. This means that a message is sent from CallManager to the IP phone. Also notice that the TCP handle is different than in the preceding trace output. Each IP phone has a unique TCP handle assigned to it at registration. You can use the unique TCP handle to differentiate between the Skinny messages sent to and from Phone A (TCP handle 1a3e8b54) and those sent to and from Phone B (TCP handle 1a3e8af0). A new TCP handle is assigned any time an IP phone such as Phone A unregisters and reregisters to CallManager, resets, or fails over or back from one CallManager to another.

Notice also that the **callReference** value (16777218) is different from the previous output. As we mentioned earlier, this is because each leg of a call is assigned a different call reference. In this case, this is the call reference for Phone B. This call reference persists for the duration of the call on Phone B.

Phone B rings and the call information shows that James called Mary. You see several messages that seem to suggest the call is being redirected; it’s not. These are just standard messages sent by CallManager.

Once again, do not be concerned about exactly what each message means at this point. Future chapters go into detail about each message; however, you can see from reading the trace that Phone B is being told to ring (**SetRinger ringMode=2(InsideRing)**) and display “From 3000” on the prompt line of the IP phone (**promptStatus=‘From 3000’**).
Now that a call is in progress, Phone A gets some updated information, including display information such as called and calling party names.

```
StationD: 1a3e8b54 SelectSoftKeys instance=1 reference=16777217
  softKeySetIndex=8 validKeyMask=-1.
StationD: 1a3e8b54 CallState callState=12 lineInstance=1 callReference=16777217
  callingPartyName='James' callingParty=3000
  cdpnVoiceMailbox= calledPartyName='Mary' calledParty=3001
  cdpnVoiceMailbox= originalCalledPartyName='' originalCalledParty=
  originalCdpnVoiceMailbox= originalCdpnRedirectReason=0
  lastRedirectingPartyName='' lastRedirectingParty=
  lastRedirectingVoiceMailbox= lastRedirectingReason=0
StationD: 1a3e8b54 StartTone tone=36(AlertingTone).
StationD: 1a3e8b54 CallState callState=3 lineInstance=1 callReference=16777217
StationD: 1a3e8b54 SelectSoftKeys instance=1 reference=16777217
  softKeySetIndex=8 validKeyMask=-1.
StationD: 1a3e8b54 DisplayPromptStatus timeOutValue=0 promptStatus='Ring Out'
  lineInstance=1 callReference=16777217.
```

Basically, these messages perform two actions. First, the display on Phone A changes now that the call is in progress. Second, the phone is told to play an alerting tone. The alerting tone is the standard ringback tone you hear when placing a call. You also see the first `callReference` value, 16777217, indicating the original call placed by Phone A. Remember each call reference is only valid for one leg of the call.

```
StationInit: 1a3e8af0 OffHook
StationD: 1a3e8af0 ClearNotify
StationD: 1a3e8af0 SetRinger ringMode=1(RingOff)
StationD: 1a3e8af0 SetLamp stimulus=9(Line) stimulusInstance=1
  lampMode=2(LampOn)
StationD: 1a3e8af0 CallState callState=1 lineInstance=1 callReference=16777218
StationD: 1a3e8af0 ActivateCallPlane lineInstance=1
```

The `StationInit OffHook` message indicates that Phone B goes off-hook and answers the call. CallManager sends a `SetRinger ringMode=1(RingOff)` message, which tells Phone B to stop ringing, and the preparation is now complete for the actual media connection.

Next, we'll examine how the audio stream is set up. As with all VoIP protocols, Skinny uses Real-Time Transport Protocol (RTP) streams over User Datagram Protocol (UDP) packets to send and receive Voice over IP (VoIP) samples. Each RTP stream is called a logical channel. A logical channel is a unidirectional RTP stream, so to have a two-way conversation, you must have two logical channels opened—one from the calling device to the called device and one from the called device to the calling device.

In a call involving a Skinny device, CallManager asks the IP phone to open a connection to receive RTP packets. CallManager asks the IP phone for specific parameters for this connection, including the codec and packet size. You can see the following `OpenLogicalChannel` messages from CallManager to each IP phone requesting that they open a connection to receive RTP packets using G.711.

```
StationD: 1a3e8b54 OpenReceiveChannel conferenceID=0
  passThruPartyID=17 millisecondPacketSize=20
  compressionType=4(Media_Payload_G711Ulaw64k)
  qualifierIn=? myIP: 3eee1cac (172.28.238.62)
StationD: 1a3e8af0 OpenReceiveChannel conferenceID=0
  passThruPartyID=33 millisecondPacketSize=20
  compressionType=4(Media_Payload_G711Ulaw64k)
  qualifierIn=? myIP: 2fee1cac (172.28.238.47)
```
Upon receiving an **OpenReceiveChannel** message, the IP phone selects the UDP port number it wants to use to receive RTP packets and reports this information back to CallManager in an **OpenReceiveChannelAck** message. Phone A responds first:

```
StationInit: 1a3e8b54 OpenReceiveChannelAck
Status=0, IpAddr=0x3eee1cac, Port=20096, PartyID=17
```

Once CallManager receives this information from Phone A, it can tell Phone B where to send its RTP stream. Until this point, CallManager could not tell Phone B which UDP port number to use because Phone A had not reported it to CallManager. Once CallManager receives the port number in the **OpenReceiveChannelAck** message from Phone A, it sends a **StartMediaTransmission** message to Phone B giving it the IP address and port number of Phone A along with information about which voice codec to use.

```
StationD: 1a3e8af0 StartMediaTransmission
conferenceID=0 passThruPartyID=33
remoteIpAddress=3eee1cac(172.28.238.62)
remotePortNumber=20096 milliSecondPacketSize=20
compressType=4(Media_Payload_G711Ulaw64k)
qualifierOut=?.
myIP: 2fee1cac (172.28.238.62)
```

Next, CallManager receives an **OpenReceiveChannelAck** from Phone B containing the UDP port number information and passes this on to Phone A in a **StartMediaTransmission** message.

```
StationInit: 1a3e8af0 OpenReceiveChannelAck Status=0,
IpAddr=0x2fee1cac, Port=19648, PartyID=33
StationD: 1a3e8b54 StartMediaTransmission
conferenceID=0 passThruPartyID=17
remoteIpAddress=2fee1cac(172.28.238.47)
remotePortNumber=19648 milliSecondPacketSize=20
compressType=4(Media_Payload_G711Ulaw64k) qualifierOut=?.
myIP: 3eee1cac (172.28.238.47)
```

So at this point, Phone A (TCP handle 1a3e8b54) is sending RTP packets to 172.28.238.47, which happens to be Phone B, and Phone B (TCP handle 1a3e8af0) is sending RTP packets to 172.28.238.62, which happens to be Phone A.

Notice that for the duration of this call, Phone A has never sent nor received any Skinny signaling to or from Phone B. This is because all the signaling goes through CallManager. The only time IP phones send packets to each other is for the actual voice stream. This is what allows CallManager to set up calls between devices that use different signaling protocols. For example, if Phone A called a phone number on the PSTN instead of another IP phone, the signaling between CallManager and Phone A remains the same. Phone A has no idea that it is sending RTP packets to a voice gateway and vice versa.

When reading CCM traces, you can usually separate each leg of the call and concentrate on one part at a time. For example, if you have a call that goes out through a voice gateway, once you have verified that the IP phone dialed the correct digits and CallManager is routing the call to a voice gateway, you can focus on the gateway debugs to determine how the call gets set up.
Also it is important to separate the signaling aspects of a call set up from the RTP media streams. All VoIP devices are blindly told to send RTP packets to an IP address and port number without knowing what type of device they are sending these packets to. As long as the terminating device provided the correct IP address and port number and CallManager relayed this information correctly, everything works properly. However, pay attention to the signaling aspects to ensure that the port number received from the terminating device is the same as the port number reported to the originating device.

**Tracing a Call Through an MGCP T1 PRI Gateway**

The next call to dissect is an IP phone (3000) making a call through a WS-X6608-T1 PRI MGCP gateway. MGCP is not described in detail in this example; however, you will see ISDN Q.931 messages because this is the protocol used over the PRI D-channel. A detailed discussion on troubleshooting ISDN PRI problems can be found in Chapter 6. Appendix A includes the standards body and specification for ISDN Q.931.

All Q.931 and H.323 (including intercluster trunk) calls look basically the same in a CCM trace. Also, some non-ISDN gateways such as the WS-X6624 and the WS-X6608 when using T1 CAS also use Q.931 messages to communicate with CallManager. This makes understanding the basic structure of this kind of trace message important.

The majority of information presented in the CCM trace file for a gateway call is in hexadecimal notation. The Q.931 specification states that phone numbers should be encoded in ASCII as well as character strings, such as display names. The CCM trace also uses hexadecimal equivalents for the IP addresses on some occasions. Do not be intimidated by the hexadecimal values. Once you understand how to decode them, they are actually easy to understand. Appendix C provides a cheat sheet for conversions.

Whenever an H.323 or Q.931 call is made, you will see a section of trace similar to the following outgoing ISDN setup message:

```
Ie - Ni2BearerCapabilityIe IEData= 04 03 00 90 A2
Ie - Q931ChannelIdIe IEData= 18 03 A9 83 97
Ie - Q931CallingPartyIe IEData= 6C 06 00 80 33 30 30 30
Ie - Q931CalledPartyIe IEData= 70 05 80 33 30 30 30 30
MMan_Id= 0. (iep= 0 dsl= 0 sapi= 0 ces= 0 IpAddr=a4e81cac IpPort=2427)
```

The first line gives you information about the direction of the call (either **Out Message** or **In Message**) followed by the type of message (**PriSetupMsg**) and the protocol used for the message (**PriNi2Protocol**). The direction is from the perspective of CallManager, so this means a device registered to CallManager is placing a call out through a gateway.

As with the IP phone, a unique identifier is used to keep track of the call. Use the hexadecimal IP address **IpAddr=a4e81cac** in conjunction with call reference number **02 00 09** to track the call throughout the trace. When viewing an ISDN trace, look at the **IsdnMsgData2** line to find the call reference ID. In some cases you will see **IsdnMsgData**
or IsdnMsgData1 instead of IsDNMsgData2. They are equivalent. Ignore the first two numbers (usually 08). The next two numbers are the call reference length (02), and the next four are the call reference value (00 09).

You might be wondering how a4e81cac is converted to an IP address in dotted decimal notation. This is a hexadecimal representation of the IP address. You can figure out the IP address by working backwards. In this example, first, take the last two digits, ac, which is hex for 172. Next, consider the 1c, which is hex for 28. Third, take e8, which is hex for 232. Finally, take a4, which is hex for 164. The IP address is 172.28.232.164.

**NOTE**
Appendix C provides a quick cheat sheet to determine how to quickly convert between decimal, hexadecimal, and binary values.

So far you know you are looking at an outbound setup message on a PRI configured for the NI2 ISDN protocol, all of which you determined from the first line of the trace. You also know the call reference from decoding the first few bytes of the IsdnMessageData2. The lines in the middle that begin with Ie are ISDN Q.931 information elements. Information elements are covered in detail in Chapter 6; however, they are all formatted the same way in the CCM trace. For example, the following is the called party information element (IE):

```
Ie - Q931CalledPartyIe IEData= 70 05 80 33 30 30 31
```

Each information element line begins with the letters Ie followed by the name of the information element, in this case Q931CalledPartyIe. After the name follows the data contained in that information element. The format of the data that follows is dependent on the particular information element. The Q.931 and H.225 specifications describe the format of each information element. Having a copy of the ITU-T Q.931 specification can prove to be invaluable when troubleshooting ISDN problems because you can get down to the exact details of each bit in the IE data.

Fortunately for most day-to-day activities you do not need to reference the Q.931 specification because the information you need is easily identifiable. For example, in the called party number information element shown above, notice the IE data contains the sequence 33 30 30 31, which is 3001 in ASCII. This represents the directory number that is being called. Now you can follow the call’s events.

Just as you searched through the CCM trace for the TCP handle of the IP phone to find the next message associated with a particular phone, you can do the same for a Q.931 or H.323 call. The call reference for a call remains the same for the duration of a call. The first bit, however, of the call reference is flipped depending on the direction of the call. Chapter 6 goes into detail about how this works, so don’t worry about it right now. All you need to know is to search for the call reference minus the first digit. In this case the call reference is 00 09, so search for 0 09.
If you search through the CCM trace, you come to the next message. Notice that the message that follows is an In Message. This means it is an inbound message sent to CallManager by the ISDN network:

```
In Message -- PriCallProceedingMsg -- Protocol= PriNi2Protocol
Ie - Q931ChannelIdIe -- IEData= 18 03 A9 83 97
MMan_Id= 0. (iep=  0 dsl=  0 sapi=  0 ces= 0 IpAddr=a4e81cac IpPort=2427
IsdnMsgData1= 08 02 80 09 02 18 03 A9 83 97
```

Notice that the call reference is now 02 80 09. The most-significant bit (MSB) is set on the call reference value. This bit determines if this message is the originating or terminating side; however, you do not need to know what this bit means because CallManager clearly tells you which direction the message is going when it says In Message or Out Message. When searching, you need to look for 02 00 09 or 02 80 09 to track all events relating to this call event, although searching for just 0 09 is usually good enough.

These are just some of the tricks that help you follow call flows through the CCM trace. Subsequent chapters provide additional trace examples as we investigate other troubleshooting scenarios. Half the battle in reading a CCM trace is knowing which pieces of the trace file to ignore so that you can focus on the important messages in the trace. As you read through the following chapters, you will get a better understanding of the different messages you might find in a CCM trace.

### Reading SDL Traces

An SDL trace is a very detailed trace mainly used by Cisco development engineers for code-level analysis of call processing events. To the average CallManager administrator, SDL trace files are far too detailed for normal practical use.

If you’re working with the Cisco Technical Assistance Center (TAC) on problem resolution, TAC might ask you to collect SDL traces so that it can forward them to CallManager developers for analysis. You need to know how to configure the trace files to capture the right information for TAC.

Although the SDL trace files are generally not used for troubleshooting purposes, you will find a few occasions in this book where you must look in the SDL trace file to get a full understanding of a particular problem. In cases like this, we will provide details on exactly what to look for.

### SDL Overview

SDL traces provide a C programming language interface to alarms and trace information in CallManager. Alarms are used to inform a TAC engineer or CallManager developer of unexpected events, such as being unable to access a file, database, Winsock, or other operating system resources.

SDL traces can span multiple servers, allowing a process on one server to communicate with a process on another server transparently. This mechanism is supported by the use of...
SDL links. An SDL link spans from one server supporting SDL to another server supporting SDL.

SDL maintains a circular queue of files to log information. Over time, a file will be overwritten. The number of files (determined by the CallManager service parameter `SdlTraceTotalNumFiles`) and the number of lines per file (determined by the CallManager service parameter `SdlTraceMaxLines`) governs how long it takes to overwrite old log files.

SDL generates two types of files:

- **Log files**—Contain the actual tracing information.
- **Index files**—Indicate which log file in the circular queue is currently being used for writing. An index file allows SDL logging to start where it left off each time an application is restarted.

Log filenames are composed of the following:

```
SDLnnn_qqq_xxxxxx.txt
```

where

- `nnn` represents the node ID
- `qqq` represents the application ID (CallManager = 100)
- `xxxxxx` represents the unique file index

Index filenames are composed of the following:

```
SDLnnn_qqq.index
```

where

- `nnn` represents the node ID
- `qqq` represents the application ID (CallManager = 100)

The actual SDL log and index files are text-based. All columns in a log entry are delimited by a vertical bar (`|`) character (0x7c). A log entry is broken into two components, prefix and detail, which appear in the following format:

```
| Prefix Component | Detail Component |
```

Every log entry contains a prefix. The prefix always has the same format. The common prefix of a trace line is as follows:

```
uuuuuuuuu | yy/mm/dd-hh:mm:ss:vvv | nnn | xxxx |
```

where

- `uuuuuuuuu` represents a unique line sequence timestamp
- `yy` represents the year
- `nnn` represents the month
- `dd` represents the day
- `hh` represents the hour
- `nnn` represents the minutes
- `ss` represents the seconds
vvv represents the milliseconds

nnn represents the node ID

xxxx represents the log entry type (which defines the type of entry being logged)

The SDL process running on each CallManager allows each node in the cluster to communicate with other nodes in the cluster to exchange information. SDL is what allows CallManager clustering to work by allowing processing for tasks to run on any node in the cluster. Generally, processing for an event occurs on the node where the device performing the given action is registered. For example, if an IP phone is registered to a particular CallManager in a cluster, that CallManager usually handles all call processing operations for that phone. But what if an IP phone registered to one CallManager in the cluster places a call out a gateway registered to a different CallManager in the cluster? In this case, some of the processing might occur on one node while the rest of the processing occurs on another node.

CallManager functions by sending signals from one process to another. These signals are just messages from one piece of software to another internal to CallManager. These signals can be sent to a piece of code on an entirely different CallManager node in the cluster. The SDL trace lets you see this happening. The best way to see this is by looking at the log entry type field at the beginning of an SDL trace file. For example,

011381785 | 02/09/16 14:51:37.874 | 002 | SdlSig-O

The log entry type is SdlSig-O. This means this CallManager server sent an SDL signal to another node (O means outgoing). Conversely, if you see SdlSig-I, this tells you the CallManager node you are looking at received an SDL signal from another node or another process. The CallManager and CTI Manager process communicate with each other using SDL links as well, so you see communication to and from CTI Manager in the SDL trace as SdlSig-O and SdlSig-I.

So how do you know which node the signal was coming from or destined to? After the SdlSig-O or SdlSig-I message you should see several columns of text separated by the vertical bar (|) character. For example,

SdlSig-O | DbDeviceClose | initialized | Db(1,100,20,1) | Db(2,100,20,1)

An actual SDL trace has many more columns and spaces between columns than what is shown here. We have condensed the trace to fit on the printed page. The two columns to examine are the third and fourth columns after SdlSig-O. These describe the process sending the signal and the process to which the signal is being sent. The third column shows the destination (Db(1,100,20,1)) and the fourth shows the source (Db(2,100,20,1)).

Now you need to understand what you are looking at when you see Db(2,100,20,1). Db is the process name. You do not need to understand what the various process names signify. The important part is the four comma-separated numbers that follow the process name. The four numbers represent (in this order)
• Node number
• Application number (100 = CallManager, 200 = CTI Manager)
• Process type
• Process instance

The only ones you should be concerned with are the node number and the application number. In this case, you can see the signal is from CallManager on node 2 to CallManager on node 1.

You might be wondering why you need to know all this. Because CallManager is a distributed architecture, you might find that when looking at a CCM trace the events occurring don’t seem to make sense. For example, you might see an IP phone making a call but never see the call go out a gateway to its destination, or you might see an IP phone told to play reorder for no apparent reason. When you see an unexplained event in the CCM trace, look at the same timestamp in the SDL trace to see if there was a signal from another node at that time. If so, look in the CCM trace on the other node at the same time to see if there is any additional detail about the call on the other node. This is the reason why having the clocks synchronized on all servers is so vitally important. Without time synchronization, you would have a very difficult task trying to match the events shown in the CCM trace on one server with the events on another server.

Although you will never read through an SDL trace the way you read through a CCM trace, you might occasionally have to look at the SDL trace to see what triggered a particular event in the CCM trace. In some cases, you will take the timestamp for an event in the CCM trace and match up that same timestamp in the SDL trace.

Enabling SDL Trace and Setting the Appropriate SDL Trace Level

Now that you know what the SDL trace is for, you need to know how to turn on SDL tracing and set the appropriate bit mask for the data you need. A bit mask is a string of bits that each represent a particular trace setting. For example, 10010011 is a bit mask. Each 1 in the bit mask indicates that a particular trace should be enabled, while each 0 indicates that a particular trace should be disabled. Bit masks are usually represented as strings of hex digits. For example, 10010011 is 0x93.

Detailed SDL tracing consumes a lot of disk space and affects the processor on the CallManager server, which can result in performance degradation under very high call volumes. However, as of CallManager 3.3, SDL trace writing is performed asynchronously which means CallManager is allowed first access to the disk and CPU for call processing. If insufficient disk or CPU bandwidth exists to write the trace file, lines are skipped in the trace, but call processing is not affected.
SDL traces are enabled in the **Service > Service Parameter** area in CallManager Administration. Table 3-3 shows the parameters you can adjust.

**Table 3-3  SDL Service Parameters**

<table>
<thead>
<tr>
<th>SDL Service Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SdlListeningPort</strong></td>
<td>This is the TCP port with which SDL links can be established between nodes in a cluster. The port is 8002 by default. There is rarely any reason to change this value.</td>
</tr>
<tr>
<td><strong>SdlMaxRouterLatencySecs</strong></td>
<td>Indicates the maximum number of seconds of signal latency before forcing a restart of the CallManager service. The default is 20.</td>
</tr>
<tr>
<td><strong>SdlMaxUnHandledExceptions</strong></td>
<td>Specifies the maximum number of CallManager exceptions that can occur before CallManager stops running. The default is 5.</td>
</tr>
<tr>
<td><strong>SdlTraceDataFlags</strong></td>
<td>This is a bit mask used to enable the tracing of SDL non-application-specific components or to modify the behavior of SDL tracing.</td>
</tr>
<tr>
<td></td>
<td>• The recommended value for normal system debugging is <strong>0x110</strong>.</td>
</tr>
<tr>
<td></td>
<td>• The recommended value when tracking problems with SDL links is <strong>0x13D</strong>.</td>
</tr>
<tr>
<td></td>
<td>See Table 3-4 for details.</td>
</tr>
<tr>
<td><strong>SdlTraceFlushImmed</strong></td>
<td>Determines whether SDL trace entries are to be flushed to disk immediately. If this parameter is set to False, SDL trace entries are flushed to disk when there is spare disk bandwidth not being used for call processing. Setting this parameter to True causes higher disk input/output but ensures that all entries are written to the disk in the unlikely event of a software error. You should set this to False during normal operating conditions. The default is True.</td>
</tr>
<tr>
<td><strong>SdlXmlTraceFlag</strong></td>
<td>Determines whether XML-formatted tracing is allowed for SDL traces. The default is False.</td>
</tr>
<tr>
<td><strong>SdlTraceDataSize</strong></td>
<td>For signal types, this constrains the number of bytes that can be dumped from the data portion of a signal. This information appears in the freeform information column at the end of each line in the SDL trace file. The default is 100 bytes.</td>
</tr>
<tr>
<td><strong>SdlTraceFilePath</strong></td>
<td>This is the directory path that SDL uses to generate the log files. If this path is not defined or is defined incorrectly, SDL uses the default root path: C:\ProgramFiles\Cisco\Trace\SDL.</td>
</tr>
<tr>
<td><strong>SdlTraceFlag</strong></td>
<td>This is a Boolean flag that indicates if SDL tracing is enabled or disabled. Set this flag to True to turn tracing on or False to turn tracing off. The default is True.</td>
</tr>
</tbody>
</table>
Reading SDL Traces

The bit mask definitions shown in Table 3-4 correlate to the Trace Characteristics on the SDL Trace Configuration page in CallManager Serviceability (Trace > Configuration > select a server > Cisco CallManager > click the link to SDL Configuration).

### Table 3-3 SDL Service Parameters (Continued)

<table>
<thead>
<tr>
<th>SDL Service Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SdlTraceMaxLines</td>
<td>This value indicates the maximum number of lines written to a log file before a new log file is created. The default is 10,000 lines.</td>
</tr>
<tr>
<td>SdlTraceTotalNumFiles</td>
<td>This value indicates the maximum number of files that can be created for logging purposes. The default is 250 files. Normally, a maximum of 250 files is not enough because the files write in round-robin or circular fashion—when the maximum limit is met, the next file starts at the beginning and overwrites the first file. Determine how much free disk space you have, deduct a safety net of 500 MB, and then use the rest of your disk space for trace space. You can calculate the size of your trace files by figuring the average CCM trace at 10,000 lines consumes about 2.5 MB; the average SDL trace at 10,000 lines consumes about 3.5 MB. Be sure to monitor free disk space by using other tools mentioned in this chapter such as PerfMon or CCEmail.</td>
</tr>
<tr>
<td>SdlTraceTypeFlags</td>
<td>This field indicates the bit mask value for collecting the trace type flag of choice.</td>
</tr>
<tr>
<td></td>
<td>• The recommended value for normal call debugging is SdlTraceTypeFlags=0x00000B04.</td>
</tr>
<tr>
<td></td>
<td>• The recommended value for low-level debugging or the debugging of voice gateways is 0xA000EB15.</td>
</tr>
<tr>
<td></td>
<td>See Table 3-5 for more details.</td>
</tr>
</tbody>
</table>

### Table 3-4 Non-application-specific Bits

<table>
<thead>
<tr>
<th>Name</th>
<th>Trace Characteristic in CallManager Serviceability</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>traceSdlLinkState</td>
<td>Enable SDL Link States Trace</td>
<td>0x00000001</td>
<td>Enables the tracing of SDL link states.</td>
</tr>
<tr>
<td>traceSdlLowLevel</td>
<td>Enable Low-level SDL Trace</td>
<td>0x00000002</td>
<td>Enables low-level SDL tracing.</td>
</tr>
<tr>
<td>traceSdlLinkPoll</td>
<td>Enable SDL Link Poll Trace</td>
<td>0x00000004</td>
<td>Enables the tracing of SDL link poll.</td>
</tr>
</tbody>
</table>

continues
### Table 3-5  
**Non-application-specific Bits (Continued)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Trace Characteristic in CallManager Serviceability</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>traceSdlLinkMsg</code></td>
<td>Enable SDL Link Messages Trace</td>
<td>0x00000008</td>
<td>Enables the tracing of SDL Link messages.</td>
</tr>
<tr>
<td><code>traceRawData</code></td>
<td>Enable Signal Data Dump Trace</td>
<td>0x00000010</td>
<td>Enables signal data dump.</td>
</tr>
<tr>
<td><code>traceSdlTagMap</code></td>
<td>Enable Correlation Tag Mapping Trace</td>
<td>0x00000020</td>
<td>Enables the tracing of correlation tag mapping.</td>
</tr>
<tr>
<td><code>traceCreate</code></td>
<td>Enable SDL Process States Trace</td>
<td>0x00000100</td>
<td>Enables the tracing of SDL process states.</td>
</tr>
<tr>
<td><code>traceNoPrettyPrint</code></td>
<td>Disable Pretty Print of SDL Trace</td>
<td>0x00000200</td>
<td>Enables no pretty print of the SDL trace. Pretty print adds tabs and spaces in a trace file without performing post processing.</td>
</tr>
<tr>
<td><code>traceSdlEvent</code></td>
<td>Enable SDL TCP Event Trace</td>
<td>0x80000000</td>
<td>Enables TCP event traces.</td>
</tr>
</tbody>
</table>

The bit mask definitions shown in Table 3-5 correlate to the Trace Filter Settings on the SDL Trace Configuration page in CallManager Serviceability (Trace > Configuration > select a server > Cisco CallManager > click the link to SDL Configuration).

### Table 3-5  
**Bit Mask Definitions**

<table>
<thead>
<tr>
<th>Name</th>
<th>Trace Filter Setting in CallManager Serviceability</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>traceLayer1</code></td>
<td>Enable All Layer 1 Trace</td>
<td>0x00000001</td>
<td>Enables all Layer 1 traces.</td>
</tr>
<tr>
<td><code>traceDetailLayer1</code></td>
<td>Enable Detailed Layer 1 Trace</td>
<td>0x00000002</td>
<td>Enables detailed Layer 1 trace.</td>
</tr>
<tr>
<td><code>Not used</code></td>
<td>—</td>
<td>0x00000008</td>
<td>This bit is not used.</td>
</tr>
<tr>
<td><code>traceLayer2</code></td>
<td>Enable All Layer 2 Trace</td>
<td>0x00000010</td>
<td>Enables all Layer 2 traces.</td>
</tr>
<tr>
<td><code>traceLayer2Interface</code></td>
<td>Enable Layer 2 Interface Trace</td>
<td>0x00000020</td>
<td>Enables Layer 2 interface trace.</td>
</tr>
<tr>
<td><code>traceLayer2TCP</code></td>
<td>Enable All Layer 2 TCP Trace</td>
<td>0x00000040</td>
<td>Enables Layer 2 TCP trace.</td>
</tr>
<tr>
<td>Name</td>
<td>Trace Filter Setting in CallManager Serviceability</td>
<td>Bit Mask</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------</td>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>traceDetailLayer2</td>
<td>Enable Detailed Dump Layer 2 Trace</td>
<td>0x00000080</td>
<td>Enables a detailed dump of Layer 2 frames.</td>
</tr>
<tr>
<td>traceLayer3</td>
<td>Enable All Layer 3 Trace</td>
<td>0x00000100</td>
<td>Enables all Layer 3 traces.</td>
</tr>
<tr>
<td>traceCc</td>
<td>Enable All Call Control Trace</td>
<td>0x00000200</td>
<td>Enables all Call Control traces.</td>
</tr>
<tr>
<td>traceMiscPolls</td>
<td>Enable Miscellaneous Polls Trace</td>
<td>0x00000400</td>
<td>Enables miscellaneous polls traces.</td>
</tr>
<tr>
<td>traceMisc</td>
<td>Enable Miscellaneous Trace (Database Signals)</td>
<td>0x00000800</td>
<td>Enables miscellaneous traces (database signals).</td>
</tr>
<tr>
<td>traceMsgtrans</td>
<td>Enable Message Translation Signals Trace</td>
<td>0x00001000</td>
<td>Enables message translation signals TranslateIsdnToSdlReq, TranslateIsdnToSdlRes, TranslateSdlToIsdnReq, TranslateSdlToIsdnRes traces.</td>
</tr>
<tr>
<td>traceUuie</td>
<td>Enable UUIE Output Trace</td>
<td>0x00002000</td>
<td>Enables UUIE output traces.</td>
</tr>
<tr>
<td>traceGateway</td>
<td>Enable Gateway Signals Trace</td>
<td>0x00004000</td>
<td>Enables gateway signals traces.</td>
</tr>
<tr>
<td>traceCti</td>
<td>Enable CTI Trace</td>
<td>0x00008000</td>
<td>Enables CTI signal traces.</td>
</tr>
<tr>
<td>traceNetworkSvc</td>
<td>Enable Network Service Data Trace</td>
<td>0x10000000</td>
<td>Enables network service data traces.</td>
</tr>
<tr>
<td>traceNetworkEvent</td>
<td>Enable Network Service Event Trace</td>
<td>0x20000000</td>
<td>Enables network service event traces.</td>
</tr>
<tr>
<td>traceIccpAdmin</td>
<td>Enable ICCP Admin Trace</td>
<td>0x40000000</td>
<td>Enables Intracluster Control Protocol (ICCP) administration traces.</td>
</tr>
<tr>
<td>traceDefault</td>
<td>Enable Default Trace</td>
<td>0x80000000</td>
<td>Enables default traces.</td>
</tr>
</tbody>
</table>
Microsoft Performance (PerfMon)

Microsoft Performance is an administrative tool provided by the Windows 2000 operating system. It is colloquially referred to as PerfMon. PerfMon can be used to monitor a variety of performance objects. A *performance object* is a set of counters reported by a process or application running on the system that can be monitored. An example of a standard performance object is the Processor object, which contains a variety of processor-related counters such as processor utilization. PerfMon allows you to look at real-time statistics. It contains logging facilities to take snapshots of any counter at user-defined intervals.

The StatisticsEnabled service parameter in CallManager Administration (*Service > Service Parameters > select a server > Cisco CallManager*) must be set to True to generate data in the counters. If statistics are disabled, neither PerfMon nor the Real-Time Monitoring Tool (RTMT) can collect data. Statistics are enabled by default.

CallManager includes several performance objects that let you monitor various counters related to the operation of the CallManager services and associated devices. The RTMT, which is discussed later in this chapter as part of the “CallManager Serviceability” section, provides much the same functionality as PerfMon.

Comparing PerfMon and the Real-Time Monitoring Tool (RTMT)

Although PerfMon and the RTMT allow you to view the same performance objects, each tool has its strengths and weaknesses.

**PerfMon Advantages**

You can configure PerfMon to log specific counters to a comma-separated values (CSV) file. This CSV file can then be imported into a spreadsheet application for further analysis.

A third-party tool called CCEmail (discussed in the next section) allows you to configure alerts in PerfMon.

Another advantage PerfMon has over RTMT is that the RTMT web browser must always be running for the counters to be monitored and alerts to be sent. However, that will likely be fixed in releases of RTMT subsequent to release 3.3(3).

**RTMT Advantages**

You can run the RTMT from a web browser on any PC that has IP connectivity to CallManager. PerfMon can be run only from a Windows NT/2000/XP PC that has PerfMon installed.

You can configure specific counters and save the configurations in RTMT. Then, each time you run RTMT, the pre-defined configurations are available.
In addition to performance-monitoring capabilities, RTMT provides two tabs—Devices and CTI Apps. The tabs allow you to search for devices and check the state of TAPI and JTAPI applications that are connected to CallManager.

You’ll read more about the additional capabilities of the RTMT later in this chapter.

**Using PerfMon to View Real-time Statistics**

PerfMon’s most basic function is to view real-time statistics on a machine. For example, you might want to know how many calls are currently active on CallManager or the status of the channels on a PRI. This kind of information can easily be obtained through PerfMon.

An example of how to view some real-time statistics with PerfMon will better familiarize you with the tool. Launch PerfMon from a CallManager server by selecting **Start > Programs > Administrative Tools > Performance**. The application looks similar to Figure 3-5.

**Figure 3-5  PerfMon**

PerfMon has three formats to display data: chart, histogram, and report. Three buttons on the toolbar shown in Figure 3-5 are used to switch between the different formats. For viewing real-time statistics, you usually want to use the report format. Click the **View Report** button on the toolbar to gray out the area below.

Next, add the counters you want to monitor. Click the **Add** button on the toolbar, as shown in Figure 3-5. You see a dialog box similar to Figure 3-6.
From this screen, you can add counters from either the machine on which you are running PerfMon or any other machine that has IP connectivity to the machine on which you are running PerfMon. For you to be able to view counters on remote machines, the account you are logged in with must have administrator privileges on the machine you want to monitor.

Next, select the object that contains the counter(s) you want to monitor. For example, if you want to monitor the number of active calls on a CallManager, select the Cisco CallManager object.

Below the object selection you can choose which counter to monitor. If you select the Cisco CallManager object, you see a counter labeled CallsActive. Select this counter and click Add. The dialog box remains open so that you can continue selecting and adding counters. To monitor all the counters for a particular object, click the All counters button. You can click the Explain button to get a short description of the counter, as shown in Figure 3-6. When you are done adding counters, click Close. Figure 3-7 shows all the counters in the Cisco CallManager object. You can see that the counters are updated every second or so.

Appendix D, “Performance Objects and Counters,” describes the meanings of each of these counters, as well as the rest of the CallManager-related objects.

Objects and counters are only available for installed components. For example, if you do not have Cisco CallManager Attendant Console installed on the server you are trying to monitor, you will not see the Cisco CallManager Attendant Console object.
Some counters give you the option of selecting a specific instance of the counter to monitor. For example, the Cisco Lines object has only one counter, named **Active**. However, for this one counter, you can select one or more instances to monitor. In this case, each instance corresponds to a particular line on CallManager. You can select a range by holding down the Shift key and clicking the first and last instances in the range. Or you can select several individual instances by holding down the Ctrl key while clicking to select specific instances.

See Appendix D for detailed descriptions of all the Cisco CallManager-related performance objects and counters available in either PerfMon or the RTMT.

### Using Counter Logs

One of PerfMon’s most powerful features is the ability to periodically log performance information to a file. This can be useful for monitoring trends or determining exactly what time a problem occurred. Probably the most common use of counter logs is to monitor memory and CPU utilization for trends. If memory utilization continues to increase, you might be running into a memory leak. If the CPU spikes during specific times, you might be encountering one of many problems that cause high CPU utilization, such as a call routing loop.
Use the following steps to configure a counter log to monitor memory and CPU utilization on a per-process basis:

**Step 1**  On a CallManager server, open PerfMon by selecting **Start > Programs > Administrative Tools > Performance**.

**Step 2**  In the left column, in the **Performance Logs and Alerts** section, select **Counter Logs**.

**Step 3**  Select **Action > New Log Settings**.

**Step 4**  In the New Log Settings dialog box, type a name for the log, such as **CPU and Memory Logging**.

**Step 5**  A dialog box similar to the one shown in Figure 3-8 appears. Click **Add**.

**Figure 3-8**  *Counter Log Configuration Dialog Box*

![Counter Log Configuration Dialog Box](image)

**Step 6**  The Select Counters dialog box appears (previously shown in Figure 3-6). The **Processor** object and **% Processor Time** counter are selected by default. Click **Add** to add this counter to the log.

**Step 7**  In the **Performance object** field, select the **Memory** object.

**Step 8**  In the list of counters, select the **Available MBytes** counter.

**Step 9**  Click **Add** and then **Close** to return to the counter log configuration dialog box.
**Step 10** In the Sample data every area, adjust the interval depending on how often you want to monitor. For example, set it to 5 seconds to take a snapshot of the selected counters every 5 seconds. Microsoft recommends this counter not be set lower than 2 seconds when logging many different counters.

**Step 11** Select the Log Files tab.

**Step 12** In the Location field, type the path where you want to save the log files (the default is C:\PerfLogs).

**Step 13** In the File Name field, type a filename for this log file, such as CPU_and_Memory_Logging.

**Step 14** The End file names with checkbox and popup menu let you append a number to the end of the filename based on either the date and time the file is created or just an arbitrary number that increments each time a new log file is created. Select the End file names with checkbox, and choose a date format from the popup menu.

**Step 15** Change the Log file type from Binary File to Text File – CSV. With the data in a CSV format, you can take the resulting data and import it into a variety of applications. Figure 3-9 shows the Log Files tab as described in the preceding steps.

![Log Files Tab for CPU and Memory Logging in PerfMon](image)
Step 16 Click the **Schedule** tab. If this is the first log on this system, you’ll be asked if you want to create the directory, C:\PerfLogs. Click **Yes**.

Step 17 Instead of creating one huge log file, you can have PerfMon automatically create a new file periodically. To do this, in the **Stop log** area, click **After** and select how often you want to create a new file. For example, to create a new file every day, enter **1** and set the **Units** to days.

Step 18 Check the **Start a new log file** checkbox so that PerfMon starts a new file for the frequency you specified.

Step 19 Click **OK**.

If you followed the preceding instructions, you should now see a log with a green icon to the left of it on the Counter Logs screen, as shown in Figure 3-10.

**Figure 3-10  CPU and Memory Logging Counter Log in PerfMon**

If the icon is red, right-click it and select **Start**. An error message should appear, telling you PerfMon is unable to start the log. The error will likely refer you to the Windows event log to get more details on the error. Correct the error, right-click the icon again, and select **Start**.

---

**CAUTION**  When logging a small number of objects, as in the preceding example, setting the logging interval to something small such as 1 or 2 seconds is not a problem. However, if you are logging a large number of counters, be careful not to set the interval too low because this can lead to performance degradation.
CAUTION  Unlike CCM trace files, PerfMon log files do not have a way to do circular wrapping (also known as round-robin). This means that if left unattended, PerfMon log files can use up all the available hard drive space on the server, leading to a multitude of other potential problems. If you are using PerfMon logging, ensure you periodically delete old files to free up hard drive space.

Using Alerts

PerfMon allows you to configure alerts based on selected counters. For example, you can monitor the OutOfResources counter in the Cisco HW Conference Bridge Device object and be alerted when the value in that counter crosses a threshold you specify. Figure 3-11 shows some common alerts you may want to configure.

Figure 3-11  Alerts in PerfMon

Other common things you might want to set an alert for are low disk space, low available memory, the D-channel going out of service on an MGCP gateway, and the number of registered phones falling below a certain value. You can place an alert on any counter in PerfMon, so the reasons and counters you may want to monitor vary based on your system.

The default action for an alert is to log an entry in Event Viewer. You can also configure the alert to send a network message to a computer you specify, run a counter log, or have a specified program run when the alert is triggered.

The Help in PerfMon (Help > Help Topics) provides detailed configuration steps for setting alerts.
The RTMT in CallManager Serviceability also allows you to configure alerts based on counters. However, in RTMT, you can have the alert sent as a network message to a computer you specify (just like in PerfMon) or to a pager or via e-mail to a specified recipient. You can also achieve this functionality using the CCEmail tool in combination with PerfMon.

CCEmail

You can use the Windows 2000-based tool CCEmail in conjunction with PerfMon to configure alerts for the performance counters. Alerts can be sent via pager or e-mail so long as the CallManager node for which the alert is configured has connectivity to an SMTP server. Also, the pager specified to receive alerts must be capable of receiving alphanumeric pages through an e-mail.

As part of the free tools provided with this book, you can download a copy of CCEmail from the Cisco Press web site. See the “Acquiring CCEmail” section for details.

Perform the following steps to install and configure CCEmail:

**Step 1** Create a folder called `ccemail` on the C: drive of each CallManager server where monitoring is to take place.

**Step 2** Download the CCEmail program from the Cisco Press website (as specified in the “Acquiring CCEmail” section) and save it to the C:\ccemail directory on each CallManager server.

**Step 3** Open PerfMon (Start > Programs > Administrative Tools > Performance) and click on Performance Logs and Alerts and then Alerts.

**Step 4** Right-click in the window and select New Alert Settings.

**Step 5** In the Name field, type `minor`. Once you finish all the steps in this section, you are asked to repeat them again to create two alert settings, minor and major. The second time you perform this step, type `major` instead.

**Step 6** Click OK.

**Step 7** (Optional) Add a comment, such as Alert settings for minor alarms or Alert settings for major alarms.

**Step 8** Click Add. The Select Counters dialog box appears.

**Step 9** Make sure that Select counters from computer is selected and select the name of the CallManager server.

**Step 10** In the Performance object list, choose Cisco CallManager.
Step 11 With the Select counters from list button selected, select counters you want to include in the alert and click Add. Add as many counters as you want to monitor with this alert. You can press and hold the Control key while clicking on counters from the list.

Some recommended counters to include for the minor alert are
- RegisteredAnalogAccess
- RegisteredDigitalAccess
- RegisteredPhones

Some recommended counters to include for the major alert are
- CallManagerHeartBeat
- RegisteredAnalogAccess
- RegisteredDigitalAccess
- RegisteredPhones

Step 12 In the Performance object list, choose Process. Select the Private Bytes counter and the instance _Total.

Step 13 In the Performance object list, choose Processor. Select the % Processor Time counter and the instance _Total.

Step 14 Click Add and then Close.

Step 15 Click on the first counter in the Counters list and set the threshold for when to be notified. Refer to Table 3-6 for suggestions for determining thresholds.

<table>
<thead>
<tr>
<th>Counter</th>
<th>Minor Alert</th>
<th>Major Alert</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegisteredAnalogAccess</td>
<td>Under 1 less than total</td>
<td>Under 50 percent less than total</td>
</tr>
<tr>
<td>RegisteredDigitalAccess</td>
<td>Under 1 less than total</td>
<td>Under 50 percent less than total</td>
</tr>
<tr>
<td>RegisteredPhones</td>
<td>Under 20 percent of total or 50, whichever is smaller</td>
<td>Under 40 percent of total or 150, whichever is smaller</td>
</tr>
<tr>
<td>Private Bytes (1 GB Total Memory)</td>
<td>900 MB</td>
<td>950 MB</td>
</tr>
<tr>
<td>Private Bytes (512 MB Total Memory)</td>
<td>450 MB</td>
<td>500 MB</td>
</tr>
<tr>
<td>% Processor Time</td>
<td>85 percent</td>
<td>95 percent</td>
</tr>
</tbody>
</table>
Step 16 Repeat the previous step until the thresholds have been set for all counters.

Step 17 Set the interval for how often the counters should be monitored. The recommended time is 15 minutes but you can choose any interval. Figure 3-12 shows the General tab for the minor alert.

Figure 3-12  Minor Alert Settings, General Tab

Step 18 Click the Action tab.

Step 19 Select the Log an entry in the application event log checkbox.

Step 20 Select the Run this program checkbox and click Browse.

Step 21 Navigate to the C:\ccemail directory and choose the proper .bat file.

If you only want to use e-mail alerts (no paging), choose email.bat. If you only want to use paging alerts (no e-mail), choose page.bat. You can also use one method for minor alarms and the other method for major alarms. For example, major alerts always page someone and minor alerts always e-mail someone. In that case, choose page.bat for the major alerts and email.bat for the minor alerts.

Figure 3-13 shows the Action tab for the minor alert.

Step 22 Click the Schedule tab.
Step 23  In the Start scan area, make sure that At is selected.

Step 24  In the Stop scan area, make sure that After is selected and that the setting is 1 day. Also, select the Start a new scan checkbox. Figure 3-14 shows the Schedule tab for the minor alert.

Step 25  Click OK.

Step 26  The alert settings for minor alarms are now complete. Repeat Steps 4 through 25 to create the alert settings for major alarms.
Perform the following steps to configure the .bat files for CCEmail:

**Step 1** In the C:\ccemail directory, double-click **Setup.exe**.

**Step 2** The CCEmail Setup Program displays. Click **Next**.

**Step 3** In the Sender Information window, type the name of the SMTP server to be used and the sender’s e-mail address.

**Step 4** Click **Next**.

**Step 5** Confirm the settings you just entered and click **Next** again.

**Step 6** In the Email Addresses window, type the e-mail addresses, if any, to be e-mailed. For multiple addresses, separate entries with this string `-to:`.

For example: `sysadmin-alias@abc.com -to:john.smith@abc.com`

**Step 7** Click **Next**.

**Step 8** Confirm the settings you just entered and click **Next** again.

**Step 9** In the Pager Addresses window, type the pager addresses, if any, to be paged. For multiple addresses, separate entries with this string `-to:`.

**Step 10** Click **Next**.

**Step 11** Confirm the settings you just entered and click **Next** again.

**Step 12** Click **Finish**.
Alerting Methods During Production and Non-production Hours

You can configure CCEmail to automatically switch between paging and e-mailing alerts depending on the time of day. Major alerts are typically important enough that a page should be received when the alert is triggered, regardless of the hour of day. Minor alerts are still important, but they’re normally not considered critical enough to warrant a page during non-production hours, such as the middle of the night.

The following steps explain how to use .bat files and the Windows Task Scheduler so that minor alerts send pages during business hours and e-mails at night.

**Step 1**
Download the `ccemail_auto.exe` file from the Cisco Press website (as specified in the “Acquiring CCEmail” section) file and save it to the CallManager server where the changes are to be made.

**Step 2**
Double-click `ccemail_auto.exe` and make sure that the **Unzip to folder** field is set to C:\ccemail.

**Step 3**
Open the Windows Task Scheduler (Start > Settings > Control Panel > Scheduled Tasks > Add Scheduled Task).

**Step 4**
The Scheduled Task Wizard displays. Click Next.

**Step 5**
Click Browse, choose the C:\ccemail\today.bat file, and click Open.

**Step 6**
Type a name for the task (such as today), select Daily and click Next.

**Step 7**
For Start time, enter the beginning of business hours, such as 8:00 AM, select Every Day, and set the start date as today’s date.

**Step 8**
Click Next.

**Step 9**
Enter the username and password of any user that has read/write permission for the C: drive and click Next.

**Step 10**
Click Finish.

**Step 11**
Repeat Steps 3 through 10 with the following changes:
- In Step 5, choose tonight.bat instead of today.bat.
- In Step 7, enter the end of business hours instead of the beginning, such as 6:00 PM.

**Step 12**
Open PerfMon (Start > Programs > Administrative Tools > Performance) and click on Performance Logs and Alerts and then click Alerts.

**Step 13**
Double-click on minor and click on the Action tab.

**Step 14**
In the Run this program field, click Browse and choose the C:\ccemail\minor.bat file.

**Step 15**
Click OK and close PerfMon.
### Acquiring CCEmail

Check the Cisco Press website for a free downloadable file containing this tool ([www.ciscopress.com](http://www.ciscopress.com) > type 1587050757 in the Search field > click the link to Troubleshooting Cisco IP Telephony). Check the site regularly as there may also be updates to the tool or the book chapters.

**CAUTION**
This is not an officially supported tool. If you download, install, or use this tool, you do so at your own risk. Cisco Systems, Inc., is not responsible for correcting problems that may arise as a result of using this unsupported tool.

### CallManager Serviceability

CallManager Serviceability is a collection of tools that help you troubleshoot various aspects of your CIPT system. CallManager Serviceability provides end user documentation online (make a selection from the Help menu) and on Cisco.com at the following location:

```
www.cisco.com/univercd/cc/td/doc/product/voice/c_callmg/index.htm > select your CallManager release > Serviceability
```

CallManager Serviceability provides the following basic services:

- Alarms
- Tracing and the web-based Q.931 Translator
- Service Activation
- Control Center
- Real-Time Monitoring Tool

### Alarms

Options under the Alarm menu let you configure the destination for the alarms ([Alarm > Configuration](#)) and search for alarm message definitions ([Alarm > Definitions](#)). Alarms are messages that notify you of basic errors. The messages can be inserted into CCM (SDI) and SDL traces and the Windows Event Viewer.

CallManager Serviceability provides pre-defined alarms, set at pre-defined levels. Use the Alarm Configuration page to set up which level of alarm you want to receive and where you want those alarms sent. Table 3-7 describes the alarm event levels.
Table 3-7  
*Alarm Event Levels*

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emergency</td>
<td>This level designates the system as unusable.</td>
</tr>
<tr>
<td>Alert</td>
<td>This level indicates that immediate action is needed.</td>
</tr>
<tr>
<td>Critical</td>
<td>This level indicates that CallManager detects a critical condition.</td>
</tr>
<tr>
<td>Error</td>
<td>This level signifies that an error condition exists.</td>
</tr>
<tr>
<td>Warning</td>
<td>This level indicates that a warning condition is detected.</td>
</tr>
<tr>
<td>Notice</td>
<td>This level designates a normal but significant condition.</td>
</tr>
<tr>
<td>Informational</td>
<td>This level designates information messages only.</td>
</tr>
<tr>
<td>Debug</td>
<td>This level designates detailed event information used for debugging.</td>
</tr>
</tbody>
</table>

**Tracing**

Options under the Trace menu let you configure trace levels and parameters for CallManager, Database Layer, CTI Manager, and other core services. CallManager Serviceability then allows you to collect this information from one node or all nodes in the cluster when necessary. You can select trace information at the device level to target one or more specific devices in the trace output. The log files generated by the trace can be .txt format or XML-enabled for detailed analysis. We have already discussed text-based and XML-based tracing in the previous sections “Reading CCM Traces” and “Reading SDL Traces.” In this section, we focus on XML-formatted tracing.

**Using XML-enabled Traces**

Using XML-enabled traces can make reading traces easier. Earlier in this chapter, you learned how to read trace files in their raw, text-based format and saw comparisons of standard text-based tracing and XML-formatted tracing. In some cases, you might find it easier to use XML-enabled traces and let the system do the searching for you. With XML-based tracing, you are given certain trace filters that you can apply to the trace output. Also, you can have the system search and compile trace information on just the devices you need (this feature is also available for standard text-based tracing). We don’t normally recommend XML-based tracing because of a 2000-line limit in the trace files and the fact that searching through a large number of trace files is very slow. If you are searching a large amount of data, for instance, a problem that occurred over the course of several hours or several days, depending on the size of your system, you will probably need to use text-based tracing instead of XML-based tracing. If you select trace collection criteria that causes the trace output to be larger than XML-based tracing can handle, a message displays in CallManager Serviceability advising you that trace analysis cannot be performed on files.
greater than 2 MB in size. You are then given the option to save the result without filtering using **File > Save As** in your web browser.

The previous section, “Reading CCM Traces” detailed the steps used to configure XML-enabled tracing. Once trace files are collected, XML-based tracing allows you to filter the trace output using Trace Analysis.

**Searching for Devices with XML Traces**

Using the Trace Analysis feature, you can search for devices in the XML traces and narrow the scope of the trace search by choosing different search criteria and display fields. Instead of scanning thousands of lines of trace files looking for an event here and there, the XML-enabled trace can compile only the lines that relate to the devices or analysis criteria you have selected. In other words, all trace file lines you aren’t looking for are filtered out so that you can concentrate on what is important to your search. Trace Analysis was discussed in detail in the previous section, “Reading CM Traces.”

**Web-based Q.931 Translator**

The Trace menu in CallManager Serviceability provides a link to a web-based interface to the Q.931 Translator. Although convenient, the web-based interface may slow the performance of the Q.931 Translator. See the “Q.931 Translator and Enhanced Q.931 Translator” section for more information about this tool.

**Service Activation**

Service Activation in CallManager Serviceability (**Tools > Service Activation**) lets you activate and deactivate CallManager services. You can activate or deactivate CallManager-related services from Automatic mode by selecting or deselecting checkboxes next to specific services and then clicking the **Update** button. Then you can use the Control Center or the Windows Services Microsoft Management Console (**right-click My Computer and select Manage > double-click Services and Applications > click Services**) to start and stop services.

**CAUTION**

If you need to deactivate a service, you should do so from the Service Activation page. If you deactivate services from the Service Control Manager, you get an error message saying that some of the services are not configured properly. This is because deactivating services from the Service Control Manager does not remove the entries from the CallManager database; therefore, the services are out of sync with the configured services in the CallManager database.
Control Center

The Control Center in CallManager Serviceability (Tools > Control Center) lets you start and stop CallManager services and view their activation status. You can stop and start services in the Windows Services Microsoft Management Console as well if you have local access to the server. If not, you can use the Control Center web page in CallManager Serviceability to do the same thing.

Real-Time Monitoring Tool (RTMT)

The RTMT is a web-based application that provides up-to-the-second information about the state of a CallManager cluster, including run-time information about CallManager and CallManager-related components such as IP phones and gateways. This includes dozens of counters on items such as call activity, trunk usage, and even memory and processor utilization.

Appendix D describes the meanings of each of the CallManager-related performance objects and counters, as well as some commonly used Windows counters.

The RTMT provides much the same functionality as PerfMon in an easy-to-use web-based tool. See the section “Comparing PerfMon and the Real-Time Monitoring Tool (RTMT),” for more information about the differences the two tools provide. You may find yourself using PerfMon for some tasks and RTMT for others. The StatisticsEnabled service parameter in CallManager Administration (Service > Service Parameters > select a server > Cisco CallManager) must be set to True to generate data in the counters. If statistics are disabled, neither the RTMT nor PerfMon can collect data. Statistics are enabled by default.

One very useful feature not provided by PerfMon (unless you use a third-party tool such as CCEmail described in this chapter) is the ability to configure alerts based on the objects. These alerts can be set to send a notification via pager, e-mail, or system message popups. These alerts can be configured for any performance object and can be set to trigger if a counter is greater or less than a specific threshold. The alerts can be configured to run only at specific times or all the time. Alerts are useful when you are troubleshooting problems because you can set the RTMT to alert you when a specific event you are trying to troubleshoot occurs. For example, if you are running out of channels on a gateway interface, you can set the RTMT to page or e-mail you when the counter for total calls to that gateway exceeds a certain value. Through release 3.3, the RTMT web browser must remain open and running for alerts to be sent. This requirement may change in future releases of CallManager Serviceability.

Objects and counters are only available for installed components. For example, if you do not have Cisco CallManager Attendant Console installed on the server you are trying to monitor, you will not see the Cisco CallManager Attendant Console object.
Performance Tab

The Performance tab in RTMT allows you to view CallManager cluster info as shown in Figure 3-15. To view this information, right-click on the cluster name and select **Properties**. The CallManager Cluster Info window displays basic statistic about the cluster, broken down by server.

![Figure 3-15 RTMT, Cluster Info](image)

This is also the tab where performance monitoring occurs, much the same functionality as with PerfMon. However, in RTMT, you can configure tabs for specific counter configurations. Each time you use RTMT, the pre-defined configurations are displayed. Figure 3-16 shows several counters used to monitor general activities on a system.

To add a new category, right-click on one of the existing tabs and select **New Category**. Building the category is as easy as selecting a counter and then dragging and dropping it onto the tab’s frame. Multiple counters can be piled in a single frame, and six frames per tab are provided.

Devices Tab

The Devices tab in RTMT allows you to view device information that you configure for various device types—phones, gateways, H.323 devices, CTI applications, voice mail, and Cisco IP Voice Media Streaming Application devices such as MOH servers, MTP resources, and conference bridges. The Devices tab is shown in Figure 3-17.
Figure 3-16  RTMT, Performance Tab

Figure 3-17  RTMT, Devices Tab
To add a new category, click on one of the existing tabs and select **New Category**. You build the category by first double-clicking on one of the device types and then making selections in a series of wizard-like screens that display, as shown in Figure 3-18 and 3-19.

**Figure 3-18 Device Search Criteria Screens, Part 1**

![Device Search Criteria Screens, Part 1](image1)

**Figure 3-19 Device Search Criteria Screens, Part 2**

![Device Search Criteria Screens, Part 2](image2)

The screens shown in Figure 3-18 and 3-19 allow you to search for real-time information about devices in the cluster regardless of their registration status. You can search for devices based on device name, IP address, DN, IP subnet, and so on.

**CTI Apps Tab**

The CTI Apps tab in RTMT allows you to view application-, device-, and line-based information for CTI applications. You can check the registration status of TAPI and JTAPI applications such as Personal Assistant, Cisco IP Manager Assistant, Cisco IP SoftPhone, and so on. The CTI Apps tab shows whether the application has an open connection with CallManager and who’s using it. The CTI Apps tab also shows you applications attempting to log in using an invalid username or password. Figure 3-20 shows the CTI Apps tab.

To add a new category, right-click on one of the existing tabs and select **New Category**. You build the category by first double-clicking on one of Applications, Devices, or Lines in the left frame and then making selections in a series of wizard-like screens that appear.
Call Manager can be configured to store CDRs for all calls generated throughout a CallManager cluster. Typically CDRs are used for billing and accounting purposes, but they can also be useful when troubleshooting certain types of problems. Usually CDRs do not provide enough information to diagnose a problem, but they can help you narrow down a problem and provide information about the specific time when a problem occurred, leading you to other trace files or debugging tools that might give you more details on the problem’s root cause.

CDRs are stored in their own SQL database on the Publisher server in the CallManager cluster. For CallManager to generate CDRs, you must set the `CdrEnabled` service parameter on each CallManager to True (select `Service > Service Parameters`). This service parameter is set to False by default. For additional details, you should also enable the `Call Diagnostics Enabled` service parameter. If the `Call Diagnostics Enabled` service parameter is set to True, CallManager also generates Call Management Records (CMRs, also known as Diagnostic CDRs). CMRs contain data such as packets sent and received, packets lost, and jitter for the duration of the call. One CDR might have multiple CMRs associated with it because each media stream creates a CMR. This means that if a call is placed on hold and then resumed, two CMRs are created—one for the media stream before
the call is on hold, and one for the media stream after the call is on hold. CMRs are especially useful for diagnosing voice quality problems because they allow you to see patterns. For example, you might notice that all the phones across a specific WAN link are experiencing high jitter or packet loss. This can indicate a possible quality of service (QoS) misconfiguration or line errors on the WAN link.

The CDR Analysis and Reporting (CAR) tool (CallManager Serviceability > Tools > CDR Analysis and Reporting) can help you analyze the raw data that comprises the CDR database and create reports based on your search criteria. For example, if you are receiving complaints of poor voice quality, one of the first things you should do is find out which phones or gateways are experiencing the poor voice quality. Although you can wait to collect data from additional user complaints, you can proactively use the data in the CDRs and CMRs to identify any trends in high jitter or packet loss to hone in on where the problem is.

For example, if a user tells you they had a problem calling someone, you can use CAR to search for the call in the CDRs. This gives you the time the call occurred, which helps you when you examine the CCM traces related to the problem.

You can search CDRs by user or specific extensions for the period that you specify. This helps you trace calls placed from specific extensions for diagnostic or informational purposes. All associated records, such as transfer and conference calls, appear together as a logical group.

CAR can also be used to send you alerts if the number of calls with poor QoS is exceeded or if the CDR database size exceeds a percentage of the maximum number of records.

For more information on the various features available in CAR, review the CAR section in the CallManager Serviceability documentation at


If you look at the raw timestamp information stored in the CallManager CDRs, you notice they are stored in a format that is not easily recognizable in standard date and time format. To quickly convert the date and time from the format in the CDRs to standard format, use the CDR Time Converter utility.

**CDR Time Converter**

Timestamps in the CallManager CDRs are stored in a format known as Epoch time. This is the number of milliseconds that have elapsed since midnight January 1, 1970 GMT. While this might be a convenient format for things like computers, humans usually prefer a more readable date and time format.
Event Viewer

The CDR Time Converter utility allows you to enter the time as stored in a CDR—for example, 1030565084—and convert it to standard date and time format—such as 8/28/2002 3:04:44 PM. Figure 3-21 shows the output of the CDR Time Converter tool.

Figure 3-21  CDR Time Converter Tool

<table>
<thead>
<tr>
<th>UNIX Epoch Time Converter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer From CDR Record</td>
</tr>
<tr>
<td>Date and Time in Greenwich Mean Time</td>
</tr>
<tr>
<td>Date and Time in Local Time Zone (Standard)</td>
</tr>
<tr>
<td>Date and Time in Local Time Zone (Daylight)</td>
</tr>
</tbody>
</table>

Notice the tool converts the number in Epoch time to Greenwich Mean Time (GMT) and the local time zone of the PC on which the tool is installed for both standard and daylight savings time.

Acquiring the CDR Time Converter

Check the Cisco Press website for a free downloadable file containing this tool ([www.ciscopress.com](http://www.ciscopress.com) > type 1587050757 in the Search field > click the link to Troubleshooting Cisco IP Telephony). Check the site regularly as there may also be updates to the tool or the book chapters.

CAUTION  This is not an officially supported tool. If you download, install, or use this tool, you do so at your own risk. Cisco Systems, Inc., is not responsible for correcting problems that may arise as a result of using this unsupported tool.

Event Viewer

Microsoft Event Viewer is a Windows 2000 Server application that displays system, security, and application events (including CallManager) for the Windows 2000 Server. These events are alarm messages generated by CallManager. CallManager Serviceability is used to configure alarm messages to be sent to the Event Viewer ([Alarm > Configuration](http://www.ciscopress.com)).
Open Event Viewer on the server running CallManager by clicking `Start > Settings > Control Panel > Administrative Tools > Event Viewer`. CallManager errors are logged in the Application log. You can double-click an event in the log to learn more about it.

Alarm definitions can be found in CallManager Serviceability (`Alarm > Definitions`).

Figure 3-22 shows an example of an alarm message in Event Viewer.

![Event Viewer](image)

Notice the App ID and the Error message. This tells you that the IP phone specified in the Device Name details has unregistered from CallManager.

All alarms fall into seven catalogs, as shown in Table 3-8.

<table>
<thead>
<tr>
<th>Catalog Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CallManager</td>
<td>All CallManager alarm definitions such as CallManagerFailure, DChannelOOS, DeviceUnregistered, and RouteListExhausted.</td>
</tr>
<tr>
<td>TFTPAlarmCatalog</td>
<td>All Cisco TFTP alarm definitions such as kServingFileWarning and kCTFTPCConnectSendFileTimeoutOccurred.</td>
</tr>
</tbody>
</table>
Based on the information given in the Event log entry in Figure 3-22 you can go to CallManager Serviceability to search for a better definition of the problem in question.

To search for alarm definitions, perform this procedure:

**Step 1** In CallManager Administration, choose **Application > Cisco CallManager Serviceability**. The Cisco CallManager Serviceability window appears.

**Step 2** Choose **Alarm > Definitions**.

**Step 3** Choose the catalog of alarm definitions from the **Find alarms where** drop-down box, or click the **Enter Alarm Name** field to enter the alarm name. Figure 3-23 shows this interface.
Step 4  Click the **Find** button. The definitions list appears for the alarm catalog or search string you entered.

Step 5  In the list, click the alarm definition for which you want alarm details. Based on the Event Viewer entry shown in Figure 3-22, choose the CallManager catalog and the DeviceUnregistered error. You see Figure 3-24, showing the severity, an explanation, and the recommended action. All alarm messages found in the Event Viewer can be researched in this fashion.
Q.931 Translator and Enhanced Q.931 Translator

Q.931 Translator is an application that takes a CCM trace and decodes the hex-formatted Q.931 messages into human-readable format. Depending on the version of CallManager you are using, there are two ways to access the application. Q.931 Translator is bundled with every CallManager installation. It can be found in the C:\Program Files\Cisco\bin directory. The file is called Q931 translator.exe. In CallManager version 3.2 and later, the application is also part of the CallManager Serviceability web page (select Trace > Q931 Translator).

Q.931 Translator is useful for quickly troubleshooting a variety of gateway problems. It is not a substitute for learning how to read the CCM trace, but it helps you resolve some gateway signaling problems without ever having to look in the CCM trace.

The name Q.931 Translator is a bit misleading because this application does more than just translate ISDN messages into a human-readable format. The Q.931 Translator also helps you troubleshoot problems with hardware that does not use the Q.931 protocol, including the WS-X6624 analog FXS card, T1 CAS on the WS-X6608 card, and calls to and from an
H.323 gateway. The Translator is helpful for troubleshooting problems with these hardware components because some of these gateways convert their native signaling to Q.931 messages. H.225 messages used in the H.323 protocol are based on the Q.931 specification; hence, the Q.931 Translator can decode H.225 messages.

The amount of information contained in a CCM trace can be intimidating. Q.931 Translator helps you quickly examine a trace in a graphical format without having to wade through irrelevant debugs. For example, the following sample is from a CCM trace of an outgoing call setup:

```
Out Message -- Pri250SetupMsg -- Protocol= Pri250Protocol
Ie - Ni2BearerCapabilityIE IData= 04 03 80 90 A2
Ie - 0931ChannelIdIE IData= 18 03 A9 83 94
Ie - 0931DisplayIE IData= 28 0C B2 56 71 65 76 20 47 69 72 61 6C 74
Ie - 0931CallingPartyIE IData= 6C 0C 21 80 39 31 39 35 35 35 36 34 34
Ie - 0931CalledPartyIE IData= 70 0D A1 39 31 32 31 30 35 35 35 25 30 30
MMAn_Id= 0. (iep= 0 dsl= 0 sapi= 0 ces= 0 IpAddr=346812ac IpPort=2427)
IsdnMsgData2= 08 02 05 C8 05 04 03 80 90 A2 18 03 A9 83 94 28 0C B2 50 61 75
  6C 20 47 69 72 61 6C 74 6C 0C 21 80 39 31 39 35 35 35 36 34 34 70 0D A1
  39 31 32 31 30 35 35 35 32 35 30 30
```

Without a copy of the ITU Q.931 specification, which explains each bit in the trace information elements, this sample looks like a bunch of numbers and letters. However, the Q.931 Translator decodes the output into a readable format, as shown in Figure 3-25.

**Figure 3-25 Q.931 Translator Application**

One thing you should know right away is not to trust the Direction column. Information in the Direction column can occasionally be inaccurate because of how the tool decodes hex messages. To be sure of the direction, look at the first line in the debug, which states Out
**Q.931 Translator and Enhanced Q.931 Translator**

**Message.** This indicates that this is an outbound (TX) setup message. Similarly, an inbound (RX) message would show In Message in the CCM trace.

Calls in the trace can be distinguished by the call reference value. The most significant bit (MSB) of the call reference toggles between 0 and 1, depending on whether the message was inbound to CallManager or outbound from CallManager to the gateway. Table 3-9 shows the difference between the first bit being 1 versus being 0. This table also shows the binary representation of the hexadecimal digits to clearly show how the last three binary digits in the left and right columns are identical. The only difference between the left and right columns is that the MSB for all the digits in the left column is 0, and the MSB for all the digits in the right column is 1.

**Table 3-9**

*Binary Representation of Hexadecimal Digits*

<table>
<thead>
<tr>
<th>MSB = 0</th>
<th>MSB = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (0000)</td>
<td>8 (1000)</td>
</tr>
<tr>
<td>1 (0001)</td>
<td>9 (1001)</td>
</tr>
<tr>
<td>2 (0010)</td>
<td>A (1010)</td>
</tr>
<tr>
<td>3 (0011)</td>
<td>B (1011)</td>
</tr>
<tr>
<td>4 (0100)</td>
<td>C (1100)</td>
</tr>
<tr>
<td>5 (0101)</td>
<td>D (1101)</td>
</tr>
<tr>
<td>6 (0110)</td>
<td>E (1110)</td>
</tr>
<tr>
<td>7 (0111)</td>
<td>F (1111)</td>
</tr>
</tbody>
</table>

The output in Figure 3-25 shows that the call reference is **0x05C8** in the transmit (TX) direction and **0x85C8** in the receive (RX) direction. Any row that has a 0x05C8 or 0x85C8 in the Call Reference column is a message for the same call. As Table 3-9 shows, 0 and 8 are equivalent, with the exception of the MSB, which is different. Call reference values are eventually reused, but it is impossible for the same call reference to appear in a single CCM trace file for one gateway because several hundred or thousand calls must occur before the value is reused.

Q.931 Translator shows the calling and called party numbers converted to easily readable text, along with the bearer channel identifier, bearer capability, and display information element. The channel identifier and bearer capability are not fully decoded. You need a copy of the Q.931 specification to understand what the various bits in those fields signify. Don’t worry if you don’t have this specification. Chapter 6 explains how to decode some of these. Some of these are decoded for you in the Enhanced Q.931 Translator explained in the next section.

Q.931 Translator is most useful for decoding cause codes that are sent as an information element (IE) in various Q.931 messages. These cause codes are always sent as part of a DISCONNECT message and may be included as part of other messages as well. Decoding the cause code IE gives you some insight into why the call was disconnected.
The following is a sample of a CCM trace showing a Q.931 DISCONNECT message:

```
Out Message -- PriDisconnectMsg -- Protocol= Pri250Protocol
Ie - Q931CauseIe IEData= 08 02 80 81
MMan_Id= 0. (iep= 0 dsl= 0 sapi= 0 ces= 0 IpAddr=a86a12ac IpPort=2427)
IsdnMsgData2= 08 01 AA 45 08 02 80 81
```

The first thing you should notice is that this is a DISCONNECT message being sent by CallManager to the PSTN. You can see that the hex representation of the data in the Q931CauseIe is **08 02 80 81**. These cause codes are defined in the ITU Q.850 specification; however, Q.931 Translator decodes these for you. Opening the trace file containing the DISCONNECT message in Q.931 Translator reveals the following:

```
DISCONNECT pd = 8  callref = 0xAA
Cause i = 0x8081 - Unallocated/unassigned number
```

You can see that 0x8081 is decoded to **Unallocated/unassigned number**, meaning that CallManager does not have a phone or route pattern that matches the digits that were sent to CallManager by the ISDN network.

---

**NOTE**

See Chapter 6 for additional information about what various cause codes mean and how to continue troubleshooting problems like these.

---

**Enhanced Q.931 Translator**

Although the Q.931 Translator allows you to quickly observe Q.931 and H.225 events, its functionality is somewhat limited. Two Cisco TAC engineers took the original source code for Q.931 Translator and enhanced it to provide additional functionality not available from the official Q.931 Translator product bundled with CallManager.

The Enhanced Q.931 Translator offers the following advantages over the standard Q.931 Translator:

- **Direction column is correct** — As mentioned earlier, the method by which the standard Q.931 Translator decodes the direction is flawed, and therefore, the direction column is sometimes incorrect. The Enhanced Q.931 Translator properly decodes the messages as **In** or **Out**.
- **Protocol column** — Tells you whether the message is a Q.931 message or an H.225 message.
- **Expanded IE decoding** — Decodes far more Q.931 information elements than the original Q.931 Translator, including bearer capability, channel ID, numbering plan and type in the calling and called party number IEs, call state, and many more.
- **Find in messages** — Allows you to search for any text that appears in the decoded message data. This means you can search for a calling or called party phone number, disconnect cause code, or any other value and quickly find the message you are looking for.
• **Filter messages**—Allows you to filter messages based on a specific call reference or protocol type. When filtering on call reference, Enhanced Q.931 Translator automatically includes all messages for that call reference regardless of the setting of the MSB of the call reference.

• **Raw ISDN message data**—After the decoded information, the raw hex bytes for the ISDN message are presented.

• **Resizable window**—Allows you to resize the window to view more messages.

• **File name display**—The currently-open filename is displayed in the title bar in case you forget where you left off when searching through several traces in a folder.

Figure 3-26 shows the same trace shown in Figure 3-25, this time as it looks in the Enhanced Q.931 Translator.

**Figure 3-26 Enhanced Q.931 Translator Application**

Notice the amount of additional data presented in the bottom pane of the tool. The bearer capability (ITU-T standard, Speech, Circuit mode, 64k, µ-law), channel ID (PRI interface, Exclusive channel 20), calling party numbering plan and type (Plan: ISDN, Type: National, Presentation Allowed, User-provided, not screened), and called party numbering plan and type (Plan: ISDN, Type: National) are all decoded for you automatically.
Future versions of CallManager may include the Enhanced Q.931 Translator, but until then, you must download the tool from the Cisco Press website (see “Acquiring Enhanced Q.931 Translator” later in this chapter).

One thing to remember is that the Q.931 Translator and Enhanced Q.931 Translator tools are not a replacement for the CCM trace. They are just another tool you can use to make decoding the CCM trace easier and finding the location of your problem quicker. Often you will have to refer back to the CCM trace after finding the call in Q.931 Translator to get additional detail regarding the events that surround a message. For example, if you find a message in Q.931 Translator indicating CallManager disconnected a call with a cause code of 0xAF, “Resources Unavailable,” Q.931 Translator tells you the cause code and the timestamp in the CCM trace. You must then go to the CCM trace at that timestamp and determine what resource was unavailable that caused CallManager to disconnect the call.

As mentioned before, Q.931 Translator decodes more than just Q.931 messages. The H.225 protocol used to communicate between H.323 gateways and other CallManager clusters uses messages similar to ISDN Q.931. Q.931 Translator translates H.225 messages that appear in CCM traces.

Some gateways use Q.931 to communicate with CallManager even though the gateway’s interface to the PSTN is not actually using Q.931. For example, the WS-X6624 24-port FXS gateway uses analog FXS signaling to communicate with analog phones, fax machines, and modems; however, this analog signaling is converted to Q.931 messages between the gateway and CallManager. These Q.931 messages appear in the CCM trace and are translated by the Q.931 Translator. The same is true of the WS-X6608-T1 gateway when communicating with the PSTN using channel associated signaling (CAS). The gateway converts CAS to Q.931 messages.

So if the gateways are converting these various protocols to Q.931, you might wonder how you can troubleshoot the signaling before it is converted to Q.931. This is one of the various uses of the Dick Tracy tool discussed in the section “Dick Tracy.”

**Acquiring Enhanced Q.931 Translator**

Check the Cisco Press website for a free downloadable file containing this tool ([www.ciscopress.com > type 1587050757 in the Search field > click the link to Troubleshooting Cisco IP Telephony](http://www.ciscopress.com)). Check the site regularly as there may also be updates to the tool or the book chapters.

---

**CAUTION**

This is not an officially supported tool. If you download, install, or use this tool, you do so at your own risk. Cisco Systems, Inc., is not responsible for correcting problems that may arise as a result of using this unsupported tool.
The Dick Tracy tool is a complex and powerful tool used to troubleshoot problems on various gateways based on the Skinny or MGCP protocols. Specifically, the Dick Tracy tool is used on the following voice gateways:

- DT-24+
- DE-30+
- WS-X6608-T1
- WS-X6608-E1
- WS-X6624

There is little to no documentation on the Dick Tracy tool because it was created as an internal development tool. However, it has been released under the condition that the tool itself is unsupported. This means that no formal documentation or release mechanism exists for the tool. This also means that the tool’s behavior might change from one release to another without warning, but as long as you understand the basics of how the tool works, you should be able to pick up any changes without too much difficulty. The “Acquiring Dick Tracy” section explains how you can download a copy of the Dick Tracy tool.

There are two versions of Dick Tracy:

- A standalone Windows 95/98/NT/2000/XP application that connects to the gateway you want to monitor over TCP/IP. This version is commonly called Dick Tracy.
- A version is embedded in the Catalyst 6000 operating system (CatOS). It can be invoked from the Catalyst 6000 command-line interface (CLI) to diagnose problems on WS-X6608 and WS-X6624 gateways in the chassis. This version is commonly called CLI Tracy or embedded Tracy.

The CLI Tracy tool can be used to connect only to gateways that are in the same Catalyst 6000 chassis to which you are connected. Using the Windows-based Dick Tracy tool is recommended here because it is more flexible and easier to use than CLI Tracy.

Why do you need the Dick Tracy tool? In case you’ve never seen a Cisco DT-24+ gateway, it is a PCI card that plugs into a PCI slot in any PC chassis. The gateway only uses the PCI slot to draw power. The PCI bus is not used for any kind of data transfer to or from the DT-24+. The DT-24+ also provides an Ethernet port and a T1 port. There is no console port or other method of out-of-band communication to the gateway; therefore, you need a tool to access the gateway so that you can determine what is going on inside the gateway. The WS-X6608 and WS-X6624 are similar to the DT-24+ because they use the Catalyst 6000 chassis they reside in for power and IP connectivity. Other than that, the Catalyst 6000 Supervisor (the management interface on the Catalyst 6000) has no out-of-band management interface to the gateways.
CAUTION Before we discuss the tool itself, you should know that the Dick Tracy tool is not the most user-friendly piece of software. It also can damage your gateway if it’s not used properly. A good working rule is that if you don’t know what something does, you probably shouldn’t mess with it—particularly most of the set commands that are available. This book covers a few useful set commands, but other than those, you should not need to use Dick Tracy to set commands on the gateways.

Using the Dick Tracy Tool

Figure 3-27 shows the Windows version of the Dick Tracy tool. The tool itself is a very simple program. Clicking the Connect button on the toolbar reveals a box indicating Connect to remote target device. Enter the IP address of the gateway you want to troubleshoot in the Target IP Address box. The port number should always be 2005.

After you connect to a gateway, you see a text box labeled Live trace on ip address, where ip address is your gateway’s IP address. You can open additional connections to other gateways using the same procedure. You will likely see several lines of tracing when you connect. This is because the gateway buffers the last few lines of tracing and shows them to you when you connect.
After you connect to a gateway, it is a good idea to enable logging of the live trace. Click **Options > Start Logging**. A dialog box appears, asking where you want to save the logged traces. Choose a convenient location on your hard drive that has enough space available. The traces are just a text log of everything you see on your screen while connected to the gateway. They usually do not get very big unless you are running a debug for an extended period of time. Logging does not affect the performance of Dick Tracy or the gateway. Enabling logging keeps a record of all the traces you capture until you close the trace window or stop logging using the menu.

At this point, you’ve started a trace, and the tool is up and running. On the menu bar you see a one-line text box and a **Send** button. Think of this field as the *command box* to communicate with the gateway. To communicate with the gateway, you must understand the concept of task IDs. Each gateway has various tasks that are responsible for various components of the software. For example, one of the tasks is responsible for sending and receiving messages to and from the digital signal processors (DSPs) on the gateway.

To view the list of available tasks on a gateway, enter the command **0 show tl** (this command uses the number 0, not the letter O, and the letter L, not the number 1) in the command box and click **Send** or press **Enter**. You should see something similar to the following sample:

```
03:15:15.450 (GEN) Lennon Tasks
  0 : GEN
  1 : AUD
  2 : TRC
  3 : SNMP
  4 : SPAN
  5 : NMP
  6 : DSP
  7 : LINE
  8 : CFG
  9 : GMSG
 10 : SOCK
 11 : TMR
 12 : Q921
 13 : XA
```

The 0 (zero) is the number of the task ID you want to issue a command to. In this case, you are issuing a command to the GEN task (ID 0). Each task has various commands that can be issued to it. Some tasks do not respond to commands at all. The **show tl** portion is the command being issued to task ID 0.

Dick Tracy offers context-sensitive help for the tasks that accept commands. For example, to see which commands are available for task 0, enter **0 ?** in the command box and press **Enter**. You see something similar to the following sample:

```
03:22:21.320 GEN   --> Help -> Available Commands:
03:22:21.320 reset <hard/soft>
03:22:21.320 read func[index] <count>
03:22:21.320 write func[index] <value>
03:22:21.320 show func
03:22:21.320 clear func
03:22:21.320 set func
```
You can see that `show` is one of the available commands for task ID 0. You can get additional context-sensitive help by entering `0 show ?`. You see the following:

```
03:24:31.500 GEN   Help: show modifiers ->
03:24:31.500 Show Task List : show tl
03:24:31.500 Show Ethernet Stat's : show ether
03:24:31.500 Show Version's : show ver
03:24:31.500 Show Time : show time
03:24:31.500 Show Reset : show reset
```

You can see that the command `show tl` performs the command **Show Task List** according to the context-sensitive help. It is difficult to provide a list of the available task IDs and Tracy commands because they vary from gateway to gateway and can change from one version of CallManager to the next. Chapter 6 covers some specific task IDs when discussing troubleshooting gateways.

You should always precede each command with a task ID. If you do not precede a command with a task ID, the gateway applies the command to the last task ID you sent a command to.

In addition to the various `show` commands, a few other commands are useful. One in particular is `set mask`. You use the `set mask` command to turn on various debugs. To view what debugs you can turn on for a particular task, issue the `show mask` command for the task in question. For example, issuing the command `6 show mask` details the various trace bits for the DSP task:

```
(DSP) Mask<0x0>
   Where Bit0 = Debug Msg's
         Bit1 = Call Progression Msg's
         Bit2 = Boot Msg's
         Bit3 = Stat Msg's
         Bit4 = Cmd Msg's
         Bit5 = RTP Msg's
         Bit6 = SID Frames
         Bit7 = Status Msg's
```

As you can see from this output, the mask is currently set to `0x0`, and eight different trace bits can be enabled. The mask is set as a hexadecimal digit. In this case because you have eight bits, 256 possible combinations of tracing can be turned on, depending on which bits are enabled. For example, to enable Debug Msg’s (Bit0) and Call Progression Msg’s (Bit1), the mask must be set to `00000011` in binary. This translates to `0x03` in hex. Therefore, the command is `6 set mask 0x03`.

**CAUTION** Be sure to set the masks back to `0x0` after enabling any trace masks. The traces continue to run until the gateway is reset, even if you close your Dick Tracy session. Failure to turn off the debug masks could create a performance impact on the gateway.

For the time being, don’t worry about what each of these trace masks or task IDs does. Chapter 6 covers the masks and IDs in detail.
Using the CLI Tracy/Embedded Tracy Tool

The less-used version of Dick Tracy is the CLI Tracy or embedded Tracy tool, available on the Catalyst 6000 CLI. To enable tracing for a particular port, enter the command `tracy_start module port`, where `module` and `port` are the module and port numbers of the port you want to debug. For example, for the gateway on port 6/3, you would enter

```
tracy_start 6 3
```

Notice that the syntax differs from the traditional Catalyst Operating System (CatOS) notation, in which ports are specified using `module/port`.

After you start CLI Tracy for a port, all the debug information for that port appears on your console or Telnet session. You can send Tracy commands to the port using the command `tracy_send_cmd module port taskid command`. When you need to send commands to a port, you should really use the Windows-based tool because sending commands via CLI Tracy can lead to switch instability. When you are done using the CLI Tracy tool, enter the command `tracy_close module port` to end the session. You can have only one CLI Tracy session open at any given time on the whole chassis. So if someone has a Tracy session open on the console port, you cannot start one from a Telnet session. Be sure to close the CLI Tracy session when you are done.

CLI Tracy does have one advantage over the Windows-based Dick Tracy tool: It can monitor a port before it obtains an IP address. For that reason, you can use CLI Tracy to troubleshoot problems where the port cannot obtain an IP address. The regular Dick Tracy tool cannot accomplish this because you need IP connectivity to the port in question before you can gather information from the port.

Acquiring Dick Tracy

Check the Cisco Press website for a free downloadable file containing this tool (www.ciscopress.com > type 1587050757 in the Search field > click the link to Troubleshooting Cisco IP Telephony). Check the site regularly as there may also be updates to the tool or the book chapters.

CAUTION

This is not an officially supported tool. If you download, install, or use this tool, you do so at your own risk. Cisco Systems, Inc., is not responsible for correcting problems that may arise as a result of using this unsupported tool.
Sniffer Traces

One of the most powerful tools for troubleshooting a large number of CallManager problems is a packet capture/analyzer tool such as Network Associates’ Sniffer Pro (www.nai.com), Finisar Surveyor (www.finisar.com/product/product.php?product_id=104), or Ethereal (www.ethereal.com). Because of Sniffer Pro’s widespread appeal, a trace file generated by any packet-capture tool is commonly called a sniffer trace. This term is used here to refer to any packet-capture software, not just Sniffer Pro.

A sniffer trace lets you see exactly what is happening on the network at any given time. Examining a sniffer trace requires a good understanding of the various layers of the OSI model, which were briefly described in the “Introduction” to this book.

To get the most benefit from a packet-capture tool, you should use a tool that can decode the various protocols you might encounter in an IP telephony network. This includes, but is not limited to, H.323 (H.225 and H.245), MGCP, RTP, SQL, and LDAP. Also extremely important is the capability to decode Skinny packets. Most newer versions of commercially available network capture application can decode Skinny. Finisar Surveyor can decode Skinny as part of the standard package while Network Associates Sniffer Pro requires you purchase the Sniffer Voice add-on to get Skinny decodes. Also, the free protocol analyzer Ethereal supports decoding Skinny as of version 0.8.20.

This book does not teach you how to use the network analysis software, but instead focuses on the kind of information you can obtain using network analysis software. Sniffer traces are most important when you’re troubleshooting problems that can’t be examined using standard trace files or debugs because the problem is either network-related or the appropriate diagnostic tool is not built into the product. For example, device registration problems are much easier to troubleshoot with a sniffer trace, as are most voice quality problems.

Voice Codec Bandwidth Calculator

Use the Voice Codec Bandwidth Calculator to determine the bandwidth used by different codecs with various voice protocols over different media.

To get a detailed analysis of all the headers for your particular medium, use the automated Voice Codec Bandwidth Calculator available on Cisco.com. You must be a registered user on Cisco.com to access the tool.

http://tools.cisco.com/Support/VBC/jsp/Codec_Calc1.jsp

Bug Toolkit (Formerly Bug Navigator)

Cisco provides a bug search feature that allows you to find known bugs based on software version, feature set, and keywords. The resulting matrix shows when each bug was integrated or fixed, if applicable. It also allows you to save the results of a search in Bug Groups and also create persistent Alert Agents that can feed those Groups with new defect alerts.
Bug Toolkit is only available to registered users on Cisco.com. Access the bug toolkit by searching for “bug toolkit” on Cisco.com or at the following link:

www.cisco.com/cgi-bin/Support/Bugtool/launch_bugtool.pl

Remote Access Tools

There are several tools you can use to remotely access your system. In this chapter, we briefly discuss Terminal Services and Virtual Network Computing (VNC).

Terminal Services

Windows Terminal Services is a feature that comes standard on all servers running Windows 2000 Server software. Terminal Services allows you to remotely access the Windows interface on a CallManager server.

Terminal Services is extremely useful for remotely troubleshooting problems on CallManager when you do not have immediate access to the console. To use Terminal Services, you must install the Terminal Services Client on any PC running most Microsoft Windows operating systems.

The installer for the Terminal Services Client is included on all CallManager servers in the C:\WINNT\system32\clients\tsclient folder. There are several folders for the 32-bit and 16-bit versions. Use the 32-bit client on any PC running Windows 95 or later.

Once installed, launch the Terminal Services Client and enter the IP address of CallManager and the screen resolution you want to connect with. You receive a login prompt for the server. Enter the administrator username and password to authenticate. Once authenticated, you have access to the CallManager desktop almost as if you were on the console.

If you need to open a hole through your firewall to access CallManager via Terminal Services, open TCP port 3389. This is the only port needed to access a Windows 2000 Server via Terminal Services.

CAUTION

Cisco does not support installing any CallManager applications, CallManager software patches, or operating system patches via Terminal Services. Terminal Services is designed for remote access to the CallManager server for troubleshooting purposes; however, some portions of the CallManager installer are not compatible with Terminal Services. You can use VNC to access the console of the server for performing upgrades and patches.
Virtual Network Computing (VNC)

VNC is basically a remote display system that allows you to view a remote desktop environment. VNC allows you to use one computer to drive actions on a target computer but differs from Terminal Services because, with VNC, any actions performed by you that occur on the target computer can be seen equally by the local user. All computers must have a local copy of VNC installed. You can use VNC to install, upgrade, or apply patches to CallManager.

You can access the VNC application and documentation files on the OS version 2000 2.2 and later CD or download. If you’re running an older version of the OS, run the OS upgrade for version 2000 2.2 or later to gain access to the VNC files. OS upgrades are available on CCO at the following link:

www.cisco.com/cgi-bin/tablebuild.pl/cmva-3des

**CAUTION** Using VNC can expose you to a security risk. Please review the “Security Best Practices” section in the Cisco-produced document for installing VNC, which is available on the OS 2000 version 2.2 and later CD or at the download link previously shown.

Websites and Further Reading

There’s a wealth of information available on the Internet. When you’re looking for more information, use the following resources:

- **Cisco Technical Assistance website**—Provides the latest information and technical documentation from Cisco’s Technical Assistance Center (TAC). Use the TAC website to search for tech tips and documentation, download software updates, or open a TAC case to obtain additional information. Access the TAC website by searching for “TAC” on Cisco.com or at
  www.cisco.com/tac

- **Technical tips for IP Telephony applications, servers and associated technologies**—This site contains a variety of technical documents written by TAC engineers that are useful for solving commonly encountered problems. Access the IP Telephony Applications website by searching for “IP Telephony Applications” on Cisco.com or at
  www.cisco.com/pcgi-bin/Support/browse/index.pl?i=Technologies&f=1533

- **Networking Professionals Connection**—This site is the gathering place for networking professionals to share questions, suggestions, and information about networking solutions, products, and technologies. Access the forum by searching for “networking professionals” on Cisco.com or at
  www.cisco.com/go/netpro
• **AnswerMonkey**—You’ll find detailed information about Cisco Unity at the home page of one Unity’s creators, Jeff Lindborg. Access the site by searching for “answermonkey” on Google.com or at www.answermonkey.net

• **Updates to this book**—Check the Cisco Press website regularly for updated information pertaining to the chapters in this book ([www.ciscopress.com](http://www.ciscopress.com) > type 1587050757 in the Search field > click the link to Troubleshooting Cisco IP Telephony).

  Check out the section “Further Reading” in the “Introduction” to this book for additional books about IP Telephony and VoIP.

---

**Best Practices**

• Become familiar with the various troubleshooting tools at your disposal. Be sure to try each of them before you encounter a problem to understand how they work so you do not have to waste time learning the tool when under pressure to resolve a network outage.

• Keep a copy of all the tools in a centralized location and check frequently for updates that may add additional functionality.

• The only way to become efficient at reading CCM traces is by practicing. The more you look through CCM traces to troubleshoot problems, the better you will understand the intricacies of the CCM trace.

• Ensure remote access to your CallManager cluster is available before a problem occurs. If you need to be able to access the CallManager cluster from outside the office, make provisions for VPN or dialup access and use Terminal Services or VNC to access your CallManagers. Terminal Services works amazingly well even over slow dialup connections.

**VNC Best Practices**

• If you’re using VNC and no longer plan to use Terminal Services for remote management, disable Terminal Services.

• Set the VNC service to Manual startup and start it only during remote management. This adds another layer of protection by requiring that users access the environment via Windows username/password authentication to start the VNC service.

• Use a complex alphanumeric password for VNC. VNC does not have a username/password structure; it only uses a single password, so make sure the password you choose is difficult to crack. VNC limits the password to eight characters. A good password includes numbers, upper- and lowercase letters, and special characters and does not use any known word. For example, 123eye67 is not as good a password choice as 4hW9Lv#g.
Summary

This chapter covered the various troubleshooting tools and resources you have at your disposal to troubleshoot a CIPT network. Which tool you use depends largely on the problem at hand. However, some problems can be resolved using more than one tool. As you become more familiar with each of the tools in your tool belt, you will begin to favor some tools over others for specific tasks. No matter how much you read about these tools, you will not learn about them until you use them to troubleshoot a real problem on your own. As you advance through the rest of this book, you will see frequent references to the tools presented in this chapter because they play an integral part in CIPT troubleshooting.
This page intentionally left blank
Symbols

! wildcard, 460
. wildcard, 461
@ wildcard, 461
   DDIs, 487–494
   route filters, 506–507
   multiple clauses, 512
   NANP tags, 508–510

Numerics

3-port switch operation, Cisco 79xx series IP phones, 161–164
7-digit local calls, delayed routing, 466–469
10-10-Dialing DDI, 487
10-10-Dialing Trailing-# DDI, 487
11/10D->7D DDI, 487
11/10D->7D Trailing-# DDI, 487
11D->10D DDI, 488
11D->10D Trailing-# DDI, 488
802.1Q protocol, 850
911 routing, Cisco ER, 478
6608 T1/E1 module
   configuring, 325–337
   D-channel establishment, 337, 340–343
   advanced troubleshooting, 344–359
   T1 CAS, 359–367
6624 Port FXS Analog Interface Module
   configuring, 367–379
7960/7940 IP Phones
   extension mobility, 756–758
   configuring, 758–763
   login/logout process, 763–765
   resolving common problems, 765–768, 772
79xx IP Phones
   3-port switch operation, 161–164
   network settings, 123–126

A

AA (Auto Attendant), 737
   traces, collecting, 748–752
AAR (automatic alternate routing), 478
   acknowledgments, 238
ACOM (combined loss), 417
   acoustic echo, isolating, 412
   acquiring
      Dick Tracy tool, 105
      Q.931 Translator, 100
active connections, 155
AD (Active Directory)
   Customer Directory Configuration plugin,
      troubleshooting, 839–844
   LDAP integration, 837–839
Ad Hoc conferences, 565
   error messages, 597–598
   locations-based CAC bandwidth reservations,
      633–635
adjusting
   fax relay data rate, 451–452
   interdigit timeout, 467
Administrative Reporting Tool (ART), 795
   alarms
      configuring on CallManager Serviceability, 82
      StationAlarmMessage field definitions,
         158–160
alerts
   configuring on CCEmail, 81
   enabling on PerfMon, 75
algorithmic delay, 386
   “Already In Conference” error messages,
      troubleshooting, 597
analog gateways, VG248 SMDI integration,
   686–692
Analog Ground Start, 850
Analog Loop Start, 850
analyzing collected data
   case study, 18–19
   CCM traces, 42, 50–57
      through MGCP T1 PRI gateways, 58–60
   CMI traces, 674–679
deductive reasoning, 11–12
ISDN traces, 258–262
calling name display, 270
cause codes, 262–269
numbering type/plan mismatches, 269–270
timer information, 271–276
locations-based CAC trace information, 628–631
SDL traces, 60–63
verifying IP network integrity, 12–13
Anlagenanschluss, 213
ANS (answer tone), 439
ANSI (American National Standards Institute) website, 849
appearances, held calls, 524
applications
CallManager Serviceability, 82
alarms, 82
Control Center, 85
RTMT, 85–88
Service Activation, 84
traces, configuring, 83
CAR, 90
CCC, 790–791
CCEmail
alerting methods, 81
configuring, 76–80
Cisco Attendant Console, 779–780
client, 781–782
resolving common problems, 782–784
server components, 780–781
Cisco AVVID IP Telephony, 34
Cisco ER, 791
Cisco IP SoftPhone, 786–788
Cisco Personal Attendant, resolving call routing problems, 785–786
CRA, 736
CTI, 736
Dick Tracy, 101–104
CLI/embedded Tracy, 105
directory-enabled, 819–820
Enhanced Q.931 Translator, 98–100
Event Viewer, 91
alarm definitions, 92–93
PerfMon
alerts, 75
counter logging, 71–75
versus RTMT, 68–69
viewing real-time statistics, 69–71
Q.931 Translator, 95–97
VNC, 108
Windows Terminal Services, 107
applying transformations
cumulative effect, 497–499
order of application, importance of, 496–497
area codes
blocking, 548–549
versus local area code, 510
ART (Administrative Reporting Tool), 795
assigning calling search spaces to devices, 471–473
audio sources (MOH), 601–603
Audio Translator, troubleshooting, 618
live
selecting recording input, 620
troubleshooting, 619–620
multicast
troubleshooting, 616
versus unicast, 615
Audio Translator, troubleshooting, 618
automatic alternate routing (AAR), 478
automatic time synchronization, configuring on CallManager servers, 39
auto-registration, controlling with PLAR, 545–546

B
backhauling, 553
on MGCP PRI gateways, 256–258
backup CallManager, 154
BackupCallManagerName parameter (CMI), 667
bandwidth requirements, locations-based CAC, 624–626
BaudRate parameter (CMI), 667
best practices, troubleshooting Cisco IP Phones, 165–166
binary values, converting to decimal and hexadecimal values, 881–889
bit masks, 63
configuring for SDL traces, 63–67
blind transfers, 529–531
blocked calls, 473
blocking area codes, 548–549
buffering delay. See queueing delay
Bug Toolkit, 106–107
busy calls, forwarding, 480

C

CAC (call admission control), 24, 623
gatekeeper CAC, 638
call setup, 647–651
CallManager registration, 645–647
RAS messages, 639
verifying configuration, 640–645
locations-based, 623–624
bandwidth requirements, 624–626
call preservation, 636–637
CCM traces, enabling, 626
conference bandwidth reservations, 633–635
configuring, 630
detecting bandwidth leaks, 635–636
location identifiers, 628
MOH bandwidth reservations, 631–633
regions, 627–629
trace information, analyzing, 628–631
call admission control. See CAC
call control, CCAPI debug commands, 196–205
call forwarding, 479
CFA, 480–485
restricting, 546–547
CFB, 480
CFF, 485–486
CFNA, 479–480
to voice mail, reading CMI traces, 678
call history information messages (SMDI), 665
call hold feature (CallManager), 522–529
call legs, 175. See also dial peers
call park feature (CallManager), 531
troubleshooting, 532–533
call pickup feature (CallManager), troubleshooting, 533–538
call preservation
locations-based CAC, 636–637
SRST, 562
troubleshooting, 561
call routing
called party transformations, effect on, 513–514
Cisco Personal Attendant, 785–786
closest-match routing, 461–464
unexpected outside dial tone, troubleshooting, 465–466
dial peers
call legs, 175
destination-pattern parameter, 176–179
incoming called number command, 181–184
matching, 175
optional parameters, 179–181
NANP, 857–879
pattern matching
blocked calls, 473
multiple partitions within calling search space, 474–475
problem resolution methodology, 515–516
reading CCM traces, 516–521
route patterns
urgent priority, 502
wildcards, 460–461
toll fraud, preventing, 544–549
translation patterns, 501–506
call setup, gatekeeper CAC, 647–651
call statistics menu (Cisco IP Phones), 165
call transfer feature (CallManager), 529–531
called party transformations
effect on call routing, 513–514
masks, 495–496
cumulative effect of changes, 497–499
order of application, 496–497
overriding, 499
CallerID service parameter transformation, 500
calling party transformations, 513–514
masks, 495–496
cumulative effect of changes, 497–499
order of application, 496–497
overwriting, 499
calling search spaces, 469–473
AAR, 637
applying to voice mail systems, 547
call forwarding, 479
CFA, 480–485, 546–547
CFB, 480
CFF, 485–486
CFNA, 479–480
device-level, 476–477
event-specific, 478
line-level, 476–477
multiple partitions, pattern-matching rules, 474–475
CallManager. See also CallManager Serviceability
audio sources
  multicast versus unicast, 615
  selecting recording input, 620
  troubleshooting live sources, 619–620
call hold feature, 522–529
call park feature, 531–533
call pickup feature, 533–538
call processing messages, 140, 144–147
call transfer feature, 529–531
calling IP phone interaction, 150
Cisco AVVID IP Telephony call processing, 24
centralized deployment model, 26
distributed deployment model, 27
multiple-site deployment model, 25
single-site deployment model, 24
closest-match routing, 461–464
unexpected outside dial tone,
  troubleshooting, 465–466
Database Layer Monitor
  CDR replication, troubleshooting,
    813–815
  verifying operation, 812–813
delayed routing, 466–469
digit analysis behavior, 463–464
embedded LDAP directory, 823–825
  logon failures, troubleshooting, 827
  reconfiguring on Publisher server,
    828–835
  reconfiguring on Subscriber server,
    835–837
epipoints, 551
MOH, troubleshooting, 611–615
nonsurvivable endpoints, 557
CTI/TAPI endpoints, 559
H.323 gateways, 558–559
Skinny gateways, 557
object counters, 893–898
  Cisco CallManager Attendant Console
    object counters, 898–900
  Cisco CallManager System Performance
    object counters, 900–902
  Cisco CIT Manager object counters, 903
partitions, 470
survivable endpoints, 500–501
TOH, 602
  investigating instances of, 617
trace files
  analyzing SCCP messages, 148–154
call state field values, 525
  configuring in CallManager
    Serviceability, 42–50
digit analysis results, 149
  fields, 44–46
  for MGCP T1 PRI gateways, 58–60
  MOH, troubleshooting, 608–611
  reading, 42, 50–57
  reviewing for call routing problems,
    516–521
unregistered IP Phones, troubleshooting
  checking inline power, 114–117
  verifying network connectivity, 117–127
CallManager Serviceability, 82
alarms, 82
  configuring CCM traces, 42–50
Control Center, 85
  RTMT, 85
    CTI Apps tab, 88
    Devices tab, 86
    Performance tab, 86
Service Activation, 84
  traces, configuring, 83
CallManagerName parameter (CMI), 667
capability bits, 200
  DTMF relay, 202
  fax, 201
capturing IP IVR/AA traces, 748–752
CAR (CDR analysis and reporting), 90
case studies
  data analysis, 18–19
  data collection, 14–18
Catalyst 4000 series switches
   AGM, hardware conferencing, 587
Catalyst 4224 switch, voice gateway functionality, 173–174
Catalyst 6000 series
   6608 T1/E1 modules
      configuring, 325–337
      D-channel establishment, 337, 340–359
   6608/6624 voice gateway modules
      DHCP, troubleshooting, 314–320
      powering up, 313–314
      registration, troubleshooting, 324–325
      TFTP, troubleshooting, 320–324
   6624 FXS Analog Interface Module, configuring, 367–379
   CMM switch, voice gateway functionality, 174
CatOS switches, time synchronization, 41
CCAPI (call control application programming interface) debugs, 196–205
CCC (Cisco Conference Connection), 790–791
CCEmail
   alerting methods, 81
   configuring, 76–80
CCMAdmin
   reset command, 156
   restart command, 156
CDCC (Call Dependent Call Control) processes, tracing locations-based CAC, 626
CDR Time Converter, 90–91
CDRs (call detail records), 89
   CAR, 90
      configuring Subscriber replication, 810–812
      replication, troubleshooting, 813–815
      storing in Publisher server, 795
      timestamps, 90–91
   centralized CallManager architecture, locations-based CAC, 26, 623–624
   AAR, troubleshooting, 637
   analyzing, 628–631
   bandwidth requirements, 624–626
   call preservation, 636–637
   CCM traces
      analyzing, 628–631
      enabling, 626
   conference bandwidth reservations, 633–635
      configuring, 627–631
      detecting bandwidth leaks, 635–636
      MOH bandwidth reservations, 631–633
      regions, 627
   CFA (call forward all), 480–485
      restricting, 546–547
   CFB (call forwardbusy), 480
   CFF (call forward on failure), 485–486
   CFNA (call forward no answer), 479–480
   CgpnScreeningIndicator service parameter transformation, 500
   choppy voice quality, sources of
      packet drops, 397–400
      queuing delay, 401
      VAD, 402–404
Cisco 7910 IP Phone, 32
Cisco 7914 IP Phone Expansion Module, 32
Cisco 7935 IP Conference Station, 32
Cisco 7960/7940 IP Phones, 31
Cisco Attendant Console, 779–780
   client, 781–782
   resolving common problems, 782–784
   server components, 780–781
Cisco AVVID IP Telephony
   applications, 34
   call processing
      centralized deployment model, 26
      distributed deployment model, 27
      multiple-site deployment model, 25
      single-site deployment model, 24
   clients, 29–31
      Cisco 7910 IP Phone, 32
      Cisco 7914 IP Phone Expansion Module, 32
      Cisco 7935 IP Conference Station, 32
      Cisco 7960/7940 IP Phones, 31
   IP Telephony infrastructure, 23–24
   network infrastructure, 23
   voice gateways, 32
Cisco CallManager Administration, viewing Route Plan Report, 466
Cisco CallManager Attendant Console object counters, 898–900
Cisco CallManager System Performance object counters, 900–902
Cisco CIT Manager object counters, 903
Cisco Customer Directory Configuration plugin, troubleshooting installation, 839–843
Cisco DPA 7630 voice mail gateway, 702
  Octel voice mail system integration with CallManager, 693, 697–703
    MWI problems, troubleshooting, 702
    port statuses, 700–702
    verifying cabling, 693
Cisco ER (Emergency Responder), 478, 791
Cisco Gatekeeper object counters, 904
Cisco H.323 object counters, 904
Cisco HW Conference Bridge Device object counters, 905
Cisco IOS Software
debugs, enabling, 185–187
dial peers
call legs, 175
destination-pattern parameter legs, 176–179
incoming called number command, 181–184
matching, 175
optional parameters, 179–181
Cisco IOS voice gateways, 169
  2600 series routers, 171–172
  3600 series routers, 172
  3700 series routers, 173
digital interfaces
  ISDN PRI signaling, 210–214
  T1 CAS, 214–218
timestamps, configuring, 185
verifying physical layer connectivity, 208–210
eliminating sources of echo, 421–424
H.323, 281
  H.225 call flow, 288–294
  H.225 signaling, 283–284
  H.245 call signaling, 295–307
  IEs, 284–287
MGCP
  cause codes (traces), 262–269
  commands, 219–221
  DTMF packages, 231–232
  DTMF trunk packages, 236–237
  endpoint identifiers, 218–219
  FXO/FXS signaling, 249–256
generic media packages, 231
  handset emulation packages, 235–236
  line packages, 234–235
  MF packages, 232–238
  packages, 229–230
  parameter lines, 221–229
  PRI backhaul, 256–258
  numbering type/plan mismatches, 269–270
  reading ISDN traces, 258–262
  response codes, 239–240
  response headers, 238
  RTP packages, 236
  T1 CAS, 276–281
timers, 271–276
  trunk packages, 233
  verifying registration status, 240–249
resolving one-way/no-way audio problems, 407–410
TDM interfaces, troubleshooting, 187
  with debug commands, 192–205
  with show commands, 187–192
VG200, 170
Cisco IP Phone Services SDK, 822
Cisco IP Phones
  79xx series
    3-port switch operation, 161–164
    call processing messages, 140, 144–148
    network settings, 123–126
    SCCP, troubleshooting, 139–140
active connections, 155
best practices for troubleshooting, 165–166
directory problems, troubleshooting, 160–161
dropped calls, troubleshooting, 157
failback, 156
failover, 155
  troubleshooting, 158–160
resetting, 156
restarting, 156
service problems, troubleshooting, 160–161, 789
Skinny client registration process
  messages, 127–132
  verifying with status messages, 133–135
soft keys, 147
TCP handle, deriving from CCM traces, 148
Temporary Failure messages, 561–562
Cisco IP SoftPhone, 786–788
  eliminating sources of echo, 428–429
Cisco IP/VC products, 789
Cisco Lines object counters, 905
Cisco Locations object counters, 906
Cisco Media Streaming App object counters, 906–909
Cisco Media Termination Point object counters, 909–910
Cisco Messaging Interface object counters, 910–911
Cisco MGCO FXI Device object counters, 911
Cisco MGCO FXS Device object counters, 912
Cisco MGCP Gateways object counters, 912
Cisco MGCP PRI Device object counters, 913–914
Cisco MGCP T1 CAS Device object counters, 914–915
Cisco MOH Device object counters, 915–918
Cisco MTP Device object counters, 916
Cisco Personal Attendant, resolving call routing problems, 785–786
Cisco Phones object counters, 918
Cisco SW Conference Bridge Device object counters, 918–920
Cisco TFTP object counters, 920–923
Cisco Transcoder Device object counters, 923
Cisco Unity, 655
  DTMF, 661–662
  MWI, 659–661
  switch configuration, verifying, 658–659
  troubleshooting resources, 662
  TSP
    compatibility, verifying, 655–656
    configuring, 656–657
Cisco WebAttendant. See Cisco Attendant Console
Cisco WS-X6608 gateway, eliminating sources of echo, 424–427
CLI Tracy, 105
clients
  Cisco 7910 IP Phone, 32
  Cisco 7914 IP Phone Expansion Module, 32
  Cisco 7935 IP Conference Station, 32
closest-match routing, 461–464
unexpected outside dial tone, troubleshooting, 465–466

clusters
  database replication, Publisher-Subscriber model, 793–796
  intercluster trunks, 311
  codec mismatches, 312
  master/replica relationship, 823
  passwords, configuring on nodes, 798–802
“CM Down, Features Disabled” message (Cisco IP Phones), 158
CMI (Cisco Messaging Interface), 666–667
  service parameters, 667–671, 674
  traces, reading, 674–679
  troubleshooting with HyperTerminal, 679–682
CMRs (Call Management Records), 89
codecs
  CallManager selection process, 568
  capability bits, 200
  configuring between regions, 569
  GSM, 855
  transcoding, 565
  wideband, 855
coder delay, isolating, 386
collecting data, 4
analyzing, 11
  CCM traces, 42, 50–57
  CMI traces, 674–679
  ISDN traces, 258–262
  locations-based CAC traces, 628–631
  SDL traces, 60–63
case study, 14–18
IP IVR/AA traces, 748–752
isolating root cause of problems, 6
deductive reasoning, 11–12
earliest occurrence of problem, referencing device-based time, 10–11
  with topology information, 7–9
user information, 10
verifying IP network integrity, 12–13
comfort noise, 402
commands
  debug ephone, 713
  debug ephone detail, 723–728
  debug ephone register, 714–717
  debug ephone state, 719–722
  debug vstp tone, 192–196
fax interface-type, 454
fax nsf, 453
fax rate, 451
fax-relay ecm disable, 452
frame-clock-select, 210
incoming called number, 182–184
show call active voice, 191, 404
show call active voice brief, 449
show ephone, 720
show gatekeeper calls, 649
show gatekeeper endpoints, 645
show gatekeeper zone status, 649
show voice port summary, 188–189
conferencing
  Ad Hoc, error messages, 597–598
  failures, troubleshooting, 592–597
configuration parameters, VG248, 686–690
configuring
  6608 T1/E1 digital gateway, 325–337
  D-channel establishment, 337, 340–359
  T1 CAS, troubleshooting, 359–367
  6624 FXS Analog Gateway, 367–379
  CallManager Serviceability
    alarms, 82
    CCM traces, 42–50
    Service Activation, 84
    traces, 83
  CCEmail, 76–80
  CMI, service parameters, 667–671, 674
codes between regions, 569
CRA, LDAP directories, 741–745
dial peers, 176–177
  incoming called number command, 181–184
  optional parameters, 179–181
  variable-length matching, 178–179
extension mobility, 758–763
fax/modem passthrough on WS-X6608 port, 441
locations-based CAC, 627–631
MWI, parameters, 682–685
passwords on cluster nodes, 798–802
regions, 571
SDL traces, 63–67
SRST, 709–712
  DHCP support, 732
  transfer patterns, 730
Subscriber CDR replication, 810–812
connectivity
  troubleshooting unregistered Skinny clients, 117–127
    configuration files, 121–127
    IP addressing, 118–121
    VLAN configuration, 118
  verifying, 12–13
Control Center (CallManager Serviceability), 85
converting decimal values
  to binary, 881–889
  to hex, 881–889
CoR (class of restriction), 708
corporate directories
  Cisco IP phone directory integration, 820
  LDAP integration
    with Active Directory, 837–839
    with Netscape iPlanet, 844
  providing endpoint access, 821–823
  troubleshooting, 823
counters
  Cisco analog access, 892
  Cisco CallManager Attendant Console object, 898–900
  Cisco CallManager object, 893–898
  Cisco CallManager System Performance object, 900–902
  Cisco CTI Manager object, 903
  Cisco Gatekeeper object, 904
  Cisco H.323 object, 904
  Cisco HW Conference Bridge Device object, 905
  Cisco Lines object, 905
  Cisco Locations object, 906
  Cisco Media Streaming App object, 906–909
  Cisco Media Termination Point object, 909–910
  Cisco Messaging Interface object, 910–911
  Cisco MGCP FXO Device object, 911
  Cisco MGCP FXS Device object, 912
  Cisco MGCP Gateways object, 912
  Cisco MGCP PRI Device object, 913–914
  Cisco MGCP T1 CAS Device object, 914–915
  Cisco MOH Device object, 915–918
Cisco MTP Device object, 916
Cisco Phones object, 918
Cisco SW Conference Bridge Device object, 918–920
Cisco TFTP object, 920–923
Cisco Transcoder Device object, 923
enabling logging on PerfMon, 71–75
Windows 2000 objects, 924–925
CRA (customer response application), 736
  AA, 737
  compatibility with CCM, verifying, 737
  extension mobility
    configuring, 759–763
    login/logout process, 763–765
    resolving common problems, 765–768, 772
  LDAP directory, configuring, 741–745
CRA Administration, 738–741
  engine status, verifying, 745–748
CTI (Computer Telephony Interface) applications, 736
  CRA
    AA, 737
    extension mobility, 759–769, 772
    IVR, 737
    LDAP directory, configuring, 741–745
  CRA Administration, troubleshooting, 738–741
  CTI Manager, 738
  nonsurvivable CTI/TAPI endpoints, 559
  verifying TSP version, 736
CTI Apps tab (RTMT), 88
CTI Manager, 738
CTIQBE (Computer Telephony Interface Quick Buffer Encoding), 736
  cumulative transformations, 497–499
Customer Directory Configuration plugin, troubleshooting installation, 839–843

D

data analysis, 4–5
  case study, 14–19
  CCM traces, 42, 50–57
    through MGCP T1 PRI gateways, 58–60
  CMI traces, 674–679
  deductive reasoning, 11–12
  ISDN traces, 258–262
    calling name display, 270
    cause codes, 262–269
    numbering type/plan mismatches, 269–270
    timer information, 271–276
  isolating root cause, 6
    with topology information, 7–9
  locations-based CAC trace information, 628–631
  SDL traces, 60–63
  user information, 10
  verifying IP network integrity, 12–13
Database Layer Monitor
  troubleshooting CDR replication, 813–815
  verifying operation, 812–813
database replication
  name resolution, 796–797
  passwords, changing, 798–802
  Publisher-Subscriber model, 793–796
  troubleshooting with Microsoft SQL Server Enterprise Manager, 802–803
DataBits parameter (CMI), 667
DC Directory, 823–825
  logon failures, troubleshooting, 827
  reconfiguring
    on Publisher server, 828–835
    on Subscriber server, 835–837
DC Directory Administrator, launching, 826
DDIs (digit discard instructions), 486–494
  alternate expansion of DDI acronym, 486
debugs
  enabling on Cisco IOS Software, 185–187
  SRST call control, 719–720
decimal values, converting to hexadecimal and binary values, 881–889
deductive reasoning, 11–12
default MRGL, 568
dejitter delay, isolating source of, 393–395
delay, 384. See also echo
effect on signaling, 395–396
isolating sources of
dejitter delay, 393–395
fixed delay, 385–389
variable delay, 390–395
delayed routing, troubleshooting, 466–469
Delayed Start (E&M), 850
deriving TCP handles of Cisco IP Phones from CCM traces, 148
destination-pattern parameter (dial peers), 176–179
developing phone services, 789
DEVICE_RESET message (SCCP), 157
DEVICE_RESTART message (SCCP), 157
device-level calling search spaces, 476–477
devices
Cisco IOS Software gateways
eliminating sources of echo, 421–424
resolving one-way/no-way audio problems, 407–410
Cisco IP SoftPhones, eliminating sources of echo, 428–429
codecs
CallManager selection process, 568
capability bits, 200
configuring between regions, 569
G.711, 855
G.723, 855
G.726, 855
G.729, 855
G.729a, 855
G.729ab, 855
G.729b, 855
transcoding, 565
echo cancellers, 384
operation of, 416–418
fax machines, 433
encoding schemes, 434
fax/modem passthrough, 437–439
isolating problems, 449–450
jitter, 446
negotiation, 434
NSF field, modifying, 453
packet loss, 446
page transmission, 434
passthrough, 440

physical layer errors, troubleshooting, 447–449
switching fax protocol, 454
T.30 transmissions, 435–437
media resources, 565
modems
passthrough, 439
physical layer errors, troubleshooting, 447–449
MOH fixed audio sources, verifying configuration, 619–620
time synchronization
CatOS, 41
Cisco IOS, 40–41
transcoders, 571–577
out-of-resource conditions, 578–580
with conference bridge resources, 581–585
with MOH servers, 585
Devices tab (RTMT), 86
DHCP (Dynamic Host Configuration Protocol), 850
troubleshooting on 6608/6624 modules, 314–320
dial peers
answer-address command, 182
call legs, 175
callID assignment, 199–200
capabilities, 200–203
inbound, disconnects, 204
tear down, 204–205
destination-pattern parameter, 176–179
inbound peer matching, 181–182
based on port configuration, 183
incoming called-number command, 181–184
interdigit timeout, 177–178
longest-match routing, 177
optional parameters, 179–181
outbound peer matching, 182–183
peer ID 0, characteristics, 183–184
POTS, 175
as destination, 179
optional parameters, 180–181
priority, assigning, 177
session-target ipv4 command, 179
temporary dial peers, viewing, 719
variable-length destination patterns, 178
viewing configuration with CCAPI debugs, 197–200
VoIP, 175
Dial Plan Path service parameter
transformation, 501
dailing forest traces, 538–542
verbose mode, 543
dailing transformations. See transformations
DialingPlan parameter (CMI), 667
Dick Tracy, 101–104
acquiring, 105
CLI/embedded Tracy, 105
digit analysis, 461–464
CCM trace results, 149
dailing forest traces, 538–542
verbose mode, 543
identifying potential route pattern matches
within partitions, 517–521
digital integration, Octel voice mail systems and
CallManager, 693, 698–700
digital interfaces
ISDN PRI signaling, 210–212
configuring, 213–214
physical layer, verifying connectivity, 208–210
T1 CAS, 214–218
directories, LDAP
access, 820
integration, 820
schema, 819
Directories button (Cisco IP Phones), request
failures, 160–161
directory-enabled Cisco applications, 819
disabling ECM on fax relay, 452–453
disconnected FXO interfaces, troubleshooting, 205
displaying
e-phone-dn configuration, 718
SRST polling statistics, 722
distributed CallManager architecture, gatekeeper
CAC, 623, 638
call setup, 647–651
CallManger registration, 645–647
RAS messages, 639
verifying configuration, 640–645
distributed deployment model, CallManager, 27
distribution agent, 804
DNS name resolution in database replication, 798
DPA voicemail gateway
event logging, 703
Octel/CallManager integration, 693, 697–703
MWI problems, troubleshooting, 702
port statuses, 700–702
verifying cabling, 693
dropped calls, 551–552
media processing resources, 560
troubleshooting, 157, 561
DSPs (digital signal processors), codec
complexity, 171
DT-24+/DE-30+ gateways, eliminating sources of
echo, 424–427
DTMF (Dual-Tone MultiFrequency) tones,
importance to voice mail systems, 661–662
DTMF packages (MGCP), 231–232
trunk packages, 236–237
DTMF relay (H.245), 303–307
Dynamic Host Configuration Protocol. See DHCP

E
E&M Delayed Start, 850
echo
eliminating sources of, 418–429
perception of as problem, 414–416
sources of, isolating, 411
acoustic echo, 412
electrical echo, 411–412
echo cancellers, 384
operation, 416–418
ECM (error control mode), disabling on fax relay,
452–453
EIA/TIA web site, 849
EIGRP (Enhanced Interior Gateway Routing
Protocol), 850
electrical echo, isolating, 411–412
eliminating
possible causes using deductive reasoning,
11–12
sources of echo, 418–429
embedded LDAP directory, 823–825
logon failures, troubleshooting, 827
reconfiguring
on Publisher server, 828–835
on Subscriber server, 835–837
embedded Tracy tool, 105
empty capabilities set, 565
enabling
debugs on Cisco IOS Software, 185–187
fax relay debugs, 455–456
H.323 Fast Connect, 396
encoding schemes, fax machines, 434
endpoint directory access, 821–823
endpoint identifiers (MGCP), 219
endpoints
nonsurvivable, 557
CTI/TAPI endpoints, 559
H.323 gateways, 558–559
Skinny gateways, 557
survivable, 552
IP Phones, 552–553
MGCP gateways, 553–557
end-to-end delay, ITU-T specifications, 384
Enhanced Interior Gateway Routing Protocol (EIGRP), 850
Enhanced Q.931 Translator, 98–100
Enterprise Manager, troubleshooting database replication errors, 802–804
ephone-dn configuration, viewing, 718
Epoch time, 90–91
ER (Emergency Responder), 478
ERL (echo return loss), 417
ERLE (echo return loss enhancement), 417
error codes, Extension Mobility (CallManager 3.3), 777–779
error messages, SMDI, 666
event-specific calling search spaces, 478
“exceeds maximum parties” error messages, 597
extension mobility, 756–758
CallManager 3.1/3.2, 756–772
login/logout process, 763–765
CallManager 3.3, 772–773
error codes, 777–779
login/logout process, 774–777
configuring, 758–763
resolving common problems, 765–772

F
failback behavior in Cisco IP Phones, 156
failed conferences, troubleshooting, 592–597
failover
behavior in Cisco IP Phones, 155
troubleshooting, 158–160
Fast Connect, enabling, 396
Fax Group 3, 433
fax interface-type command, 454
fax machines, 433
encoding schemes, 434
isolating problems, 449–450
jitter, 446
negotiation, 434
NSF field, modifying, 453
packet loss, 446
page transmission speed, 434
physical layer errors on digital interfaces, 447–449
switching fax protocol, 454
T.30 transmissions, 435–437
fax nsf command, 453
fax preamble, 440
fax rate command, 451
fax relay, 444–445
adjusting data rate, 451–452
debugs, enabling, 455–456
ECM, disabling, 452–453
switching to fax passthrough, 450
T.38, 445–446
troubleshooting, 450
fax/modem passthrough, 437, 440
NSE, 439
NTE, 438
troubleshooting, 450
verifying configuration, 441–444
fax-relay ecm disable command, 452
features of CallManager
call hold, 522–529
call park, 531–533
call pickup, 533–538
call transfer, 529–531
fields of CCM traces, 44–46
filtering CCM trace results, 49–50
firewalls, resolving one-way/no-way audio problems, 410
firmware, Cisco IP Phones, 165
fixed delay, 384
coder delay, isolating, 386
effect on signaling, 395–396
packetization delay, isolating, 386–387
propagation delay, isolating, 389
serialization delay, isolating, 387–389
sources of, isolating, 385
formatting
called/calling party numbers with
transformations, 501–506
called/calling party transformations with masks,
495–496
  cumulative effect of changes, 497–499
  order of application, 496–497
forwarding in SRST mode, 731
frame-clock-select command, 210
FXO interface
disconnects, 205
supervisory disconnect tone, 207
FXO/FXS signaling on MGCP gateways, 249–256
FXS (Foreign Exchange Station) gateways, applying
restrictive calling search spaces, 547–548
gateways, Cisco IOS MGCP
  FXO/FXS signaling, 249–256
  PRI backhaul, 256–258
  reading ISDN traces, 258–276
  T1 CAS, 276–281
  verifying registration status, 240–249
gathering data, 4
  analyzing collected data
    case study, 18–19
    CCM traces, 42, 50–60
    CMI traces, 674–679
    deductive reasoning, 11–12
    ISDN traces, 258–262
  case study, 14–18
  earliest occurrence of problem, referencing
device-based time, 10–11
  isolating root cause, 6–9
  user information, 10
  verifying IP network integrity, 12–13
generic media packages (MGCP), 231
Group 3 fax devices, 433
group pickup, 533–538
GSM (Global System for Mobile Communications)
codes, 855
G.711 codecs, 855
  fax passthrough, 437
G.723 codecs, 855
G.726 codecs, 855
G.729 codecs, 855
G.729a codecs, 855
G.729ab codecs, 855
G.729b codecs, 855
garbled audio, sources of
  packet drops, 397–400
  queuing delay, 401
  VAD, 402–404
gatekeeper CAC, 638
call setup, 647–651
  CallManager registration, 645–647
  RAS messages, 639
  verifying configuration, 640–645
H.225 signaling, 283, 850
  call flow, 288–294
  call setup messages, 283–284
H.245, 851
  call signaling, 295
    DTMF relay, 303–307
    logical channel signaling, 300–303
    master/slave determination, 296
    terminal capabilities exchange, 297–300
H.323, 281, 851
gatekeepers, 638
H.225 signaling
  call flow, 288–294
H.245 signaling, 295
  DTMF relay, 303–307
  logical channel signaling, 300–303
  master/slave determination, 296
  terminal capabilities exchange, 297–300
H.225 signaling, 283
  call setup messages, 283–284
IEs, 284–287
nonsurvivable endpoints, 558–559
null capabilities set, 565
versus MG CP, 281
H.323 Fast Connect, enabling, 396
handset emulation packages (MGCP), 235–236
hardware conferencing
  “No Conference Bridge Available” messages, 587–591
  Catalyst 4000 AGM, 587
held calls, 522–525
held party, 602
hexadecimal conversion table, 881–889
high complexity calls, 171
high-compression codecs, 437
  fax relay, 444–445
  T.38, 445–446
holding party, 602
hub-and-spoke topology, locations-based CAC, 624
  AAR, troubleshooting, 637
  analyzing trace information, 628–631
  bandwidth requirements, 624–626
  call preservation, 636–637
  CCM traces, enabling, 626
  conference bandwidth reservations, 633–635
  configuring, 627–631
  detecting bandwidth leaks, 635–636
  location identifier assignments, 628
  MOH bandwidth reservations, 631–633
  regions, 627
HyperTerminal, troubleshooting CMI problems, 679–682
inbound call legs, 175
incoming called-number command, dial peer configuration, 182–184
initiating transactional replication, 804
inline power problems (Cisco IP Phones), troubleshooting unregistered Skinny clients, 114–117
InputDnSignificantDigits parameter (CMI), 668
inside dial tone, 465
installation of Cisco Customer Directory Configuration plugin, troubleshooting, 839–843
intercluster trunks, 311
codec mismatches, 312
interdigit timeout, adjusting, 466–467
international numbers, preventing unauthorized access, 545
Intl TollBypass DDI, 488
Intl TollBypass Trailing-# DDI, 488
investigating sources of delay, 385
  fixed delay, 385
coder delay, 386
  packetization delay, 386–387
  propagation delay, 389
  serialization delay, 387–389
variable delay
dejitter delay, 393–395
low-speed links, 391–393
queuing delay, 390–391
IP addressing
  SRST, DHCP support, 732
  resolving one-way/no-way audio problems, 405–406
  verifying IP Phone configuration, 118–121
IP IVR traces, collecting, 748–752
IP Phones
  auto-registration, controlling, 545–546
call forward fields, 479
  CFA, 480–485
  CFB, 480
  CFF, 485–486
  CFNA, 479–480
directory access, 820
extension mobility, 756–758
configuring, 758–763
login/logout process, 763–765
resolving common problems, 765–768, 772
Skinny client registration
troubleshooting inline power, 114–117
troubleshooting network connectivity, 117–127
verifying, 133
IP Telephony infrastructure, call processing, 24
centralized deployment model, 26
distributed deployment model, 27
multiple-site deployment model, 25
single-site deployment model, 24
IP/VC products, 789
iPlanet (Netscape), LDAP integration, 844
IPV MSApp (Cisco IP Voice Media Streaming Application), software conferencing, 586
ISDN (Integrated Services Digital Network)
Anlagenanschluss, 213
PRI signaling, 210–212
configuring on Cisco IOS voice gateways, 213–214
traces, reading from MGCP gateways, 258–276
isolating
fax problems, 449–450
root cause of problems, 6
case study, 16
with topology information, 7–9
sources of echo, 411
acoustic echo, 412
electrical echo, 411–412
sources of fixed delay, 385
coder delay, 386
packetization delay, 386–387
propagation delay, 389
serialization delay, 387–389
sources of variable delay
dejitter delay, 393–395
low-speed links, 391–393
queuing delay, 390–391
voice quality problems
packet drops, 397–400
queuing delay, 401
VAD, 402–404
ITS (IOS Telephony Services), 707
ITU-T
H.225 specification, 651
Recommendation G.114, delay specifications, 384
web site, 849
IVR (Integrated Voice Response) scripts, 737

J

jitter
effect on fax machines and modems, 446
isolating source of, 391–392
JTAPI (Java Telephony Application Programming Interface), 852
verifying CRA engine status, 745–748

K-L

KeepAliveDn parameter (CMI), 668
LDAP (Lightweight Directory Access Protocol), 852
Active Directory integration, 837–839
corporate directory access, 821–823
Customer Directory Configuration plugin, troubleshooting, 839–844
directories, 819
configuring, 741–743
verifying configuration, 745
directory integration versus directory access, 820
embedded directories, 823–825
logon failures, troubleshooting, 827
reconfiguring on Publisher server, 828–835
reconfiguring on Subscriber server, 835–837
iPlanet integration, 844
LFI (link fragmentation and interleaving), 391
line packages (MGCP), 234–235
line-level calling search spaces, 476–477
listener echo, isolating sources of, 412–413
live audio sources, troubleshooting, 619–620
LMHOSTS file, name resolution, 796–797
local area code versus area code, 510
local calls, delayed routing, 466–469
locating alarm definitions, 93–94
locations-based CAC, 623–624
  bandwidth requirements, 624–626
call preservation, 636–637
CCM traces, enabling, 626
conference bandwidth reservations, 633–635
configuring, 627–631
detecting bandwidth leaks, 635–636
location identifier assignments, 628
MOH bandwidth reservations, 631–633
regions, 627
trace information, analyzing, 628–631
log reader agent, 804
logging call statistics on Cisco IP Phones, 166
logical channel signaling (H.245), 300–303
login/logout process
  Extension Manager (CallManager 3.3), 774–777
  extension mobility, 763–765
low-speed links, isolating delay source, 391–393

M

manual time synchronization, configuring on CallManager servers, 40
  masks, 495–496
master/replica relationship in clusters, 823
master/slave determination in H.245 call signaling, 296
MatchingCgpnWithAttendantFlag service parameter transformation, 500
MCM (Multimedia Conference Manager), 638
media processing resources, 560
media resource group lists (MRGLs), 566, 602
media resource groups (MRGs), 566, 602
media resources, 565
  selecting, 567
medium complexity calls, 171
Meet-Me conferences, 565
  locations-based CAC bandwidth reservations, 633–635
Message Waiting Indicator On/Off Messages (SMDI), 665
MessageDeskNumber parameter (CMI), 668
messages
  “CM Down, Features Disabled,” troubleshooting, 158
  in H.225 call setup, 283–284
  IP Phone status, verifying registration, 133–135
  SCCP, 140, 144–147
    in CCM traces, 148–154
  Skinny client registration process, 127–132
  SMDI, 664–666
  T.30, 435–437
  “Temporary Failure,” troubleshooting dropped calls, 561–562
messaging. See voice mail systems
methodology for resolving call routing problems, 515–516
  reading CCM traces, 516–521
MF packages (MGCP), 232–233
MF trunk packages (MGCP), 237–238
MGCP (Media Gateway Control Protocol), 852
  commands, 219–221
  endpoint identifiers, 218–219
  packages, 229–230
    DTFM package, 231–232
    DTMF trunk package, 236–237
generic media package, 231
  handset emulation package, 235–236
  line package, 234–235
  MF package, 232–233
  MF trunk package, 237–238
  RTP package, 236
  trunk package, 233
  parameter lines, 221–229
  response codes, 239–240
  response headers, 238
See also Cisco IOS MGCP gateways
MGCP gateways, survivable endpoints, 553–557
MGCP T1 PRI gateways, tracing calls, 58–60
Microsoft AD (Active Directory)
  Customer Directory Configuration plugin, troubleshooting, 839–844
  LDAP integration, 837–839
Microsoft Event Viewer, 91
alarm definitions, 92–93
Microsoft PerfMon, 68
alerts, 75
counter logging, 71–75
versus RTMT, 68–69
viewing real-time statistics, 69–71
Microsoft SQL Server Enterprise Manager, 802–803
Replication Monitor
correcting replication errors, 804–806
reestablishing broken replication
subscription, 807–809
reinitializing subscriptions, 809
misconfigured 6608 T1/E1 modules,
troubleshooting, 326–337
MIVR traces, capturing, 748–752
models of Cisco 2600 series routers, 171–172
modems
jitter, 446
packet loss, 446
passthrough, 437
ANS, 439
NSE, 439
NTE, 438
verifying configuration, 441–444
physical layer errors, troubleshooting, 447–449
modules
6608 T1/E1
advanced troubleshooting, 344–359
configuring, 325–337
D-channel establishment, 337, 340–343
T1 CAS, troubleshooting, 359–367
6608/6624 voice gateways
DHCP, troubleshooting, 314–320
powering up, 313–314
registration, troubleshooting, 324–325
TFTP, troubleshooting, 320–324
6624 Port FXS Analog Interface Module,
configuring, 367–379
MOH (Music On Hold). See also TOH
audio sources, 601–603
multicast versus unicast, 615–616
selecting recording input, 620
Audio Translator, troubleshooting, 618
CAC bandwidth reservations, 631–633
performance counters, 915–918
troubleshooting, 611–615
CCM trace files, 608–611
performance counters, 605–607
MOHAudioSourcesActive counter
(CallManager 3.3), 604
MOHConnectionsLost counter
(CallManager 3.3), 606
MOHConnectionState counter
(CallManager 3.3), 604
MOHHighestActiveResources counter
(CallManager 3.3), 607
MOHMulticastResourceActive counter
(CallManager 3.3), 606
MOHMulticastResourceAvailable counter
(CallManager 3.3), 607
MOHOutOfResources counter
(CallManager 3.3), 607
MOHStreamsActive counter (CallManager 3.3), 605
MOHStreamsAvailable counter
(CallManager 3.3), 605
MOHStreamsTotal counter (CallManager 3.3), 606
MOHTotalMulticastResources counter
(CallManager 3.3), 606
MOHTotalUnicastResources counter
(CallManager 3.3), 606
MOHUnicastResourceActive counter
(CallManager 3.3), 606
MOHUnicastResourceAvailable counter
(CallManager 3.3), 607
MRGLs (media resource group lists), 566, 602
MRGs (media resource groups), 566, 602
MTPs (media termination points), null capabilities
set, 565
multicast audio sources (MOH), troubleshooting,
615–616
multiple-site deployment model (CallManager), 25
MWIs (Message Waiting Indicators), 709
configuration parameters, 682–685
toggling on/off, 659, 661
VG248 platform, troubleshooting, 690–692
MwiSearchSpace parameter (CMI), 668
name resolution

LMHOSTS file, 796–797
NetBIOS in database replication, 796–798
NANP (North American Numbering Plan)
call routing information, 857–879
route filters, 506–510
multiple clauses, 512
tags, 507
NAT (Network Address Translation), resolving
one-way/no-way audio problems, 410
negotiation process, fax machines, 434
NetBIOS name resolution in database replication, 796–798
Netscape iPlanet, LDAP integration, 844
network diagrams, required information, 7–9
network hold MOH audio source, 601
network integrity, verifying, 12–13
network settings, Cisco 79xx IP Phones, 123–126
Network Time Protocol. See NTP
“No Conference Bridge Available,” troubleshooting,
587–591
NoDigits DDI, 488
nonproduction hours, troubleshooting methodologies, 5–6
nonsurvivable endpoints, 557
CTI/TAPI endpoints, 559
H.323 gateways, 558–559
Skinny gateways, 557
no-way audio, isolating sources of
Cisco IOS Software gateways, 408–410
firewalls, 410
IP connectivity, 405–406
NAT, 410
PAT, 410
NSE (Named Service Event), 439
NSF (Nonstandard Facilities) field, modifying, 453
NTE (Named Telephony Event), 438
NTP (Network Time Protocol), 852
time synchronization, 39
on CatOS devices, 41
on Cisco IOS devices, 40–41
null capabilities set, 565

numbering plans
NANP, call routing information, 857–879
route filters, 506–510
multiple clauses, 512
NANP tags, 508–510

object counters
Cisco analog access, 892
Cisco CallManager, 893–898
Cisco CallManager Attendant Console, 898–900
Cisco CallManager System Performance, 900–902
Cisco CIT Manager, 903
Cisco Gatekeeper, 904
Cisco H.323, 904
Cisco HW Conference Bridge Device, 905
Cisco Lines, 905
Cisco Locations, 906
Cisco Media Streaming App, 906–909
Cisco Media Termination Point, 909–910
Cisco Messaging Interface, 910–911
Cisco MGCP FXO Device, 911
Cisco MGCP FXS Device, 912
Cisco MGCP Gateways, 912
Cisco MGCP PRI Device, 913–914
Cisco MGCP T1 CAS Device, 914–915
Cisco MOH Device, 915–918
Cisco MTP Device, 916
Cisco Phones, 918
Cisco SW Conference Bridge Device, 918–920
Cisco TFTP, 920–923
Cisco Transcoder Device, 923
logging on PerfMon, 71–75
Windows 2000, 924–925
obtaining
Dick Tracy tool, 105
Enhanced Q.931 Translator, 100
Octel voice mail systems, CallManager integration, 693, 698–700
OffHookMessage message, SCCP call processing, 144
one-way audio, isolating sources of
- Cisco IOS Software gateways, 408–410
- firewalls, 410
- IP connectivity, 405–406
- NAT, 410
- PAT, 410
- open trees, 673
operating systems, Windows 2000
- CCEmail, 76–81
- performance counters, 924–925
operation of echo cancellers, 416–418
optional dial peer parameters, 179–181
OSI reference model, verifying connectivity at every layer, 12–13
OSPF (Open Shortest Path First), 852
outbound call legs, 175
out-of-resource conditions, 578–580
OutputDnFor parameter (CMI), 668
OutputExternalFormat parameter (CMI), 669
OverlapReceivingForPriFlag service parameter transformation, 501
overwriting transformations, 499

P

packages (MGCP), 229–230
- DTMF, 231–232
- DTMF trunkRTP, 236–237
generic media, 231
- handset emulation, 235–236
- line, 234–235
- MF, 232–233
- MF trunkRTP, 237–238
- RTP, 236
- trunk, 233
packet drops
- as source of voice quality degradation, 397–400
effect on fax machines and modems, 446
packet-capture software, 106
packetization delay, isolating, 386–387
page transmission speed, 434
parameter lines (MGCP), 221–229
Parity parameter (CMI), 669
parked calls, 531
- troubleshooting, 532–533
partitions, 469–470. See also calling search spaces
calling search spaces, AAR, 637
identifying potential route pattern matches, 517–519
pattern-matching rules, 474–475
passwords, configuring on cluster nodes, 798–802
PAT, resolving one-way/no-way audio problems, 410
pattern matching. See also calling search spaces
blocked calls, 473
closest-match routing, 461–464
delayed routing, troubleshooting, 466–469
multiple partitions within a calling search space, 474–475
variable-length, 178–179
wildcards, 460–461
PBXs (private branch exchanges), troubleshooting
calling name display problems, 270
perception of echo as problematic, 414–416
PerfMon, 68. See also CCEmail
alerts, 75
counter logging, 71–75
versus RTMT, 68–69
- viewing real-time statistics, 69–71
performance counters
- Cisco analog access, 892
- Cisco CallManager, 893–898
- Cisco CallManager Attendant Console, 898–900
- Cisco CallManager System Performance, 900–902
- Cisco CIT Manager, 903
- Cisco Gatekeeper, 904
- Cisco H.323, 904
- Cisco HW Conference Bridge Device, 905
- Cisco Lines, 905
- Cisco Locations, 906
- Cisco Media Streaming App, 906–909
- Cisco Media Termination Point, 909–910
- Cisco Messaging Interface, 910–911
- Cisco MGCP FXO Device, 911
- Cisco MGCP FXS Device, 912
- Cisco MGCP Gateways, 912
- Cisco MGCP PRI Device, 913–914
- Cisco MGCP T1 CAS Device, 914–915
- Cisco MOH Device, 915–918
- Cisco MTP Device, 916
Performance tab (RTMT), 86
phone registration, SRST, 712–717
phone services, 789
physical layer
   connectivity, verifying on digital interfaces, 208–210
   troubleshooting fax/modem errors, 447–449
pinpointing earliest occurrence of problems, 10
referencing device-based time, 11
plain-text protocols, 218
PLAR (Private Line Automatic Ringdown),
   controlling IP phone auto-registration, 545–546
plugins, Cisco Customer Directory Configuration,
   troubleshooting installation, 839–843
polling statistics (SRST), viewing, 722
POTS dial peers, 175
   variable-length pattern matching, 179
power denial, 206
powering 6608/6624 voice gateway modules, 313–314
PreAt 10-10-Dialing DDI, 489
PreAt 10-10-Dialing Trailing-# DDI, 489
PreAt 11/10D->7D DDI, 490
PreAt 11/10D->7D Trailing-# DDI, 490
PreAt 11D->10D DDI, 491
PreAt 11D->10D Trailing-# DDI, 491
PreAt DDI, 489
PreAt Intl TollBypass DDI, 492
PreAt Intl TollBypass Trailing-# DDI, 492
PreAt Trailing-# DDI, 489
PreDot 10-10-Dialing DDI, 493
PreDot 10-10-Dialing Trailing-# DDI, 493
PreDot 11/10D->7D DDI, 493
PreDot 11/10D->7D Trailing-# DDI, 493
PreDot 11D->10D DDI, 494
PreDot 11D->10D Trailing-# DDI, 494
PreDot DDI, 487, 492
PreDot Intl TollBypass DDI, 494
PreDot Intl TollBypass Trailing-# DDI, 495
PreDot IntlAccess IntlDirectDial DDI, 494
PreDot Trailing-# DDI, 493
preventing
   service-affecting problems, 5–6
toll fraud, 544–549
PRI backhaul on MGCP gateways, 256–258
primary CallManager, 154
processing delay, isolating, 386
production hours, troubleshooting methodologies, 5–6
progress tones, 307
propagation delay, isolating, 389
Publisher server
   DC, reconfiguring, 828–835
   Replication Monitor
      correcting replication errors, 804–806
      reestablishing broken replication subscription, 807–809
      reinitializing subscriptions, 809
Publisher-Subscriber model
   database replication, 793–796
   name resolution
      LMHOSTS file, 796–797
      NetBIOS, 796–798
   Subscriber server, configuring CDR replication, 810–812

Q

Q.850, 852
Q.921, 853
Q.931 Translator, 95–97. See also Enhanced Q.931 Translator
queuing delay
   as source of voice quality degradation, 401
   isolating source of, 390–391
RAS (Registration, Admission, and Status) messages, 639, 853
reading traces
  CCM traces, 42, 50–57
  through MGCP T1 PRI gateways, 58–60
  CMI traces, 674–679
ISDN traces from MGCP gateways, 258–262
calling name display, 270
cause codes, 262–269
numbering type/plan mismatches, 269–270
timer information, 271–276
SDL traces, 60–63
Real-Time Monitoring Tool, monitoring MOH performance counters, 605–607
real-time statistics, viewing with PerfMon, 69–71
reconfiguring DC
  on Publisher server, 828–835
  on Subscriber server, 835–837
recording input of live audio sources, selecting, 620
redirecting calls, group pickup, 533–538
reestablishing broken replication subscription, 807–809
regions, 571
codec configuration, 568–569
codec matrix, 571–577
configuring for locations-based CAC, 627
registration (Skinny clients)
  608/6624 modules, 324–325
checking phone status display, 133
inline power, troubleshooting, 114–117
messages, 127–132
network connectivity, 117, 120–127
  configuration files, 121–127
  IP addressing, 118–121
  VLAN configuration, 118
verifying with IP Phone status messages, 133–135
reinitializing subscriptions, 809
remote access tools
  VNC, 108
  Windows Terminal Services, 107
replication
  correcting with replication agents, 804–806
  name resolution, 796–797
  of CDRs
    configuring, 810–812
    troubleshooting, 813–815
  passwords, configuring on cluster nodes, 798–802
  Publisher-Subscriber model, 793–796
  reestablishing broken subscription, 807–809
  troubleshooting with Microsoft SQL Server Enterprise Manager, 802–803
Replication Monitor
  reestablishing broken replication subscription, 807–809
  reinitializing subscriptions, 809
  troubleshooting replication errors, 804–806
resetting
  Cisco IP Phones, 156
  NSF field, 453
resolving call routing problems, 515–516
  reading CCM traces, 516–521
  response codes, 239–240
  response headers, 238
restarting Cisco IP Phones, 156
restrictions of SRST, 708
ringback, troubleshooting absence of, 307
during call transfer, 309
on IP phones calling PSTN, 308
on PSTN phones calling IP phones, 309
RIP (Routing Information Protocol), 853
robbed-bit signaling, 214
rollover cables, 679
route filters, 506–507
  multiple clauses, 512
  NANP tags, 508–510
route patterns. See also translation patterns
  closest-match routing, 461–464
  pattern-matching, delayed routing, 466–469
  urgent priority, 502
  wildcards, 460–461
Route Plan Report, viewing in Cisco CallManager Administration, 466
RouteFilter parameter (CMI), 669
routers, voice gateway functionality
Cisco 2600 series, 171–172
Cisco 3600 series, 172
Cisco 3700 series, 173
routing calls to voice mail (SRST), 731
RSVP (Resource Reservation Protocol), 853
RTMT (Real-Time Monitoring Tool), 85
CTI Apps tab, 88
Devices tab, 86
Performance tab, 86
verifying Skinny client registration, 135–137
RTP (Real-Time Protocol), 853
dropped calls, 551–552
packages (MGCP), 236
RUDP (Reliable User Datagram Protocol), 853

S
sa (system administrator) user account, changing password, 802
sample CCM trace, 51
SCCP (Skinny Client Control Protocol), 139
messages
analyzing in CCM traces, 148–154
call processing, 140, 144–148
DEVICE_RESET, 157
DEVICE_RESTART, 157
Skinny client registration, 127–135
608/6624 modules, 324–325
checking phone status display, 133
configuration files, 121–127
inline power, troubleshooting, 114–117
IP addressing, 118–121
messages, 127–132
network connectivity, troubleshooting, 117, 120–127
verifying with RTMT, 135–137
verifying with status messages, 133–135
scheduled outages, preventing service-affecting problems, 5–6
SDI traces, reading, 42, 50–57
SDKs, Cisco IP Phone Services, 822
SDL traces
configuring, 63–67
reading, 60–63
troubleshooting held calls, 527–529
secondary CallManager, 154
selecting
appropriate troubleshooting tools, 13
MRGLs, 567
recording input for live audio sources, 620
serialization delay, isolating, 387–389
SerialPort parameter (CMI), 669
Service Activation, configuring on CallManager
Serviceability, 84
service parameters
CMI, 667–671, 674
transformations, 500–501
VG248, 686–690
service-affecting problems, preventing, 5–6
services button (Cisco IP Phones), request failures, 160–161
SGCP (Skinny Gateway Control Protocol), 853
shared lines, calling search spaces, 477
show call active voice brief command, 449
show call active voice command, 191, 404
show ephone command, 720
show gatekeeper calls command, 649
show gatekeeper endpoints command, 645
show gatekeeper zone status command, 649
show voice port summary command, 188–189
signaling
H.225, 283–284
problem isolation, 18
silence suppression. See VAD
single-site deployment model (CallManager), 24
Skinny Client Control Protocol. See SCCP
Skinny clients, registration
608/6624 modules, 324–325
checking phone status display, 133
configuration files, 121–127
inline power, troubleshooting, 114–117
IP addressing, 118–121
messages, 127–132
network connectivity, troubleshooting, 117, 120–127
verifying with RTMT, 135–137
verifying with status messages, 133–135
Skinny gateways, nonsurvivable endpoints, 557
SMDI (Simple Message Desk Interface), 854
CallManager integration, 662–666
messages, 664–666
MWI, configuration parameters, 682–685
VG248 SMDI integration, 686
configuration parameters, 686–690
MWI problems, troubleshooting, 690–692
snapshot agent, 804
sniffer traces, 106
soft keys, 147
events, 522–523
software conferencing
“No Conference Bridge Available” messages, troubleshooting, 587–591
IPV MSApp, 586
sources of delay, investigating
fixed delay, 385
coder delay, 386
packetization delay, 386–387
propagation delay, 389
serialization delay, 387–389
variable delay
dejitter delay, 393–395
low-speed links, 391–393
queuing delay, 390–391
sources of echo
eliminating, 418–429
isolating, 411
acoustic echo, 412
electrical echo, 411–412
SQL servers
database replication
changing passwords, 798–802
name resolution, 796–798
Microsoft SQL Server Enterprise Manager, 802–803
Publisher-Subscriber model, 793–796
SRST (Survivable Remote Site Telephony), 562
call control, debugging, 719–720
call transfer, debugging, 729–730
configuring, 709–712
CoR, 708
DHCP support, 732
ephone-dn configuration, viewing, 718
forwarding calls, 731
phone registration, 712–717
polling, 722
restrictions, 708
routing calls to voice mail system, 731
transfer patterns, configuring, 730
SsapiKeepAliveInterval parameter (CMI), 669
standards
ITU-T H.225 specification, 651
standby CallManager, 154
StationActivateCallPlane message, SCCP call processing, 144
StationAlarmMessage, 129
field definitions, 158–160
StationCallInfo message, SCCP call processing, 145
StationCallState message, SCCP call processing, 145
StationClearNotify message, SCCP call processing, 146
StationClearPromptStatus message, SCCP call processing, 146
StationCloseReceiveChannel message, SCCP call processing, 146
StationConnectionStatisticsRequest message, SCCP call processing, 146
StationConnectionStatisticsResponse message, SCCP call processing, 147
StationDisplayNotify message, SCCP call processing, 145
StationDisplayPromptStatus message, SCCP call processing, 144
StationKeepAliveAck messages, 129
StationKeepAliveMsg, 129
StationKeypadButtonMessage message, SCCP call processing, 144
StationOpenReceiveChannel message, SCCP call processing, 146
StationOpenReceiveChannelAck message, SCCP call processing, 146
StationOutputDisplayText message, SCCP call processing, 144
StationRegisterAck messages, 129
StationRegisterMessage, 129
StationRegisterReject messages, 129
StationSelectSoftKeys message, SCCP call processing, 144
StationSetLamp message, SCCP call processing, 144
StationSetRinger message, SCCP call processing, 145
StationSetSpeakerMode message, SCCP call processing, 146
StationSoftKeyEventMessage message, SCCP call processing, 147
StationStartMediaTransmission message, SCCP call processing, 146
StationStartTone message, SCCP call processing, 144
StationStopMediaTransmission message, SCCP call processing, 146
StationStopTone message, SCCP call processing, 146
status messages (IP Phones), verifying Skinny client registration, 133–135
StopBits parameter (CMI), 669
StripPoundCalledPartyFlag service parameter transformation, 501
Subscriber server, 794
CDR replication, configuring, 810–812
dc, reconfiguring, 835–837
subscriptions, reinitializing, 809
substrings, 507
tags, 507–510
supervisory disconnect tone, 207–208
survivable endpoints, 552
dropped calls, troubleshooting, 561–562
IP Phones, 552–553
MGCP gateways, 553–557
switching, fax protocol, 454

T
T.30 fax transmissions, 435–437, 854
T.38 fax relay, 445–446, 854
T1 CAS (Channel Associated Signaling), troubleshooting
on 6608 module, 359–367
on Cisco IOS voice gateways, 214–218
on MGCP-enabled ports, 276–281
tagS, 507–510
tail circuits, 416
talker echo, isolating sources of, 412–413
TAPI (Telephony Application Programming Interface), 854
TCP (Transmission Control Protocol), 854
backhauling, 554
failback, 156
failover, 155
tCP handle, 52
TDM interfaces
ISDN PRI, 210–212
configuring on Cisco IOS voice gateways, 213–214
on Cisco IOS voice gateways, 187
debug commands, 192–205
show commands, 187–192
telcordia web site, 849
temporary dial peers, viewing, 719
Temporary Failure messages (IP Phones), 561–562
terminal capabilities exchange in H.245 call signaling, 297–300
terminal emulation, troubleshooting HyperTerminal CMI problems, 679–682
terminal Services, 107
TFTP (Trivial File Transfer Protocol), 854
configuration files, 154
troubleshooting on 6608/6624 modules, 320–324
third-party voice mail systems, applying restrictive calling search spaces, 547–548
time synchronization, 38
on CallManager servers, 39–40
on CatOS devices, 41
on Cisco IOS devices, 40–41
timestamps
configuring, 185
on CDRs, 90–91
toggling MWI on/off, 659, 661
TOH (tone on hold), 602
investigating instances of, 617
toll fraud, preventing, 544–549
topologies, required documentation, 9
traces. See also CDRs
CCM
analyzing SCCP messages, 148–154
call state field values, 525
configuring for CallManager serviceability, 42–50
digit analysis results, 149
fields, 44–46
reading, 42, 50–57
reviewing for call routing problems, 516–521
through MGCP T1 PRI gateways, 58–60
CMI, reading, 674–679
configuring
for locations-based CAC, 626
for CallManager Serviceability, 83
dialing forests, 538–542
verbose mode, 543
IP IVR/AA, capturing, 748–752
ISDN, analyzing, 258–276
MOH, troubleshooting, 608–611
SDL
configuring, 63–67
reading, 60–63
troubleshooting held calls, 527–529
sniffer traces, 106
Trailing-# DDI, 495
training, 434
transactional replication, 794
initiating, 804
transcoders, 565, 571–577
out-of-resource conditions, 578–580
with conference bridge resources, 581–585
with MOH servers, 585
transfer patterns, configuring, 730
transferred calls, 529–531
transformations, 513–514
DDIs, 486–494
overriding, 499
rules
cumulative effect of changes, 497–499
order of application, 496–497
service parameter-related, 500–501
translation patterns, 501–506
translation patterns, 501–506
transmission rates
fax devices, 434
fax relay, adjusting, 451–452
transmitting faxes through voice codecs, 437
troubleshooting methodologies
data analysis, case study, 18–19
data collection, 4–5
analyzing collected data, 11
case study, 14–18
earliest occurrence of problem, 10–11
identifying root cause of problem, 6
isolating root cause of problem, 7–9
user information, 10
production versus nonproduction outages, 5–6
trunk packages (MGCP), 233
trunks, intercluster, 311
codec mismatches, 312
TSP (TAPI service provider)
verifying compatibility with Cisco Unity, 655–656
verifying configuration, 656–657
UDP (User Datagram Protocol), 854
umbrella recommendations, H.323, 281
unanswered calls, forwarding, 479–480
unauthorized access to international numbers, preventing, 545
unexpected outside dial tone, troubleshooting, 465–466
unicast audio sources (MOH), troubleshooting, 615–617
Unity voice mail systems
applying restrictive calling search spaces, 547
DTMF, 661–662
MWI, 659–661
troubleshooting resources, 662
verifying switch configuration, 658–659
verifying TSP compatibility, 655–656
verifying TSP configuration, 656–657
UnknownCallerId service parameter
transformation, 501
UnknownCallerIdFlag service parameter
transformation, 501
UnknownCallerIdText service parameter
transformation, 501
unregistered IP Phones, tracing, 131–132
unregistered Skinny clients
troubleshooting inline power problems, 114–117
troubleshooting network connectivity, 117, 120–127
configuration files, 121–127
IP addressing, 118–121
VLAN configuration, 118
urgent priority route patterns, 502
user hold audio source (MOH), 601
user information, collecting, 10
user search requests, directory access, 820
UseZerosForUnknownDn parameter (CMI), 670
utilities, CDR Time Converter, 91.

See also applications

V

V.21 HDLC, 854
VAD (voice activity detection)
   as source of voice quality degradation, 402–404
   comfort noise, 402
ValidateDns parameter (CMI), 670
variable delay, 384
   dejitter delay, isolating, 393–395
   effect on signaling, 395–396
   low-speed links, isolating, 391–393
   queuing delay, isolating, 390–391
variable-length matching (dial peers), 178–179
VAT (Voice Anomaly Tracking), 166
verbose dialing forest traces, 543
verifying
   CAC configuration, 640–645
   Cisco IOS MGCP registration status, 240–249
   Cisco IP Phone firmware, 165
   Cisco Unity switch configuration, 658–659
   CRA engine status, 745–748
   Database Layer Monitor operation, 812–813
   fax/modem passthrough configuration, 441–444
   IP network integrity, 13
   LDAP directory configuration, 745
   MOH fixed audio source device configuration, 619–620
   physical layer connectivity on digital interfaces, 208–210
   Skinny client registration with RTMT, 135–137
   SRST configuration, 709–712
   TSP compatibility with Cisco Unity, 655–656
   TSP configuration, 656–657
   TSP version on CTI applications, 736
VG200 voice gateway, 170
VG248 voice gateway, 521
SMDI integration, 686
   configuration parameters, 686–690
   MWI problems, troubleshooting, 690–692
viewing
   ephone-dn configuration, 718
   real-time statistics with PerfMon, 69–71
   Route Plan Report in Cisco CallManager Administration, 466
   SRST polling statistics, 722
   virtual dial peers, viewing, 719
   VNC (Virtual Computer Networking), 108
   Voice Codec Bandwidth Calculator, 106
   voice codecs, fax/modem passthrough, 437
   voice gateways
      Catalyst
      Catalyst 4224, 173–174
      Catalyst 6000 CMM, 174
      configuring 6624 Analog Interface Module, 367–379
      Cisco AVVID IP Telephony, 32
      Cisco IOS, 169
         2600 series routers, 171–172
         3600 series routers, 172
         3700 series routers, 173
         H.323, 281–307
         MGCP, 218–240
         T1 CAS, troubleshooting, 214–218
         timestamps, configuring, 185
         troubleshooting TDM interfaces, 187–205
      VG200, 170
      Dick Tracy tool, 101–104
      CLI/embedded Tracy, 105
      FXO interface, troubleshooting disconnects, 205
voice mail systems
   applying restrictive calling search spaces, 547
   Cisco Unity, 655
      DTMF, 661–662
      MWI, 659–661
      troubleshooting resources, 662
      verifying switch configuration, 658–659
      verifying TSP compatibility, 655–656
      verifying TSP configuration, 656–657
CMI, 666–667
  service parameters, 667–671, 674
  traces, reading, 674–679
  troubleshooting with HyperTerminal, 679–682
Octel, CallManager integration, 693, 698–700
SMDI
  CallManager integration, 662–666
  messages, 664–666
  MWI, 682–685
  VG248 integration, 686–692
voice quality
  choppy audio, isolating sources of, 397–404
  echo
    acoustic echo, 412
    electrical echo, 411–412
    eliminating sources of, 418–429
    isolating sources of, 411
    perception of as problem, 414–416
  one-way/no-way audio, isolating sources of, 405–410
voice streaming
  dropped calls
    media processing resources, 560
    RTP/UDP, 551–552
  nonsurvivable endpoints, 557
    CTI/TAPI endpoints, 559
    H.323 gateways, 558–559
    Skinny gateways, 557
  survivable endpoints, 552
    IP Phones, 552–553
    MGCP gateways, 553–557
VoiceMailDn parameter (CMI), 670
VoiceMailPartition parameter (CMI), 670
VoIP dial peers, 175
  variable-length pattern matching, 179
VSTP (Voice Telephony Service Provider) states, 190–191
  debug commands, 193–196

W-X-Y-Z

WANs, fax relay, 444–445
wideband codecs, 855
wildcards, 460
  ! wildcard, 460
  . wildcard, 461
  @ wildcard, 461
    DDIs, 487–494
    route filters, 506–512
    multiple clauses, 512
  X wildcard, 460
NANP tags, 508–510
Windows 2000
  CCEmail
    alerting methods, 81
    configuring, 76–80
    object counters, 924–925
Windows Terminal Services, 107
Wink Start (E&M), 850
winks, 214
WS-X6608 module, 587

X wildcard, 460